Available online at www.sciencedirect.com
ScienceDirect
Transportation Research Part B 41 (2007) 756–771
TRANSPORTATION RESEARCH PART B
www.elsevier.com/locate/trb

An improved solution algorithm for the constrained shortest path problem

Luis Santos [a], João Coutinho-Rodrigues [b], John R. Current [c,*]

[a] Superior Institute Bissaya Barreto, Bencanta, 3040 Coimbra, Portugal
[b] Department of Civil Engineering, Faculty of Sciences and Technology, Polo II, University of Coimbra, 3030 Coimbra, Portugal
[c] Department of Management Sciences, The Fisher College of Business, The Ohio State University, 632 Fisher Hall, 2100 Neil Avenue, Columbus, OH 43210-1144, USA

Received 11 November 2004; accepted 4 December 2006

Abstract

The shortest path problem is one of the classic network problems. The objective of this problem is to identify the least cost path through a network from a pre-determined starting node to a pre-determined terminus node. It has many practical applications and can be solved optimally via efficient algorithms. Numerous modifications of the problem exist. In general, these are more difficult to solve. One of these modified versions includes an additional constraint that establishes an upper limit on the sum of some other arc cost (e.g., travel time) for the path. In this paper, a new optimal algorithm for this constrained shortest path problem is introduced. Extensive computational tests are presented which compare the algorithm to the two most commonly used algorithms to solve it. The results indicate that the new algorithm can solve optimally very large problem instances and is generally superior to the previous ones in terms of solution time and computer memory requirements. This is particularly true for the problem instances that are most difficult to solve. That is, those on large networks and/or where the additional constraint is most constraining.

Keywords: Network routing; Constrained shortest path problem; Exact algorithms

1. Introduction

The Shortest Path (SP) problem is one of the oldest and most widely used problems in network optimization (Dijkstra, 1959; Dantzig, 1960; Floyd, 1962). The objective of the SP problem is to identify the least cost path (or route) through a network from a pre-determined starting node to a pre-determined terminus node. The SP problem is one of the relatively few network optimization problems for which exact, polynomial time solution algorithms exist (e.g., see Magnanti and Wong, 1984; Evans and Minieka, 1992; Daskin, 1995).

* Corresponding author. Tel.: +1 614 292 3166; fax: +1 614 292 1272.
E-mail addresses: lsantos@dec.uc.pt (L. Santos), coutinho@dec.uc.pt (J. Coutinho-Rodrigues), current.1@osu.edu (J.R. Current).

0191-2615/$ - see front matter
doi:10.1016/j.trb.2006.12.001

Various "constrained" versions of the basic SP problem exist. For example, the path may be constrained to include specific nodes, or be constrained to include a specific number of nodes (e.g., see Deo and Pang, 1984), or include nodes within a pre-specified "covering" distance of every node in the network (Current et al., 1984; Current et al., 1994). In general, the term "constrained shortest path problem" (cSP) refers to the constrained problem in Handler and Zang (1980) that includes one additional constraint that establishes an upper limit on the sum of some other arc cost (e.g., travel time) for the path. Unfortunately, the addition of such constraints to the SP problem generally results in a problem that belongs to the set of problems known as NP-hard (e.g., see Garey and Johnson, 1979).

In this article, we introduce an improved exact algorithm for the cSP as defined in Handler and Zang (1980). Although this problem is NP-hard, we demonstrate that very large problem instances (40 000 nodes and 800 000 arcs) can be solved in reasonable time. We compare this new algorithm's results and solution times to the Lagrangian relaxation and the k-shortest path approaches presented in Handler and Zang (1980). The results of these tests indicate that the proposed algorithm can solve large problems in reasonable time and has advantages over the other methods in terms of solution time and computer memory requirements.

2. Mathematical formulation of the cSP problem

Consider a directed graph, $G = (N, A)$ where $N = \{1, 2, \ldots, n\}$ represents the set of nodes and $A = \{(i,j)\colon i, j \in N, i \neq j\}$ represents the set of m directed arcs. Two, non-negative weights c_{ij} and t_{ij} are associated with each arc (i,j). These weights may represent, for example, the length (or cost) and the travel time associated with the respective arc. Label the origin node, 1, and the destination node, n.

