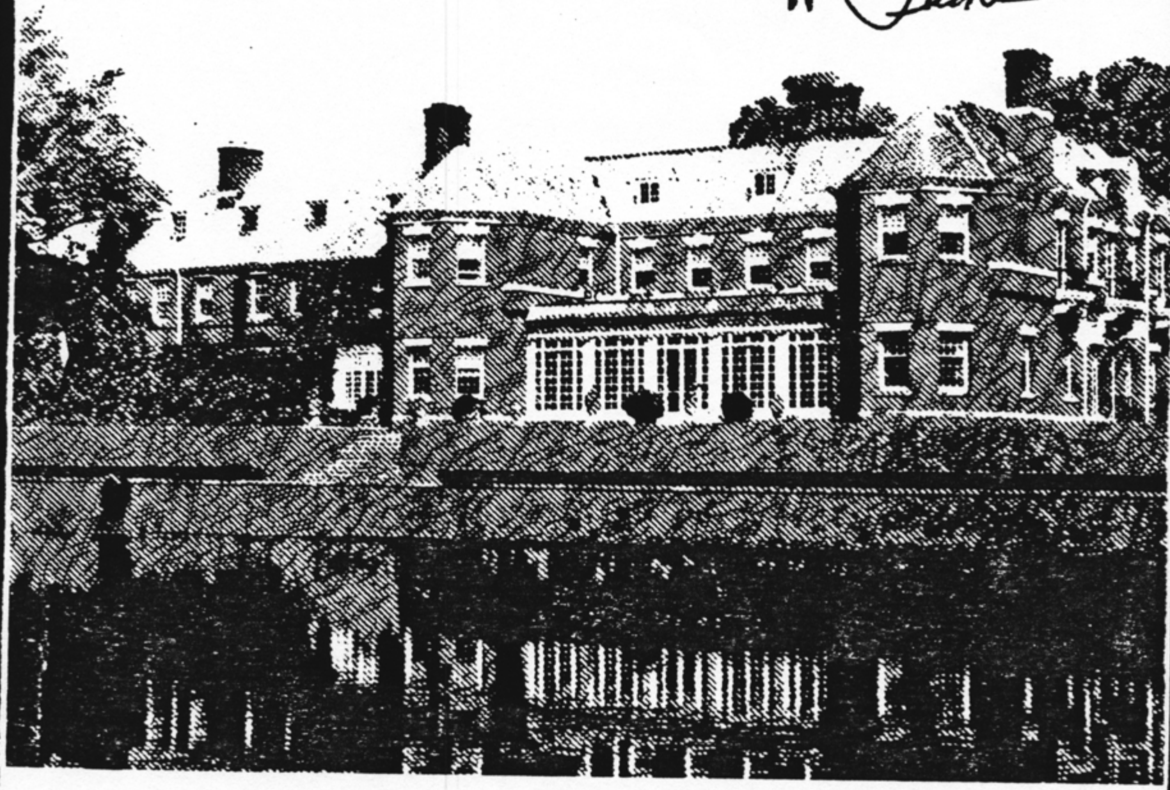

# Proceedings

## EIGHTEENTH ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

October 8-10, 1980

Allerton House, Monticello, Illinois
Sponsored by the
Department of Electrical Engineering and the
Coordinated Science Laboratory of the
University of Illinois at Urbana-Champaign

# ON FINDING SEVERAL SHORTEST PATHS IN CERTAIN GRAPHS*

(Preliminary Report)

ZVI M. KEDEM
Department of Computer Science
S.U.N.Y. at Stony Brook
Stony Brook, New York 11794

and

HENRY FUCHS
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, North Carolina 27514

## ABSTRACT

Given a graph, the problem of finding shortest paths from $u_1^{(i)}$ to $u_2^{(i)}$ for a set of pairs of vertices $\{(u_1^{(i)}, u_2^{(i)}) \mid i=1,\ldots,m\}$ is considered.

A fast algorithm for a special class of problems is presented. The algorithm is applied to the Circular String-to-String Correction Problem.

*Research partially supported by NSF under Grants Number MCS-7901168 and MCS-7902593, and facilitated by the use of Theory Net, NSF Grant Number MCS-7801689.

