# The Shortest and the K-Shortest Routes as Assignment Problems

A. Weintraub
University of California
Berkeley, California

ABSTRACT

*The problem of finding a shortest route in a network with unrestricted costs is approached through solving an assignment problem associated to the network.*

*The upper bound on the number of elementary calculations required for the solution is $O(m^3)$. However, in most cases, the actual number of computations is considerably less and depends on different network characteristics than Dynamic Programming algorithms do. In examples of networks generated stochastically, this number was below $O(m^{2.5})$.*

*A parametric analysis is presented. It is shown that if after a shortest route is determined, the costs on all arcs incident into or out of a node are modified in any form, at most $O(m^2)$ elementary calculations will determine a new optimal solution. This feature, shared by Dynamic Programming algorithms only for cases where all cost decrease, can be applied to problems such as the determination of the K-shortest routes and the K-smallest assignments, leading to upper bounds of $O(Km^3)$ in both cases.*

## INTRODUCTION

The problem of finding the shortest route in a network with directed arcs and unrestricted costs on them has been usually approached through dynamic programming techniques [1,5,12]. A less used approach is studied in this paper, where a shortest route is determined by solving an assignment problem associated with the network.

The algorithm for finding the shortest route between nodes 1 and m in a network is based on the following notion: add an artificial arc from node m to node 1 with a very high negative cost. Obtain then, in the modified network, circuits with minimum added cost. The optimal solution to this problem can be determined by solving an assignment problem and will either determine a shortest route, or prove the existence of negative circuits.

To obtain such optimal solution, a modified version of an algorithm presented in references [2,9] for finding minimum cost flows in networks with linear, nonnegative costs is used. Through this method, the shortest route problem is solved in a number of elementary calculations with the usual upper bound of $O(m^3)$. However, in most cases, the actual number of calculations will be considerably lower. Moreover, this number will depend on network characteristics distinct from those which determine the number of iterations in the Dynamic Programming approach.

A parametric analysis of this algorithm is presented. It is shown that if the cost in any arc (or all arcs incident to or leaving a node) is varied in any direction, after a shortest route has been determined, at most $O(m^2)$ additional elementary calculations will be needed to determine a new optimal solution. This feature is shared by Dynamic Programming algorithms only for cases where all costs decrease.

This property can be applied, among others, to the determination of the K-loopless shortest routes and K-smallest assignments. In both cases, the upper bound on the number of elementary calculations is $O(K \cdot m^3)$.

## 1. THE SHORTEST ROUTE AND THE ASSIGNMENT PROBLEMS

### a) The Assignment Problem:

Consider a matrix $D = (d_{ij})$. We define as an *assignment* in D any set of m independent cells, where a set of independent cells is such that one cell is marked in each row and in each column.

An assignment is said to be *optimal* if the summation of costs $d_{ij}$ of its marked cells is minimum among all assignments. Finding an optimal assignment can also be expressed as the following linear programming problem:

$$\sum_{i=1}^{m} x_{ij} = 1$$

$$x_{ij} \geq 0$$

$$\sum_{j=1}^{m} x_{ij} = 1$$

$$\text{Min} \sum_{i,j} d_{ij}\, x_{ij}$$

By unimodularity of the constraint matrix, one optimal solution will correspond to integer, 0-1 values of the variables $x_{ij}$.

References [2,9] offer an algorithm to find minimum cost flows in networks with linear, nonnegative costs. In this algorithm, each iteration consists in driving as much flow as is feasible between an origin and a destination node, along a path of minimum incremental cost. Node weights are used to modify arc costs in such a way that all shortest routes are determined over networks with nonnegative costs.

This algorithm can be specialized to the assignment problem as follows:

Define $U_i$ and $V_j$ as the cost of rows i and columns j.

1) Set $U_i = 0$ for all i, and $V_j = \text{Min}_i\ (d_{ij})$.
2) Set $d_{ij} = d_{ij} + U_i - V_j$.
3) If $d_{ij} = 0$, cell (ij) is admissible. Find N, the maximum number of independent admissible cells and mark them [4,p.55]. If N = m, we have an optimal solution. Otherwise, go to 4.
4) Set $U_i = 0$ for all rows such that no cell in that row is marked and label those rows. Set $U_i = M$ (very large) for all other rows. Set $V_j = M$ for all j.
5) If no unlabelled rows or columns exist, go to 2. Otherwise, find the minimum value among all labelled $U_i$ and $V_j$. Label the corresponding row or column. If it is a row, go to 6. It it is a column, go to 7.
6) Set $V_j = \text{Min}\ (V_j,\ U_i + d_{ij})$. If column j contains a marked cell, go to 6', j = 1,m.