Given these definitions, the cSP problem may be formulated as a binary integer program as follows:

$$\min \quad \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{j} x_{ij} - \sum_{k} x_{ki} = \begin{cases} 1 & \text{if} \quad i = 1, \\ 0 & \text{if} \quad i = 2, \ldots, n-1, \\ -1 & \text{if} \quad i = n, \end{cases} \tag{2}$$

$$\sum_{(i,j)\in A} t_{ij} x_{ij} \leqslant T \tag{3}$$

$$x_{ij} \in \{0, 1\}, (i,j) \in A, \tag{4}$$

where x_{ij} are binary variables associated with each arc (i,j). If arc (i,j) is included in the optimal path, then $x_{ij} = 1$; otherwise $x_{ij} = 0$. The parameter, T, represents the maximum value allowed for the sum of the t_{ij} arc weights. For example, T could represent the maximum allowable time to transverse the path. The shortest path problem is formulated by (1), (2), and (4). As noted before, this problem can be solved optimally in polynomial time. Unfortunately the addition of constraint (3) results in a problem that belongs to the set on NP-hard problems (Garey and Johnson, 1979).

3. Existing solution algorithms for the cSP problem

Handler and Zang (1980) proposed two methods to solve the cSP problem. One requires solving a k-Shortest Path (kSP) problem and the other is based upon Lagrangian relaxation. Other approaches (e.g., Johsch, 1966, which is based on dynamic programming), have been proposed. However, the Handler and Zang Lagrangian relaxation-based method is generally considered the most efficient.

3.1. Algorithm 1: k-Shortest path method

This algorithm was introduced in Handler and Zang (1980). It identifies the optimal solution to the cSP by solving a k-Shortest Path (kSP) problem. The objective of the kSP is to identify k paths from the origin node to

the destination node. These k paths are sorted by ascending length from the shortest to the kth shortest. In essence, Algorithm 1 evaluates successive shortest paths, sorted by (5), until the first path that is feasible to constraint (3) is obtained.

Algorithm 1 can be summarized as follows.

Let $X = \lfloor x_{ij}: (i,j) \in A \rfloor$ be a vector whose components are, x_{ij} for all (i,j), and let Y be the set of vectors X satisfying the set of Eqs. (2) and (4). Define the following for any $X \in Y$:

$$f_1(X) = \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{5}$$

and

$$f_2(X) = \sum_{(i,j)\in A} t_{ij}x_{ij}. \tag{6}$$

An optimal solution to the cSP may be attained by applying an algorithm that solves kSP with (5) as the objective function until it identifies the first path $X_k \in Y$ such that $f_2(X_k) \leqslant T$ (Handler and Zang, 1980).

The procedure is presented graphically in Fig. 1 where the solutions, $X_k \in Y$, are represented by points in a two-dimension space. The horizontal axis represents $f_1(X)$ and the vertical axis represents $f_2(X)$. Therefore, the coordinates of each solution are. In the example presented in Fig. 1 the optimal solution is X_4, which is obtained in the 4th iteration of the algorithm (i.e., $k = 4$).

Unfortunately, the computational complexity of kSP problem increases with k. Several algorithms have been developed to solve the kSP problem (e.g., Dreyfus, 1969; Martins, 1984; Azevedo et al., 1993). The main disadvantage of Algorithm 1 is that the kSP algorithm may require a large value for k before a feasible solution to the cSP is obtained. As a consequence, the required computer time and memory may be unacceptable for many real-world problems.

3.2. Algorithm 2: Lagrangian relaxation method

Due to the computational complexity of the kSP based algorithm, Handler and Zang (1980) introduced a Lagrangian relaxation approach to solve the cSP. Their approach may be summarized as follows.

Given the previous definitions, define the following function for any $X \in Y$:

$$f_3(X) = \sum_{(i,j)\in A} t_{ij}x_{ij} - T. \tag{7}$$

The cSP problem may now be stated as:

$$f^* = f_1(X^*) = \min_{X\in Y} f_1(X) \tag{8}$$

$$\text{s.t.} \qquad f_3(X) \leqslant 0, \tag{9}$$

where X^* represents the optimal solution and f^* is the optimum value for the objective function.