## GENERAL THEOREM

The problem of finding shortest paths in graphs has been studied extensively. Two main variants have been of interest:

I. For one pair of vertices $(u,w)$ in the graph find a shortest path from $u$ to $w$.

II. For all pairs of vertices $(u,w)$ in the graph find a shortest path from $u$ to $w$.

Well-known methods exist for dealing with these variants.

We consider in this paper another possible variant, which falls between the two variants above:

III. For several pairs of vertices $\{(u_1^{(i)}, u_2^{(i)}) \mid i=1,\dots,m\}$ in the graph find a shortest path from each $u_1^{(i)}$ to its corresponding $u_2^{(i)}$.

Let then $G = \langle V,E,L\rangle$ be an undirected or a directed and acyclic, labeled graph, where $V$ is the set of vertices, $E$ the set of edges, and $L$ the labeling function assigning to each element of $E$ a real number (length), which will be non-negative if $G$ is undirected.

Variant I is normally solved by algorithms which require $O(|V|^2)$ or $O(|E| \cdot \log_2 |V|)$ operations in the worst case, and variant II is normally solved by algorithms which require $O(|V|^3)$ or $O(|E| \cdot \log_2 |V|)$ operations in the worst case. (See e.g. [AHU].)

A solution of variant III may be obtained by either solving an instance of variant I for each of the given pairs or by solving variant II and extracting the required shortest paths. In this paper we will describe a method which can be used to design efficient algorithms for solving variant III. We restrict the development here only to directed acyclic graphs, although the results also hold for undirected graphs. As we will be concerned here solely with directed graphs we will refer to edges as arcs.

For a graph $G = \langle V,E\rangle$ and $V' \subseteq V$ we shall denote by $G(V')$ the subgraph of $G$ spanned by $V'$. A path in $G$ will be written as a sequence of vertices; a path will also be considered as a subgraph of $G$ spanned by the traversed arcs.

Assume that shortest paths from $u_1^{(i)}$ to $u_2^{(i)}$ for $i = 1,2$ are to be found. If a shortest path $P(1)$ from $u_1^{(1)}$ to $u_2^{(1)}$ is found first then, as we shall see, the search for a shortest path $P(2)$ from $u_1^{(2)}$ to $u_2^{(2)}$ may be limited to a subgraph of $G$. If one has to find more than two shortest paths, then possibly even "smaller" subgraphs of $G$ may be considered during the search for some paths.



<u>Theorem 1</u>: Let $P_0 = (u_1, u_2, \dots, u_p)$ be a shortest path from $u_1$ to $u_p$ in a directed acylic graph $G$. Let $V_0 \hat{=} \{u_1, u_2,\dots, u_p\}$, and let the weak (undirected) components of $G(V-V_0)$ be spanned by the (pairwise disjoint) sets of vertices $V_1, V_2,\dots, V_r$. If $v_a \in V_i$ and $v_b \in V_j$ for $0 \le i, j \le r$ are such that $v_b$ is reachable from $v_a$, then there exists a shortest path $P = (v_a = w_1, w_2,\dots, w_q = v_b)$ in $G$ from $v_a$ to $v_b$ such that

1. $w_k \in V_0 \cup V_i \cup V_j$, $k = 1, 2,\dots, q$,
2. if $w_k, w_{k+1} \in V_0$ for some $k = 1, 2,\dots, q-1$, then $w_k = u_\ell$ and $w_{k+1} = u_{\ell+1}$ for some $\ell = 1, 2,\dots, p-1$. ■

For $P_0, V_0, v_a, V_i, v_b, V_j$ as in Theorem 1, $G(P_0, v_a, v_b)$ will denote the subgraph of $G$ to which the search for a shortest path from $v_a$ to $v_b$ and be limited. More formally, if $\tilde{V} = V_0 \cup V_i \cup V_j$ and $P_0 = \langle V_0, E_0\rangle$ ($E_0$ consists of the arcs traversed by the path $P_0$) then $G(P_0,v_a,v_b) \hat{=} \langle \tilde{V}, (\tilde{V}^2 \cap E) - (V_0^2 - E_0)\rangle$.