6') Let (kj) be the marked cell in column j. If there exists a column r such that:
   i) Cell (kr) is admissible,
   ii) Column r has no marked cell and
   iii) $V_r > V_j$

   then, erase the mark from cell (kj) and place it on cell (kr). If such a cell does not exist, do nothing. Go to 5.

7) Let $Z_j$ be the set of marked cells in column j, then $Z_j$ is either empty or it contains one cell, say $d_{kj}$. In the first case, no value changes. If $d_{kj} \in Z_j$, set $U_k = \text{Min}\ (U_k, V_j)$. Go to 5.

Step 6' is a slight modification to the original algorithm, which reduces the number of iterations required to reach optimality [10].

If we consider Steps 2 through 7 as an iteration, it is clear that the number of elementary calculations per iteration is of order $m^2$. Since at each iteration the number of independent cells is increased by at least one, the number of iterations is at most m. Thus, the total number of elementary calculations has an upper bound of order $m^3$.

We prove now an important property of the above algorithm.

*Theorem 1: Consider a matrix $D = (d_{ij})$ and an optimal assignment S in D. Let $\underline{D} = (\underline{d}_{ij})$ be the cost matrix associated with the final, optimal solution S.*

*If for any row r (or column s) all values $d_{rj}$, $j = 1,..,m$ ($d_{is}$, $i = 1,..,m$) are modified, at most $O(m^2)$ additional elementary calculations will be required to determine an optimal solution.*

*Proof:* Suppose all values $d_{rj}$ of row r are modified to $d'_{rj} = d_{rj} + K_{rj}$, $-\infty < K_{rj} < \infty$. Since all transformations throughout the algorithm are additive, the modified values in the final matrix will be:

$$\underline{D}' = \begin{cases} \underline{d}'_{ij} = \underline{d}_{ij} & i \neq r \quad j = 1,..,m \\ \underline{d}'_{rj} = \underline{d}_{rj} + K_{rj} & j = 1,..,m \end{cases}$$

Obviously, S may no longer be optimal solution for the modified cost matrix D'.

Let (rp) be the marked cell corresponding to row r in S.

Let $\underline{d}'_{rv} = \min_{j=1,m} \{\underline{d}'_{rj}\}$

If $\underline{d}'_{rv} < 0$, take $\underline{d}'_{rj} = \underline{d}'_{rj} + |\underline{d}'_{rv}|$ $\quad j = 1,..,m$

If $\underline{d}'_{rv} > 0$, take $\underline{d}'_{rj} = \underline{d}'_{rj} - |\underline{d}'_{rv}|$ $\quad j = 1,..,m$

Note that an additive transformation in all cells of a row or column (arcs incident into or out of a node) does not alter the solution of the optimal assignment.

Consider the new matrix $\underline{D}'$: We have (m-1) independent admissible cells in rows $i \neq \overline{r}$. In row r, all values $\underline{d}_{rj} \geq 0$, j = 1,..,m (dual feasibility), while one of these values at least, has a value 0. If cell (rp) corresponds to one of these values, S is also optimal for the modified cost matrix D'. Otherwise, since the algorithm determines at least one added independent cell per iteration, at most $O(m^2)$ elementary calculations will be required to determine a new optimal solution.

The same argument follows for the case when costs are modified in a column.

b) Relation to the Shortest Route Problem:

Consider now a network E of m nodes and n directed arcs. Arcs will be referred to as elements $e_k$ of E, or as elements (ij), indicating their corresponding nodes. Let $d_{ij}$ be the cost of arc (ij), where the values $d_{ij}$ are considered bounded and unconstrained in sign. We can then construct a cost matrix $D = (d_{ij})$, where $d_{ii} = 0$ for i = 1,..,m and

$d_{kr} = M'$ (very large) for nonexistent arcs (kr).

We define a *circuit* in E as a set of nodes and arcs i,(ij),j,..,(ki),i, such that no intermediate node or arc is repeated. A circuit will be called *negative* if the summation of costs of arcs in the circuit is negative.