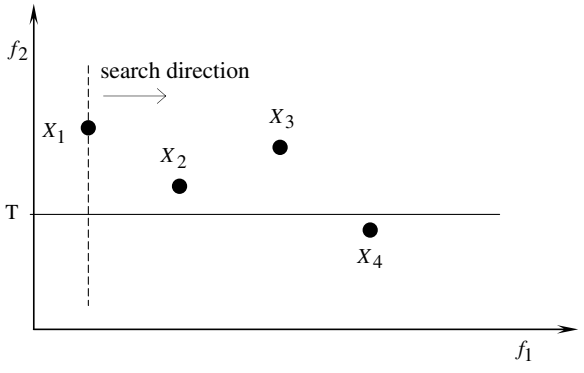

Fig. 1. kSP-based search approach.

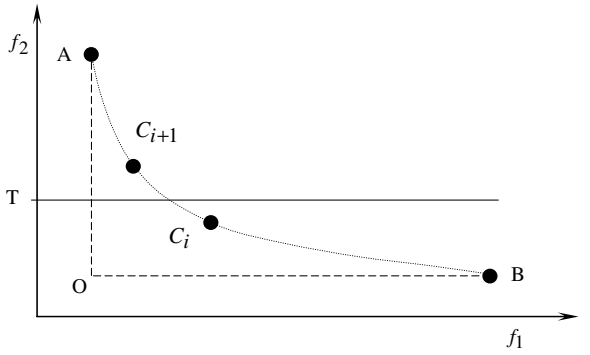

Fig. 2. Lagrangian relaxation algorithm 1st step.

The technique relaxes constraint (9). Therefore, for any $u \in R$ and $X \in Y$, the Lagrangian function is defined by

$$L(u,X) = f_1(X) + uf_3(X) \tag{10}$$

and

$$L(u) = \min_{X \in Y} L(u,X). \tag{11}$$

From (8) and (10), the following property is true:

$$L(u) \leqslant f^* \quad \forall\, u \geqslant 0. \tag{12}$$

As a consequence, a lower bound for the objective function of the primal problem may be attained by solving the following dual problem:

$$L^* = L(u^*) = \max_{u \geqslant 0} L(u). \tag{13}$$

Although the dual problem is convex, the primal problem is not. Consequently, $L^* < f^*$.

The details of Algorithm 2 are not presented here as the computational results presented in this article utilize a procedure identical to that presented in Handler and Zang (1980). However, the algorithm may be summarized as follows. The procedure begins by solving two shortest path problems to identify two solutions, A and B, that minimize respectively (5) and (6). It continues by identifying optimal solutions to (10) with new values of u. The initial value of u is $(f_1(\mathrm{B}) - f_1(\mathrm{A}))/(f_2(\mathrm{A}) - f_2(\mathrm{B}))$. This phase stops when the two closest solutions to the line defined by $f_2(X) = T$ are identified (e.g., C_i and C_{i+1} in Fig. 2). The first phase only identifies solutions on the convex hull of the cSP problem (represented by the curve in Fig. 2).

Given that the primal problem is not convex, the dual problem (13) may not identify the optimal solution to the primal problem. Consequently, Algorithm 2 invokes a k-shortest path solution procedure to identify any "non-convex" feasible solutions (i.e., any that may exist between C_i and C_{i+1} in Fig. 2). This second phase identifies the best (if any exist) solution in the "duality gap" between C_i and C_{i+1} by solving a kSP problem whose objective function is defined by (10) using the final value of u from the previous phase of the algorithm (i.e., the search direction is orthogonal to $\overline{C_i C_{i+1}}$).

4. New solution algorithm for the cSP problem

A new optimal solution algorithm for the cSP is introduced in this section. This algorithm utilizes a kSP algorithm in a manner similar to Algorithm 1. The primary difference is the way in which the k-shortest paths are sorted (i.e., the evaluation/ranking of $X_k \in Y$). The underlying concept of the new approach is to define a more efficient search direction for the kSP problem that considers the two network costs involved ($c_{i,j}$ and $t_{i,j}$) as well as the relative importance (i.e., "tightness") of the additional constraint (3).