Theorem 1 can be used to design efficient algorithms for finding shortest paths for several pairs of vertices. Indeed, assume that are to find shortest paths from $u_1$ to $u_p$ and from $v_a$ to $v_b$. Once a shortest path $P_0$ from $u_1$ to $u_p$ has been found, one may search for a shortest path from $v_a$ to $v_b$ in $G(P_0, v_a, v_b)$ only. If additional pairs of vertices are given, then possibly even smaller subgraphs of $G$ have to be considered. The subgraphs to which the search may be limited, $G(P_0, v_a, v_b)$ and the appropriate subgraphs for the other pairs, may easily be computed in a linear number of operations but any possible savings greatly depend on $G$ and the given pairs of vertices, as in the general case there is no reason to assume that these subgraphs are "smaller" than the original graph $G$.

## APPLICATION TO PLANAR GRAPHS

We shall now develop the use of the Theorem for certain cases in which it can be assured that the size of the subgraphs to be considered rapidly decreases as paths for additional pairs of vertices are found. As there can be no path in $G$ whose terminals lie in distinct components of a graph, we shall assume, without loss of generality, that $G$ consists of a single component.

Let $G$ now be a planar graph, which is identified with an actual embedding in the plane. A sequence of vertices $(u_1,u_2,\dots,u_r)$ will be called a <u>circuit</u> in $G$ if and only if it is an (undirected) walk in $G$ and $u_r = u_1$. Each such circuit $\alpha = (u_1,u_2,\dots,u_r)$ defines a closed curve $\ell_\alpha$ on the plane which is defined by the sequence of the edges "traversed" by $\alpha$. Let $V^{(\alpha)}$ be the set of those vertices of $G$ which lie on or inside $\ell_\alpha$. We define $G_\alpha$, a subgraph of G, to be the subgraph of $G(V^{(\alpha)})$ obtained from $G(V^{(\alpha)})$ by removing all those edges which lie outside of $\ell_\alpha$. (See Fig. 1.) A circuit $\alpha$ will be called <u>non-crossing</u> if and only if the corresponding curve $\ell_\alpha$ does not cross itself. (It may "touch" itself.)

Without loss of generality, we may assume that an arbitrary contour is the contour of the infinite region of G and thus the following development, even though formulated for the contour $\Gamma_0$ of the infinite region, holds with appropriate changes for the contour of an arbitrary region.

A sequence $(z_1,z_2,\ldots,z_h)$ of vertices in G is a <u>list</u> for $\Gamma_0$ if and only if it is a non-crossing circuit in G and h is minimal such that the set $\{z_i | i=1,\ldots,h\}$ is the set of all the vertices in $\Gamma_0$.

Let now $(z_1,z_2,\ldots,z_h)$ be a list for $\Gamma_0$ and let $P = (z_s=w_1,w_2,\ldots,w_r=z_t)$ be a path from $z_s$ to $z_t$ for $1\le s,t\le h$. Note that as the graph is acyclic P cannot "cross" itself. Define the sequences

$$\Gamma_0 \oplus P \hat{=} \begin{cases} (z_s,z_{s+1},\ldots,z_t=w_r,w_{r-1},\ldots,w_1=z_s) & ; s \le t \\ (z_s=w_1,w_2,\ldots,w_r=z_t,z_{t+1},\ldots,z_s) & ; s > t \end{cases}$$

$$P \oplus \Gamma_0 \hat{=} \begin{cases} (z_s=w_1,w_2,\ldots,w_r=z_{t+1},\ldots,z_h=z_1,\ldots,z_s) & ; s \le t \\ (z_s,z_{s+1},\ldots,z_h=z_1,\ldots,z_t=w_r,w_{r-1},\ldots,w_1=z_s); & s > t \end{cases}$$

$\Gamma_0\oplus P$ and $P\oplus\Gamma_0$ are non-crossing circuits and are furthermore lists for the contours of the infinite regions of the subgraphs $G_{\Gamma_0\, P}$ and $G_{P\,\Gamma_0}$ of G respectively. (See Fig. 2.)

