AN ALGORITHM FOR THE SOLUTION OF SHORTEST PATH PROBLEMS WITH POSITIVE AND NEGATIVE ARC LENGTHS

B. Dorhout [1]
M.M.G. Hunting [2]

Abstract:

One of the most successful methods for the one-to-all shortest path problem in a directed graph is Dijkstra's method. If m is the number of arcs and n is the number of nodes of the graph there are implementations with worst case complexity $O(m + n \log n)$. Unfortunately this method can only be applied on graphs with nonnegative arc lengths, or costs, while sometimes, for instance if traversing some arcs yields income and therefore has negative costs, an algorithm for one-to-all shortest path problems with arbitrary arc lengths is needed. In this paper a method is described for graphs with arbitrary arc lengths. It can be considered as an extension of Dijkstra's method, using this method in two ways: for the transformation of the problem into a problem with only nonnegative arc lengths and for the solution of this new problem. The transformation is made through the solution of a linear assignment problem. If r is the number of nodes which are tails of one or more arcs with negative length, this assignment problem can be solved by applying Dijkstra's method at most r times on small subproblems. So the worst case complexity of the problem is $O((r + 1)(m + n \log n))$ if the algorithm described in this paper is applied. The numerical results suggest that at the moment other existing methods solve these problems in shorter computation times.

---

[1])Universiteit Twente
faculteit Toegepaste Wiskunde
postbus 217
7500 AE Enschede
tel. 053-4893447

[2])idem, tel. 053-4893385

1. Introduction

We consider the problem of determining the shortest paths from one arbitrary node of a digraph, 1 say, to all other nodes. It is assumed that all arcs have a given length and that these paths exist.
In case the graph does not contain a negative cycle, i.e. a cycle for which the sum of the arc lengths is negative, the solution of the problem coincides with the optimal solution to a transshipment problem, which asks to minimize the sum of the lengths of all paths used to transport 1 unit of a good from node 1 to each other node. We solve the original problem by first verifying that G does not contain negative cycles and subsequently solving the transshipment problem by reformulating that as a transportation problem in the classical way. (See, e.g. Wagner (1975)). The basic variables in the optimal solution of the transshipment problem represent a spanning tree of shortest paths, rooted at node 1, which is an optimal solution to the original problem.
If the graph contains a negative cycle the optimal solution to the transportation problem differs from a solution to the original problem. In this case the algorithm stops, and no solution is found.

We denote the graph by $G = (N,A)$, where $N = (1,\ldots,n)$, and A is the set of pairs (i,j), which define the arcs (i,j) of G, with given lengths c_{ij}. If x_{ij} is the number of shortest paths that contain arc (i,j), the transshipment problem (P) associated to the one to all shortest path problem is to minimize

$$\sum_{i=1}^{n} \sum_{j=2}^{n} c_{ij}\, x_{ij} \tag{1}$$

subject to

$$\sum_{k=2}^{n} x_{1k} = n - 1, \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} - \sum_{k=2}^{n} x_{jk} = 1, \qquad j = 2,\ldots,n, \tag{3}$$

$$x_{ij} \geq 0, \qquad (i,j) \in A. \tag{4}$$

Here, for ease of notation, summations over indices only concern pairs that belong to A. Without loss of generality one may assume that there are no arcs (i,n), $i = 1,\ldots,n$.

We describe how to solve (P) by applying a moderate number of times Dijkstra's method (Dijkstra, 1959), which is suited only for the solution of shortest path problems in a graph with nonnegative arc lengths. In section 2 we transform (P) into a transportation problem (P'). It is shown that by solving an assignment problem the objective function of (P') can be changed into an objective function with only nonnegative cost coefficients. This gives the optimal solution of (P') if G does not contain negative cycles. In section 3 we describe phase 1 of our method, in which the assignment problem is solved and the existence of negative cycles is detected. In section 4 we see that in phase 2, executed only in absence of negative cycles, (P) can be solved by using Dijkstra's method once. In section 5 we describe some experimental results. These indicate that, in spite of the good complexity of our method, other algorithms perform better at this time.

2. Transformation of the problem

Problem (P) is transformed into a transportation problem (P') in the following way. As an optimal solution to (P) is associated with a spanning tree, the number of shortest paths arriving at node j, $j = 2,\ldots,n$, can never be greater than $n - 1$.