In Algorithm 1, the $X_k \in Y$ are sorted by the value of $f_1(X)$. Hypothetical results of this approach are shown in Fig. 1. The new algorithm sorts the $X_k \in Y$ by an evaluation/ranking of the paths that considers both $f_1(X)$ and $f_2(X)$. It does this by ranking the paths by $f_4(X)$ defined in (14).

$$f_4(X) = f_1(X) + wf_2(X). \tag{14}$$

The underlying assumption of Algorithm 3 is that the kSP search will be more efficient (i.e., identify the optimal solution to the cSP with a lower value of k) if the search direction is orthogonal to the tangent of the convex hull of the cSP solution set where $f_2(X) = T$. Unfortunately, the convex hull of the cSP solution set is unknown in advance of solving the kSP problem. Consequently, w is defined to approximate this direction. To accomplish this, we first identify the solutions which minimize $f_1(X)$ and $f_2(X)$. As was the case in Algorithm 2, these solutions are obtained by solving two SP problems and are labeled A and B, respectively. Given these solutions, we define w as:

$$w = \frac{(1-\sqrt{p})[f_1(\mathrm{B}) - f_1(\mathrm{A})]}{\sqrt{p}[f_2(\mathrm{A}) - f_2(\mathrm{B})]}, \tag{15}$$

where,

$$p = \frac{T - f_2(\mathrm{B})}{f_2(\mathrm{A}) - f_2(\mathrm{B})}. \tag{16}$$

The parameter, p, is a problem specific ratio that measures the "tightness" of constraint (3) such that $0 < p < 1$. The more constraining (or "tight") the value of T in constraint (3), the lower the value of p will be (e.g., see Fig. 3). Fig. 3 demonstrates the desired search direction if we knew the convex hull of the cSP solution set. If p is small, the kSP should put a heavier weight on $f_2(X)$ and if p is large, the search should put a heavier weight on $f_1(X)$. The different kSP search directions for the various values of T shown in Fig. 3, result in the identification of feasible solutions to constraint (3) (i.e., under T in Fig. 3) at lower values of k. Consequently, a kSP algorithm using these weights will identify the optimal solution in fewer iterations of the algorithm than will the search direction used in Algorithm 1.

The weighting scheme generated by (15) approximates the desired search weight. If p is small (i.e., constraint (3) is tight) then (14) will result in a larger weight on $f_2(X)$ in the kSP procedure. If p is large, then (14) will result in a smaller weight on $f_2(X)$.

The computational results presented in the next section indicate that the underlying assumption of Algorithm 3 is valid and that the value or w selected is effective. This is especially true for problems with low values of p where constraint (3) is "tighter".

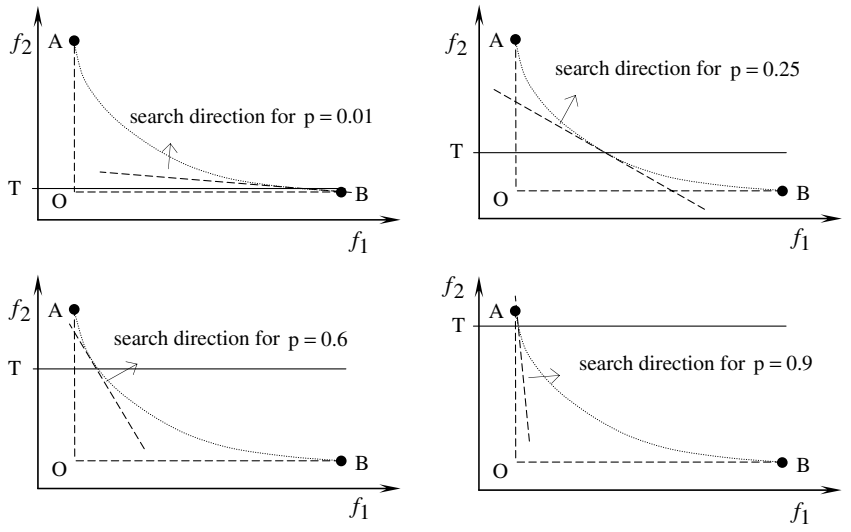

Fig. 3. Relationship of the search direction to the value of p.