A set of disjoint circuits or *circuit-set* is defined by the fact that for each node there exists one and only one incident and outgoing arc. Nodes not belonging to a physical circuit are related to a self-loop.

A *route* from node 1 to node m in E will be defined as a sequence of nodes and arcs: 1,(1i),i,..,r,(rm),m such that no node is repeated.

A *shortest route* S from 1 to m will be such that $\sum_{(ij)\varepsilon S} d_{ij}$ is minimum among all routes from node 1 to m.

*Lemma 1: There exists a one to one correspondence between assignments in D and circuit-sets in E.*

By Lemma 1, determining an optimal assignment in D is equivalent to finding an optimal circuit-set in E.

We can solve then the shortest route problem as an assignment one:

We first inspect for the existence of negative circuits in E, by determining an optimal assignment in D. An all-diagonal optimal solution will indicate that no negative circuits exist. (This solution corresponds to a set of self-loops in E).

If this is the case, we add an artificial arc from node m to node 1, with a very high negative cost ($d_{m1} = - M$, $|M| < < |M'|$,) and solve the modified assignment problem.

*Lemma 2: If no negative circuits exist in E, an optimal circuit-set in the modified network, with $d_{m1} = - M$, will consist of self-loops and, if a directed path exists from node 1 to m, one circuit S, which includes the arc (m1) and a shortest route from 1 to m. If no directed path exists, the circuit-set will consist of one self-loop for each node.*

By Theorem 1, only $O(m^2)$ additional elementary calculations will be required to determine the new optimal assignment, which either corresponds to a shortest route or shows that none exists.

The algorithm and the corresponding proofs are detailed in [11].

## c) Comparison of the Assignment Algorithm to the Dynamic Programming Approach:

The literature offers different algorithms to determine a shortest route in a network with unrestricted costs, all of which have an upper bound of $O(m^3)$ elementary calculations.

The effective number of calculations, however, is usually lower, depending for each algorithm, on the structure of the network.

It is of interest then to establish the determining factors for the actual number of iterations in each algorithm, in order to evaluate which one is preferable for a given problem.

In the usual Dynamic Programming approach [1], the number of iterations required will depend on the length of the shortest route, or the distribution of negative circuits, should they exist.

In contrast, the number of required iterations for the presented algorithm will depend essentially on the correlation in the values $d_{ij}$ among the different columns of D: In case of total correlation, where, for example, the costs $d_{ij}$ are increasing with the number of the row in each column, the algorithm will obtain only one added independent cell in each iteration. In case of no correlation among the columns, the number of iterations decreases considerably [10].

Computational experience showed that in this case the required number of iterations was consistently of order $(m)^{1/2}$, regardless of other variations in the network structure.

We present two illustrative cases.

Networks of 15 to 80 nodes were built randomly from a (0,1) uniform and a (0,1) normal distributions.

Two parameters were considered:

i) d, the density of arcs in the network.
ii) p, the % of negative cost arcs.

The effect of varying both parameters did not significantly affect the number of iterations required to reach optimality. (This number increased with d, and decreased with p).

Results corresponding to 10 example runs of each case are presented in Table 1.

| No. of Nodes | Average Number of Iteration, 10 Ex. Uniform Dist. d=50%, p=10% | Normal (0,1) Dist. d=100%, p=50% |
|---|---|---|
| 15 | 3 | 3.3 |
| 30 | 4.3 | 4.7 |
| 50 | 5.2 | 4.9 |
| 80 | 6.8 | 5.8 |

Table 1

As can be seen from Table 1, the number of iterations required was, in all cases, below $(m)^{½}$.

The following two examples show how, for given networks, the number of iterations will differ according to the approach used.

Network 1: $d_{i,i+1} < 0$ $i=1,m-1$, all other costs nonnegative.

Network 2: $d_{i+1,j} > d_{i,j}$ $i=1,m-1$ $j=1,m$

$d_{1,m} < 0$, all other costs nonnegative.

In Network 1, the length of the shortest route is $(m-1)$, whereas only one iteration will be required if the Assignment Algorithm is used. These values are reversed in Network 2.

One additional advantage of the presented algorithm is that it can determine, with no modifications, a shortest route in networks with nonnegative costs in $O(m^2)$ elementary calculations [11].