<u>Corollary 1</u>: Let $(z_1,z_2,\ldots,z_h)$ be a list for $\Gamma_0$ and let $z_s$ and $z_t$ for $1\le s\le t\le h$ be terminals of a shortest path P. If a and b are any integers such that $\{a,b\} \subseteq \{s,s+1,\ldots,t\}$ (or $\{a,b\} \subseteq \{t,t+1,\ldots,h,1,\ldots,s\}$ respectively), and $z_b$ is reachable from $z_a$, then there exists a shortest path (in G) from $z_a$ to $z_b$ which is wholly contained in $G_{\Gamma_0\oplus P}$ (or $G_{P\oplus\Gamma_0}$ respectively). ■

The Corollary defines in a natural way a family of problems for which it can be utilized for the design of a "divide and conquer" algorithm.

Let again $(z_1,z_2,\ldots,z_h)$ be a list for $\Gamma_0$. For some $m > 0$ let $\phi$ be a nondecreasing function from $\{1,2,\ldots,2m\}$ to $\{1,2,\ldots,h\}$ and let $\delta$ be a function from $\{1,2,\ldots,m\}$ to $\{0,1\}$. For every k, $k=1,2,\ldots,m$, let $z_k^{(2)} = z_{\phi(2m+1-k)+\delta(k)\cdot(\phi(k)-\phi(2m+1-k))}$ be reachable from $z_k^{(1)} = z_{\phi(k)+\delta(k)\cdot(\phi(2m+1-k)-\phi(k))}$. (See Fig. 3) We are to find $P(1),P(2),\ldots,P(m)$ where $P(k)$ is a shortest path from $z_k^{(1)}$ to $z_k^{(2)}$. ($P(k)$ will be a shortest path whose terminals are $z_{\phi(k)}$ and $z_{\phi(2m+1-k)}$.)



We shall now present an algorithm for finding $P(1),P(2),\ldots,P(m)$. We assume that a procedure SINGLEPATH(k,Γ) which finds P(k) in the subgraph $G_\Gamma$ of G has already been defined. A single invocation of the algorithm SEVERALPATHS(i,j,Γ) finds $P(i),P(i+1),\ldots,P(j)$:

```
Algorithm SEVERALPATHS(i,j,Γ)
1.  k    := ⌊(i+j)/2⌋;
2.  P(k) := SINGLEPATH(k,Γ);
3.  if i< k then SEVERALPATHS (i,k-1,P(k)⊕Γ);
4.  if k < j then SEVERALPATHS(k+1,j,Γ⊕P(k));
```

All m shortest paths $P(1), P(2),\ldots, P(m)$ are found as a result of a single invocation of SEVERALPATHS$(1,m,\Gamma_0)$. In this invocation, line 2 finds a shortest path P(h) in $G_{\Gamma_0}=G$, where $h = \lfloor(1+m)/2\rfloor$. This lets the algorithm apply the "divide and conquer" approach to finding the remaining paths. Indeed, line 3 finds $P(1),P(2),\ldots,P(h-1)$ in $G_{P(h)\oplus\Gamma_0}$ and line 4 finds $P(h+1),P(h+2),\ldots, P(m)$ in $G_{\Gamma_0\oplus P(h)}$. The Corollary assures that the algo- will terminate with correct results.

The procedure SINGLEPATH can be easily implemented by using say, Dijkstra's algorithm in G, and abandoning the search along any path which crosses Γ. If the data structure defining the graph is implemented suitably by associating with each vertex a list of its neighbors in a counter-clockwise order, such "crossings" can be easily detected.

Let us now consider the time required by SEVERALPATHS (i,j,Γ) (Γ will be $P(j+1)\oplus\Gamma_0\oplus P(i-1)$ where P(0) and P(m+1) are null. For a connected subgraph H of G we shall denote by T(H) the number of operations required to find any single shortest path in H using SINGLEPATH. If $|H|$ is the number of arcs in H then we can write $T(H)\approx t(|H|)$ for an appropriate function t. (For Dijkstra's algorithm $t(x) = O(x\cdot\log_2 x)$ as in a connected planar graph the number of vertices is bounded from below and from above by linear functions of $|H|$ ). We know that $t(0) = 0$ and $t(x)$ grows at least linearly with x and thus we shall assume that if $0 \le x_1,x_2$ then $t(x_1) + t(x_2) \le t(x_1+x_2)$. Furthermore, if t grows faster than linearly, then $t(x_1) + t(x_2) < t(x_1+x_2)$, for $0 < x_1,x_2$.