Recall that Algorithm 1 terminates when it identifies the first $X_k \in Y$ such that $f_2(X) \leqslant T$. This is not the case with Algorithm 3 as the first such X_k may not be optimal to the cSP. The search procedure and stopping rule for Algorithm 3 are first described graphically and then more formal descriptions are given.

From the definitions given above of solutions A and B, no feasible solutions exist to the left of $\overline{OA}$ and below $\overline{OB}$ in Figs. 4–6. Solutions A and B are now used to calculate the values for p and w using (16) and (15), respectively. The kSP problem for $k = 1$ is now solved with (14) as the objective function. This results in solution X_1 shown in Fig. 4. This is a feasible solution; however, it may not be the optimal cSP solution. Consequently, it is stored as a candidate to be the optimal solution. If a better solution, say X_i exists (i.e., $f_1(X_i) < f_1(X_1)$ and $f_2(X_i) \leqslant T$) then it will be located inside the shaded triangle in Fig. 4.

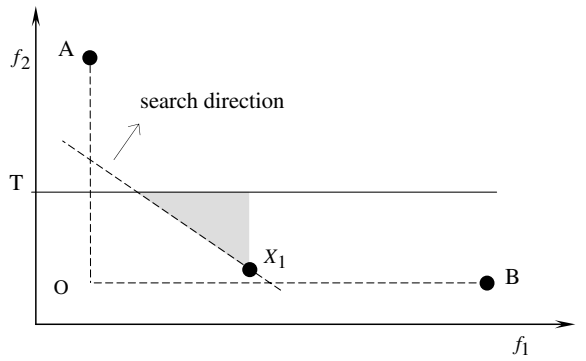

Fig. 4. Approach based on the new kSP evaluation (1st iteration).

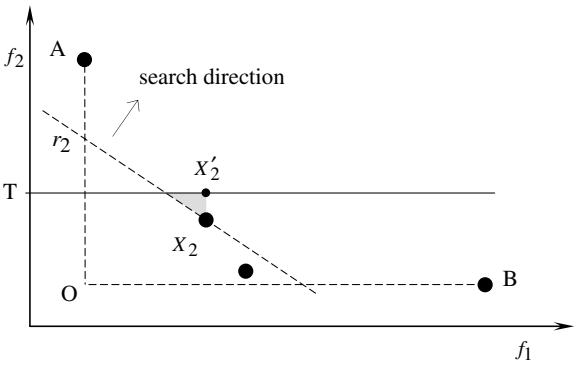

Fig. 5. Approach based on the new kSP evaluation (2nd iteration).

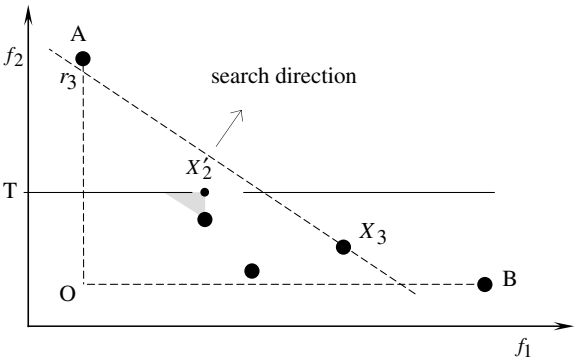

Fig. 6. New approach based on the kSP evaluation (stopping condition).

The $k = 2$ path in now identified. It is labeled X_2 in Fig. 5. As $f_2(X_2) \leqslant T$ it is a feasible solution to the cSP problem. It is in the shaded area of Fig. 4; therefore, $f_1(X_2) < f_1(X_1)$ and it replaces X_1 as the candidate to be the optimal solution to the cSP problem. If a feasible solution, say X_i, exists such that $f_1(X_i) < f_1(X_2)$, then it must be located in the shaded triangle in Fig. 5.