## 2. PARAMETRIC ANALYSIS

Let S be an optimal solution corresponding to a cost matrix $D = (d_{ij})$.

If values $d_{ij}$ corresponding to either a row or a column are modified in any direction, a new optimal solution S' can be determined in at most $O(m^2)$ elementary calculations. (Theorem 1).

This feature is not shared by Dynamic Programming algorithms, which attain a similar efficiency only for the case when the costs in arcs are decreased, [7]. This advantage of the proposed algorithm can be used either directly or in implementing some algorithms.

As direct applications we have:

i) The obvious case where, after an optimal solution S to the shortest route (assignment) problem has been determined, one (or a small number) of values $d_{ij}$ are modified.

ii) Case where, after a shortest route S in a network E (matrix D) has been determined a node is to be added or deleted.

If a node is to be deleted, we simply eliminate the row and column in D corresponding to that node. There remain at least (m-2) independent marked cells in a (m-1) × (m-1) matrix. Hence, at most $O(m^2)$ additional elementary calculations will be required to determine the new optimal solution.

If a node (m+1) is to be added, with costs $(d_{i,m+1}, d_{m+1,j})$, $i,j = 1,..,m$, a small addition to the algorithm is needed: A vector of values $U' = (u_i')$, $i = 1,..,m$, is kept, where $u_i' = \sum_k u_i^k$, $u_i^k$ being the cost of row i in iteration k. After obtaining the initial optimal solution for the m node network, we add a (m+1) row and column corresponding to the cost values of the (m+1) node. By assigning values $d'_{i,m+1} = d_{i,m+1} + u'_i$, the (m+1) column is updated to all additive transformations performed through the iterations. The (m+1) row is updated similarly. After performing, if needed, row or column additions, to preserve nonnegativity, at most $O(m^2)$ elementary calculations will determine the new optimal solution.

As examples of implementation of Algorithms we have:

i) Minimum Flow Cost in Networks with Linear Costs.

With no loss of generality we can consider the total flow F going from node 0 to a node Z. This problem can be solved through an algorithm which, for each feasible flow f, finds the path of minimum additional cost between 0 and Z, and assigns as much flow as is possible through that path. If costs are unrestricted in sign, a basic assumption of the algorithm proposed in reference [2,9] is not satisfied and shortest routes will have to be determined over networks with unrestricted costs.

If the number $K^t$ of arcs which simultaneously block a shortest route as the flow increases through it is small, which is usually the case, the presented algorithm can be advantageous. As arcs become blocked, their cost goes to infinity for the determination of the next shortest route, while all other costs remain equal. By Theorem 1, at most $K^t \cdot O(m^2)$ elementary calculations will determine the new shortest route.

ii) The K-Smallest Assignments.

This problem can be solved by the following general scheme [6]:

0) Determine an optimal solution $X^1 = (x_1^1,..,x_m^1)$, without fixing any values of the variables. Place this solution in a tentative list of solutions L.

1) (k-1) smallest assignments have been determined. Remove the least costly solution in L. This corresponds to the k th smallest assignment and is denoted as $X^k$. $X^k$ is characterized by those variables at value 1, and those which are fixed, some of them at value zero.

2) If $k = K$, terminate. Otherwise, go to 3.

3) Determine additional solutions for L.

Suppose $X^k$ was determined by fixing the values of the variables $x_1, \ldots, x_s$ at level one, others at level 0. We leave these variables fixed and create (m-s) new assignment problems, by fixing the remaining variables in $X^k$ as follows:

(1) $x_{s+1} = 1 - x^k_{s+1} = 0$

$\vdots$

(m-s) $x_{s+1} = x^k_{s+1}, \ldots, x_{m-1} = x^k_{m-1}, x_m = 1 - x^k_m$

The optimal solutions of these problems are determined and placed in L. Go to 1.

It is clear that Step 3 involves the determining number of iterations. Let $\underline{D}^k$ be the cost matrix corresponding to the optimal assignment $X^k$.

By using the algorithm presented above, each optimization i, i = 1,..,(m-s) involves the following steps:

Consider $D^k(i)$, the cost matrix corresponding to optimization i. $(D^k(1) = \underline{D}^k)$. In order to perform optimization (i+1), only the following alterations in $D^k(i)$ are required to obtain $D^k(i+1)$:

1) $x^k_{s+i-1}$ goes from 0 to 1.