So $\sum_{i=1}^{n} x_{ij} < n$, $j = 2,\ldots,n$, holds, and redundant constraints $\sum_{i=1}^{n} x_{ij} \leq n$, $j = 2,\ldots,n$ may be added to the formulation of (P). Besides, an optimal solution to (P') is an optimal solution to the original problem only if the 'smaller than' - signs hold. Then, after introduction of nonnegative slack variables x_{jj}, defined as $x_{jj} = n - \sum_{i=1}^{n} x_{ij}$, with $c_{jj} = 0$, $j = 2,\ldots,n$, and subsequent substitution in (3), (P) is converted into a transportation problem in the bipartite graph $G' = (I,J,B)$, with $I = \{1, \ldots ,n\}$, $J = \{2, \ldots ,n\}$, and $B = A \cup \bigcup_{j=2}^{n} \{(j,j)\}$. Thus (P') is:

minimize

$$\sum_{(i,j)\in B} c_{ij}\, x_{ij} \tag{1'}$$

subject to

$$-\sum_{j=2}^{n} x_{ij} = -n + 1, \qquad i = 1,\ldots,n \tag{5}$$

$$\sum_{i=1}^{n} x_{ij} = n, \qquad j = 2,\ldots,n \tag{6}$$

$$x_{ij} \geq 0, \qquad (i,j) \in B, \tag{7}$$

where now summations in (5) and (6) are taken over indices associated with elements of B. These represent arcs (i,j) with $i \in I$ and $j \in J$.

This conversion is made in order to apply the well known reduction property, which says that the optimal solution of a transportation problem does not change if in the objective function all c_{ij} are replaced by $c_{ij} - p_i - q_j$, where p_i and q_j are constants which only depend on i and j respectively.

Problem (P') is nondegenerate, as it is impossible to express 0 as the sum of less than all right hand sides of (5) and (6). Each optimal solution to (P) coincides with an optimal solution to (P') from which the x_{jj}-variables, all with positive values, are removed. Only if a negative cycle exists, there is a difference between the outcomes of (P) and (P'). In that case (P) does not have an optimal solution, as (2), (3), (4) allow feasible solutions with arbitrarily low values of (1), whereas (P') clearly only has solutions with bounded values.

Problem (P) is solved in the following way: In order to determine whether G contains negative cycles we solve the linear assignment problem, (P"), in the graph G" = (I',J,B'):

minimize

$$\sum_{(i,j)\in B'} c_{ij}\, x_{ij} \tag{8}$$

subject to

$$-\sum_{j=2}^{n} x_{ij} = -1, \quad i = 2,\ldots,n, \tag{9}$$

$$\sum_{i=2}^{n} x_{ij} = 1, \quad j = 2,\ldots,n, \tag{10}$$

$$x_{ij} \geq 0, \quad (i,j) \in B', \tag{11}$$

where $I' = \{i \mid i = 2, \ldots, n\}$ and $B' = \{(i,j) \mid i \in I';\ j \in J,\ (i,j) \in B\}$.

As G" is the subgraph of G', induced by (I',J), it contains all cycles of G', and these are uniquely associated with all cycles in G. So if the optimal value of the objective function of (P") is negative, thenG' and so G contains at least one negative cycle, and the algorithm is stopped. Otherwise we use the optimal solution of the dual problem (D") of (P"), which is obtained as a by-product of the algorithm for solving (P"). (D") is the problem to

maximize

$$-\sum_{i=2}^{n} u_i + \sum_{j=2}^{n} v_j \tag{12}$$

subject to

$$-u_i + v_j \leq c_{ij}, \quad (i,j) \in B'. \tag{13}$$

We denote the optimal solutions of (P") and (D") by $\bar{x}$ and $(\bar{u},\bar{v})$ respectively. As G does not contain a negative cycle the optimal solution to (P") is $\bar{x}_{jj} = 1$, $j = 2, \ldots, n$, and thus, by complementary slackness, $-\bar{u}_j + \bar{v}_j = c_{jj} = 0$, $j = 2, \ldots, n$. So, if we define $\bar{u}_1 = -\min_{j=1\ldots n} (c_{1j} - \bar{v}_j)$, problem (P') may be solved with c_{ij} in (1') replaced with $\bar{c}_{ij} = c_{ij} + \bar{u}_i - \bar{v}_j$. But in the optimal solution to (P') the values of x_{ij} for $i \neq j$ are equal to the values of x_{ij} in the optimal solution to (P). So the optimal solution to (P) is obtained by determining the shortest path spanning tree in G, with arc lengths $\bar{c}_{ij}$ which are nonnegative, by (13) and the definition of $\bar{u}_i$. This is done by Dijkstra's algorithm.