For ease of notation, assume that $m=2^\mu - 1$ for some $\mu$. One can show that the number of operations, OPS, is bounded by the formula:

$$\text{OPS} \le \sum_{s=1}^{\mu} t\,(|G| + \sum_{j=1}^{2^{s-1}-1} |\,P(j\cdot 2^{\mu-s+1}) \le$$

$$t\,(\log_2 (m+1)\cdot |G| + \sum_{s=1}^{\log_2(m+1)} \sum_{j=1}^{2^{s-1}-1} |\,P(2j(m+1)/2^s)\,|).$$

If $|P(2j(m+1)/2^s|$'s are not known (or at least no upper bound is known) we conjecture that

$$OPS \leq \sum_{s=1}^{\mu} 2\cdot t(|G|) = 2\log_2 (m+1)\cdot t(|G|)$$

## CIRCULAR STRING-TO-STRING CORRECTION PROBLEM

In [WF] a fast algorithm to solve the String-to-String Correction Problem has been presented. Following a suggestion by M. Fischer we apply our algorithm to solve the correction problem for "circular" strings. We assume in the sequel that the reader is familiar with [WF]. Briefly, the paper shows how to utilize dynamic programming to find a minimum cost edit sequence taking a string $A=A_1A_2\ldots A_I$ to a string $B=B_1B_2\ldots B_J$. This problem can also be solved, with the same complexity, by finding a shortest path in an appropriate planar graph.

Indeed consider the graph $G=\langle V,E,L\rangle$ where

$V=\{v_{ij} \mid 0\leq i\leq I,\ 0\leq j<J\}$,

$\langle v_{ij},v_{k\ell}\rangle \in E \Longleftrightarrow (i,j)\neq(k,\ell),\ 0\leq k-1\leq i\leq k\leq I,\ 0\leq \ell-1\leq j\leq \ell\leq J$.

Then every possible edit sequence taking A to B corresponds to a path in G from $v_{00}$ to $v_{IJ}$. For example, the diagram:

String A: x y z w t w x z x

String B: y w x z x y x w

copied from [WF] corresponds to the path drawn in Fig. 4.

We now associate the following length function with the arcs of G, ($\gamma$ is defined in [WF]).

$$L(\langle v_{ij},v_{k\ell}\rangle)=\begin{cases}\gamma(A_k\to B_\ell) & ;\quad k=i+1,\ \ell=j+1\\ \gamma(A_k\to\Lambda) & ;\quad k=i+1,\ \ell=j\\ \gamma(\Lambda\to B_\ell) & ;\quad k=i\ ,\ \ell=j+1\end{cases}$$

Then a minimum cost edit sequence corresponds to a shortest path.

Extend now the problem to consider circular strings. Namely, assume that one is permitted at no cost to circularly shift each string by an arbitrary number of positions before commencing the edit sequence. Formally we wish to minimize the edit cost for taking

$A_iA_{i+1}\ldots A_IA_1\ldots A_{i-1}$ to $B_jB_{j+1}\ldots B_JB_1\ldots B_{j-1}$

over all possible i and j. This problem is transformed into finding min {I,J} shortest paths in an appropriate planar graph, similar to the one used in [FKU] to solve the problem of optimal surface reconstruction.



Define $G^1=(V^1,E^1,L^1)$ where

$V^1=\{v_{ij} \mid 0\leq i\leq 2I-1, 0\leq j\leq J\}$,

$\langle v_{ij},v_{k\ell}\rangle \in E^1 \Longleftrightarrow (i,j)\neq(k,\ell),\ 0\leq k-1\leq i\leq k\leq 2I-1, 0\leq \ell-1\leq j\leq \ell\leq J$, and

$L^1\langle v_{ij},v_{k\ell}\rangle) = L(\langle v_{i(\text{mod } I),j},v_{k(\text{mod} I),\ell}\rangle)$. (See Fig. 5.)