2) $x^k_{s+i}$ is fixed at 0.

1) is satisfied by fixing $x_{s+i-1}$ to 1 and eliminating the row and column corresponding to that variable in $D^k(i+1)$. This involves no additional calculations.

2) requires modifying the cost corresponding to variable $x^k_{s+i}$ to infinity for $D^k(i+1)$.

Then, from Theorem 1, at most $0(m^2)$ elementary calculations will be required to solve each additional optimization. If the matrices $\underline{D}^k$ are not preserved (storage requirements), they will have to be recreated when needed. This will require an additional number of iterations of order $K\ 0(m^2)$. Hence, the total number of calculations required to determine the K-smallest assignment will be $2\ 0(Km^3)$. This value improves the bound of $0(Km^4)$ obtained when a different algorithm is used to solve each assignment optimization [8].

iii) The K-Shortest Loopless Paths in a Network.

This problem is approached in the same form as the above. Step 3 in this case is considered in the following way:

Let $X^k$ be the k shortest route, consisting of arcs $x_1^k,..,x_p^k$ and nodes $y_o = 0, y_1,..,y_p = Z$.

For each node $y_i$ in the $k^{th}$ shortest route, consider all subpaths of shortest routes 1 through k, which coincide with the $k^{th}$ up to node $y_i$. This determines a set C of shortest routes.

We define the set $(a_i)$ as composed of the arcs leaving node $y_i$ in the set of routes C. Obviously, arc $x_{i+1}^k \varepsilon (a_i)$. Then, each optimization i, $i = 0,..,p-1$, consists of finding the shortest route from node $y_i$ to Z, without going through nodes $y_o,..,y_{i-1}$ (for $i = 0$, no node is excluded), and excluding arcs in the set $(a_i)$.

We note that from one optimization problem to the next, the only modification involving calculations consists of driving the cost of some arcs leaving the node $y_i$ to infinity. Then, by Theorem 1, the above algorithm will require at most $0(m^2)$ elementary calculations to determine each succeeding shortest route. Since at most m such optimization are needed for each shortest route, the K-loopless shortest routes can be determined in at most $0(Km^3)$ elementary calculations, as obtained also in [6] through a Dynamic Programming approach. Both these results reduce the value $0(Km^4)$ given in [13].

## REFERENCES

1. Dreyfus, S., "An Appraisal of Some Shortest Path Algorithms," *Operations Research*, Vol. 17, No. 3, pp. 395-412, May 1969.

2. Edmonds, J. and R. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *Journal A.C.M.*, Vol. 19, No. 2, pp. 248-264.

3. Florian, M. and P. Robert, "A Direct Method to Locate Negative Cycles in a Graph," *Management Science*, Vol. 17, No. 5, pp. 307-310.

4. Ford, L. R. and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, 1962.

5. Hu, T. C., *Integer Programming and Network Flows*, Addison-Wesley, 1969.

6. Lawler, E. L., "A Procedure for Computing the K Best Solutions to Discrete Optimization Problems and its Application to the Shortest Path Problem," *Management Science*, Vol. 18, No. 7, pp. 401-405.

7. Murchland, J. D., "The Effect of Increasing or Decreasing the Length of a Single Arc on all Shortest Routes," Transport Network Theory Unit, London School of Economics, *Report LBS-TNT-26*, September 1967.

8. Murty, K. G., "Algorithms for Ranking all the Assignments in Order of Increasing Cost," *Operations Research*, Vol. 16, 1968, pp. 682-687.

9. Tomizawa, N., "On Some Techniques Useful for Solution of Transportation Network Problems," *Networks*, Vol. 1, No. 2, pp. 173-194.

10. Weintraub, A., "A Heuristic Analysis of Karp-Edmonds Algorithm for the Assignment Problem," unpublished report.

11. Weintraub, A., "The Assignment and the Shortest Route Problems," Operations Research Center, Berkeley, *Report 72-19*, presented at the TIMS-ORSA-AIIE Joint Meeting, Atlantic City, November 8-10, 1972.

12. Yen, J. Y., "A Shortest Path Algorithm," Ph.D. Dissertation, University of California at Berkeley, 1970.

13. Yen, J. Y., "Finding the K Shortest Loopless Paths in a Network," *Management Science*, Vol. 17, No. 11, pp. 712-716, July 1971.

*Paper received August 21, 1972.*