3. Phase 1: Solution of an assignment problem

It is possible to solve assignment problem (P") by the algorithm described in Dorhout (1977). A short description and a FORTRAN subroutine for this algorithm can also be found in Burkard and Derigs (1980). It is an improved version of an algorithm published by Tomizawa (1971). We give a short description of this iterative algorithm for the present special case.

The idea of the algorithm is to solve subsequently for $p = 2, \ldots, n$ the problems (P_p):

$$\sum_{(i,j)\in B'_p} \bar{c}_{ij}\, x_{ij} \tag{14}$$

subject to

$$-\sum_{j=2}^{n} x_{ij} = -1, \quad i = 2,\ldots,p, \tag{15}$$

$$\sum_{i=2}^{n} x_{ij} \leq 1, \quad j = 2,\ldots,n, \tag{16}$$

$$x_{ij} \geq 0, \quad (i,j) \in B'_p, \tag{17}$$

where $I'_p = \{i \mid i = 2, \ldots, p\}$, $B'_p = \{(i,j) \mid i \in I'_p;\ j \in J,\ (i,j) \in B\}$, and $\bar{c}_{ij} \geq 0$ for all $(i,j) \in B'_p$.

We call arcs (i,j) with $x_{ij} = 1$ *assigned*, and its end nodes i and j *matched*. Other nodes are *free*. With each solution of a problem (P_p) the number of *assignments* $x_{ij} = 1$ increases with one. Finally the solution to (P_n) coincides with the optimal solution to (P"). The optimal solution to (P_p) is derived from the optimal solution to (P_{p-1}) by applying Dijkstra's method for finding a shortest alternating path from node $p \in I'_p$ to the nearest free node in J. It consists of alternating forward, not assigned arcs, and backward, assigned arcs. The optimal solution to (P_p) is obtained by assigning all forward arcs, and deleting the assignments of the backward arcs in the shortest path. As arc lengths we use $\bar{c}_{ij} = c_{ij} + \bar{u}_i - \bar{v}_j$, with $\bar{u}_p = -\min_{j=1\ldots n} (c_{pj} - \bar{v}_j)$, and all other $\bar{u}_i$ and $\bar{v}_j$ taken from the optimal solution to (D_{p-1}), the dual problem to (P_{p-1}). So all $\bar{c}_{ij} \geq 0$, and

for each $i \in I'_{p-1}$ there is one assigned arc (i,j), with $\bar{c}_{ij} = 0$, by complementary slackness. (P_1) can be solved by setting $\bar{u}_1 = -\min_{j=1\ldots n} c_{1j}$, $\bar{v}_j = 0$, $j = 1, \ldots, n$, and, if $\bar{u}_1 = -c_{1k}$, $\bar{x}_{1k} = 1$.

The optimal u_i- and v_j- values for (D_p) are easily derived from the associated optimal values for (D_{p-1}) and the distance labels.

It is not necessary to solve all n-1 shortest path problems: If $R = \{i \in N\backslash(1) \mid \exists\ j \in N,\ (i,j) \in A: c_{ij} < 0\}$, the set of nodes which are tails of one or more arcs with negative length, and $r = |R|$, then only r times applying Dijkstra's algorithm is sufficient. For after renumbering nodes 2 to n in such a way that the first n-r nodes in I' are not in R, the n-r first iterations consist of simply assigning (i,i), $i=1, \ldots, n-r$. As the complexity of an efficient implementation of Dijkstra's method by Fredman and Tarjan (1987) is $O(m + n \log n)$, with $m = |A|$, the complexity of phase 1 is $O(r\ (m + n \log n))$.

4. Phase 2: A single execution of Dijkstra's procedure

Consider the optimal solutions to (P") and (D"). If the optimal solution to (D") is $u_i = v_i = \bar{u}_i = \bar{v}_i$, $i = 2, \ldots, n$, then this solution, supplemented with $\bar{u}_1 = -\min_{j=1\ldots n} (c_{1j} - \bar{v}_j)$, is a feasible solution to (D'), the dual problem of (P'). Now the complementary slackness property holds with respect to the optimal solution $x_{ii} = 1$, $i = 2, \ldots, n$, $x_{ij} = 0$, $(i,j) \in A$, to (P"), and consequently also with respect to this solution, multiplied by (n-1). This latter solution satisfies (5) for $i = 2, \ldots, n$. The optimal solution to (P') is found by sending 1 supplementary unit from node 1 to all $n - 1$ nodes of J over the arcs of B. This can be done because all forward arcs of A have infinite capacity and all backward arcs (i,i) have capacity $n - 1$. But this solution gives the same shortest path spanning tree as the optimal solution to (P) with all c_{ij} replaced with $c_{ij} + \bar{u}_i - \bar{v}_j$.