Let $P_i$ for $i=0,1,\ldots,I-1$ denote a shortest path in $G^1$ from $v_i$ to $v_{i+I,J}$. Then a shortest path in $\{P_0,P_1,\ldots,P_{I-1}\}$ corresponds to a minimum cost edit sequence for the circular strings A and B. Using the algorithm SEVERALPATHS, it can be computed in the number of operations less than $\log_2 (I+1) + 2$ times that required to solve the standard String-to-String Correction Problem. (We assumed that $I\leq J$. If $J<I$ a simple modification is required.)

## REFERENCES


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

[FKU] Fuchs, H., Z.M. Kedem, and S.P. Uselton, Optimal Surface Reconstruction from Planar Contours, CACM 10 (1977), 693-701.

[WF] Wagner, R.A. and M.J. Fischer, The String-to-String Correction Problem, JACM 21 (1974), 168-173.

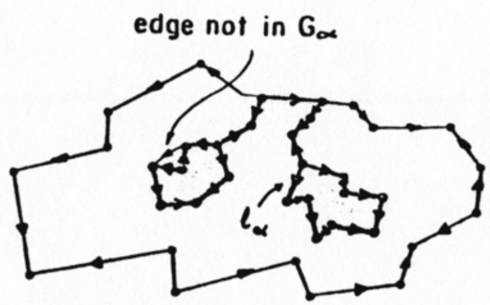


Figure 1

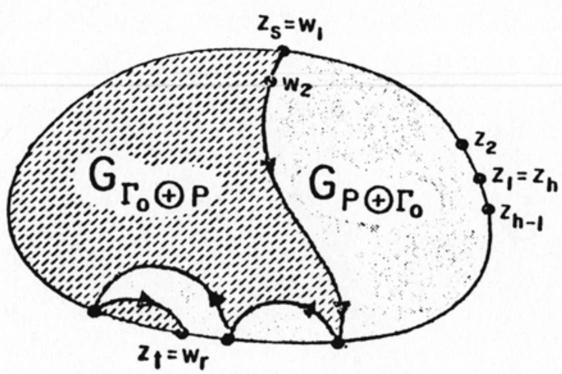


Figure 2



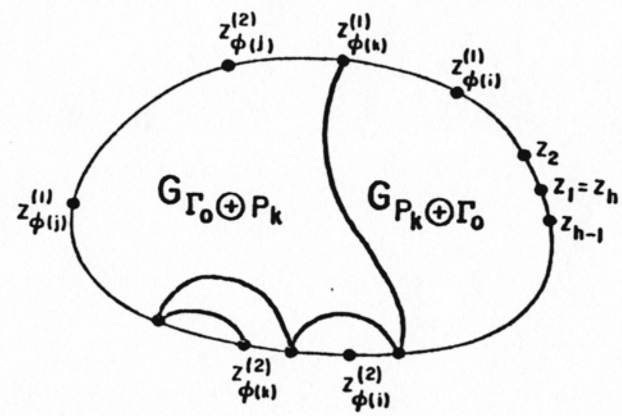


Figure 3

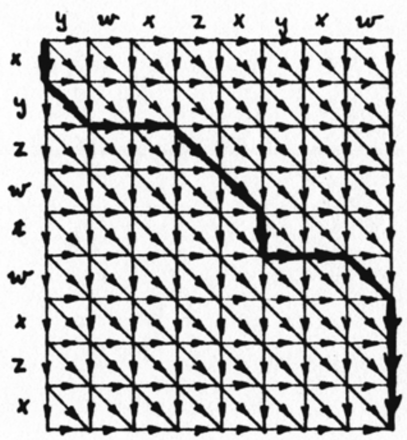


Figure 4

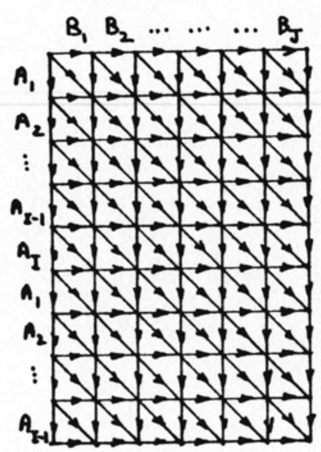


Figure 5