After the application of Dijkstra's method in this phase the total complexity of our problem is $O((r + 1)(m + n \log n))$.

5. Computer experiments

Cherkassky et al. (1996) did an extensive computational study of one-to-all shortest paths algorithms. All of these algorithms have been implemented by them and are tested on several sets of randomly generated graphs. They are available on the Internet (1). One of these algorithms is DIKF, an implementation of Dijkstra's algorithm that uses Fibonacci heaps as proposed by Fredman and Tarjan (1987). In the implementation of our method we used DIKF for solving phase 1 and 2. We denote this implementation of our method by DIJK.

The results of the study by Cherkassky et al. showed that the algorithm called GOR1 was the fastest one in their test set. GOR1 is a modification of a topological ordering algorithm designed by Goldberg and Radzik (1993). We used this algorithm for comparing the performance of our algorithm. For that purpose we also used a variant of the Bellman-Ford-Moore algorithm, due to Bellman (1958), Ford (1962) and Moore (1959), and denoted as BFP by Cherkassky et al. Both GOR1 and BFP have a time bound of $O(nm)$, which is almost always worse than $O((r + 1)(m + n \log n))$.

The first tests with our method were very disappointing, since GOR1 and BFP were much faster in solving the shortest path problems we generated. We decided to use a different approach for solving the assignment problem in phase 1, and for that purpose we used the cost scaling algorithm CSA designed by Goldberg and Kennedy (1995). This algorithm has the nice feature that it provides also an optimal dual solution to the assignment problem, which we need in phase 2. (Actually, CSA returns non-integers u_i and v_j such that $c_{ij} + u_i - v_j \leq \epsilon$, where $\epsilon = 1/(2n+1)$. Using an $O(m)$ algorithm of Dial (1969) and Wagner (1976) one obtains an integer dual solution, as explained by Goldberg and Tarjan (1990). In all our experiments, however, rounding down the values of u_i and v_j was sufficient.) We used Goldberg and Kennedy 's implementation which is available on the Internet (1), and by SCAL we denote the method that solves shortest path problems by using CSA in phase 1 and DIKF in phase 2.

DIJK, GOR1 and BFP are strongly polynomial time bounded algorithms, SCAL is not. If the interval from which the arc lengths are selected grows, SCAL will take more time to solve the problem.

Three problem types were used for comparing DIJK, SCAL, GOR1 and BFP. The first type of problems, Rand-Mix, consists of graphs with both positive and negative arc lengths. They were generated by first assigning a shortest path from the source to each node, and then randomly assigning the other arcs without generating negative cycles. The fraction of negative arcs is about 30% and $r = n$. The second type of problems, Frac-Five, is generated in the same way, but r is kept at 5 % of n. The third type of problems, Acyc-Neg, consists of acyclic graphs with negative arc lengths, and $r = n$. They were generated by using the generator SPACYC, designed by Cherkassky et al. The arc lengths for Rand-Mix and Frac-Five problems are selected from the interval [-10 000, 10 000], and from the interval [-10 000, 0] for Acyc-Neg problems.

Tables 1, 2 and 3 summarize the results of our experiments, for which a HP 9000 series 300/800 computer was used. In each table entry the running time in seconds is given above and (in parentheses) the number of scan operations per node below (except for SCAL where we applied Dijkstra's algorithm only in phase 2 on a graph with nonnegative arc lengths, and thus the number of scan operations per node is exactly one). The running time is the user CPU time and excludes the input and output times. For each problem instance we did three runs and took the average over those three runs. The running times of the different runs did not fluctuate much.

Table 1. Random-mix data.

nodes/arcs	DIJK	GOR1	BFP	SCAL
2000 40000	73.18 (156.5)	0.21 (4.1)	0.09 (2.2)	4.90
4000 80000	256.19 (238.6)	0.52 (4.4)	0.23 (2.4)	11.26
8000 160000	1063.80 (423.5)	1.12 (4.5)	0.53 (2.6)	26.48

Table 2. Frac-Five data.

nodes/arcs	DIJK	GOR1	BFP	SCAL
2000 40000	4.43 (11.8)	0.22 (4.0)	0.09 (2.0)	3.00
4000 80000	14.21 (16.5)	0.54 (4.4)	0.23 (2.2)	6.57
8000 160000	50.50 (25.0)	1.34 (4.7)	0.50 (2.2)	14.13

Table 3. Acyc-Neg data.

nodes/arcs	DIJK	GOR1	BFP	SCAL
2000 40000	56.86 (120.9)	0.13 (2.0)	8.98 (298.4)	5.50
4000 80000	208.58 (193.0)	0.25 (2.0)	36.48 (588.3)	12.08
8000 60000	771.17 (307.9)	0.50 (2.0)	148.26 (1160.1)	28.65

In SCAL almost 95% of the user CPU time was spent in phase 1, in DIJK more than 99%, except for the Frac-Five problems where DIJK spent around 95% of the user CPU time in phase 1.
From the tables we see that DIJK performs less well than expected. SCAL already is a great improvement compared to DIJK, but is still much slower than GOR1. To us the problems of type Random-Mix seem more difficult to solve than those of type Acyc-Neg, but BFP performs much better on the first two types of problems (even better than GOR1). We like to remark that although the number of scan operations per node used by DIJK is smaller than those used by BFP in the case of Acyc-Neg, DIJK takes much more time to solve the problem. The results of tables 1 and 2 indicate that the running times of DIJK are more dependent on r than those of SCAN.

6. Conclusion

We have described a solution method for the one to all shortest path problem in a graph with arbitrary arc lengths which also detects negative cycles. This is done in two phases, by transforming the problem into a problem with only nonnegative arc lengths, followed by applying Dijkstra's method to this new problem. In phase 1 a linear assignment problem is solved. If this is done by a shortest augmenting

path method, one needs to solve shortest path problems with nonnegative arc lengths a moderate number of times. As a consequence the efficiency of the algorithm depends in both phases on the implementation of Dijkstra's method. Different implementations can be applied for dense graphs and for sparse graphs. Therefore the worst case complexity of our algorithm is at least as good as the O(mn) worst case complexity of all presently published algorithms, such as Bellman - Ford - Moore's algorithm (Ford and Fulkerson, 1962) or its improved version by Goldberg and Radzik (Goldberg and Radzik, 1993). In spite of this theoretical advantage, our current implementation of DIJK is not yet competitive with other algorithms for the shortest path problem. Exploitation of the property that (in the ordinary case of absence of negative cycles) the optimal solution is known, will probably improve the efficiency of DIJK. A substantial improvement of running time should be possible if one pays more attention to a careful implementation of phase 1.

References:

R.E. Bellman (1958), "On a routing problem", *Quart. Appl. Math. 16*, p. 87 - 90.

R.E. Burkard and U. Derigs (1980), *"Assignment and Matching Problems: Solution Methods with FORTRAN-Programs"*, Springer, Berlin.

B.V. Cherkassky, A.V. Goldberg and T. Radzik (1996), "Shortest paths algorithms: Theory and experimental evaluation", *Math. Programming 73*, p. 129 - 174.

R.B. Dial (1969), "Algorithm 360: Shortest path forest with topological ordering", *Comm. ACM 12*, p. 632 - 633.

E.W. Dijkstra (1959), "A note on two problems in connection with graphs", *Numerische Mathematik 1*, p. 269 - 271.

B. Dorhout (1977), "Experiments with some algorithms for the linear assignment problem", Report BW 39, Stichting Mathematisch Centrum, Amsterdam.

L.R. Ford Jr. and D.R. Fulkerson (1962), *"Flows in Networks"*, Princeton Univ. Press, Princeton, NJ.

M.L. Fredman and R.E. Tarjan (1987), "Fibonacci heaps and their uses

in improved network optimization algorithms", *J. ACM 34*, p. 596 - 615.

A.V. Goldberg and R. Kennedy (1995), "An efficient cost scaling algorithm for the assignment problem", *Math. Programming 71*, p. 153 - 177.

A.V. Goldberg and T. Radzik (1993), "A heuristic improvement of the Bellman - Ford algorithm", *Applied Math. Let. 6*, p. 3 - 6.

A.V. Goldberg and R.E. Tarjan (1990), "Finding minimum-cost circulationsby successive approximation", *Math. Op. Res. 15*, p. 430 - 466.

E.F. Moore (1959), "The shortest path through a maze", in: *Proceedings of the Int. Symp. on the Theory of Switching*, Harvard University Press, p. 285 - 292.

N. Tomizawa (1971), "On Some Techniques Useful for Solution of Transportation Network Problems", *Networks 1*, p. 173 - 194.

H.M. Wagner (1975), *"Principles of Operations Research"*, 2nd edition, Prentice-Hall, Englewood Cliffs.

R.A. Wagner (1976), "A shortest path algorithm for edge-sparse graphs", *J. ACM 23*, p. 50 - 57.

(1) ftp.bilkent.edu.tr at /pub/IEOR/Opt/goldberg directory.

Ontvangen: 25-11-1996
Geaccepteerd: 24-12-1997