Algorithm 3 continues in this manner by setting $k = k + 1$ at each iteration until the stopping condition is satisfied. The stopping condition is based upon the value of the search direction (represented by the dashed line segment, r_k, in Figs. 5 and 6) and X'_j. Every time that the candidate to the optimal solution, X_j, is updated, a new "stopping point" (X'_j) is identified. The coordinates for this point are $f_1(X_j)$ and T. The algorithm terminates when the line r_k (which is a mapping of the search direction function (14) that includes X_k) intercepts the line $f_2(X) = T$ to the right of the current "stopping point", (X'_j). When this occurs, X_j is the optimal solution to the cSP as no solution X_i exists where $f_1(X_i) < f_1(X_j)$ and $f_2(X_i) \leqslant T$ (i.e., in the shaded area of Figs. 4–6). This is demonstrated in the example in Fig. 6 where the line $r_k(k = 3)$ intercepts the line $f_2(X) = T$ to the right of the current "stopping solution" (i.e., X'_2). Consequently, the optimal solution to this cSP is X_2.

A more formal statement of Algorithm 3 follows. Let $\mathrm{SP}(a, b)$ and $k\mathrm{SP}(a, b)$, respectively represent the shortest path and the k-shortest path between the origin node and the destination node, calculated by considering the costs associated with the arcs being $(ac_{ij} + bt_{ij})$. Let BS denote the best cSP problem solution evaluated so far and S^* denote the optimal cSP problem solution. Given these definitions, Algorithm 3 may be stated as follows:

A $\leftarrow$ SP $(1, 0)$
B $\leftarrow$ SP $(0, 1)$
Case 1: **If** $(T \geqslant f_2(\mathrm{A}))$ **then**
  $S^* \leftarrow$ A
Case 2: **If** $(T < f_2(\mathrm{B}))$ **then**
  Infeasible problem
Case 3: **If** $(T = f_2(\mathrm{B}))$ **then**
  $BS \leftarrow$ B
  $k \leftarrow 1$
  $X \leftarrow k\mathrm{SP}\ (0, 1)$
  While $(T = f_2(X))$
    If $(f_1(X) < f_1(BS))$ **Then**
      $BS \leftarrow X$
    End If
    $k \leftarrow k + 1$
    $X \leftarrow k\mathrm{SP}\ (0, 1)$
  End While
  $S^* \leftarrow BS$
 End if
Case 4: **If** $(T < f_2(\mathrm{A})$ and $T > f_2(\mathrm{B}))$ **then**
  $w = \dfrac{(1 - \sqrt{p})[f_1(\mathrm{B}) - f_1(\mathrm{A})]}{\sqrt{p}[f_2(\mathrm{A}) - f_2(\mathrm{B})]}$
  $BS \leftarrow$ B
  $k \leftarrow 1$
  $X \leftarrow k\mathrm{SP}\ (1, w)$
  While $((w(f_2(X) - T) + f_1(X)) < f_1(BS))$
    If $(f_2(X) \leqslant T$ and $f_1(X) < f_1(BS))$ **Then**
      $BS \leftarrow X$
    End If
    $k \leftarrow k + 1$
    $X \leftarrow k\mathrm{SP}\ (1, w)$
  End While
  $S^* \leftarrow BS$
 End if

5. Computational comparisons of the solution algorithms

The goal of the computational testing was to compare the three algorithms on two criteria: solution time and computer memory requirements. The three algorithms were encoded and tested on 180 randomly generated networks of various sizes. Algorithm 1 requires solving the k-shortest path problem and Algorithms 2 and 3 require solving the k-shortest path problem as well as solving shortest path problems. Several algorithms have been developed to solve these problems optimally. For example, Cherkassky et al. (1996) have implemented and compared 17 variations of various shortest path algorithms using experimental networks. Zhan and Noon (1998) have implemented and compared 15 of those 17 algorithms using real networks. These comparisons indicate that when negative arc costs exist, the Bellman-Ford-Moore algorithm (Bellman, 1958; Ford and Fulkerson, 1962; Moore, 1959) is the most efficient; and when they do not exist, the Dijkstra (1959) algorithm is the most efficient. Given that the test networks used in this research do not contain negative arcs, the tests reported in this paper solve the shortest path problems via Dial's (1969) implementation of the Dijkstra algorithm. Some adaptations were made to the original algorithm in order to assign real numbers for the arc costs (e.g., algorithms 2 and 3 assign $ac_{ij} + bt_{ij}$ to arc costs).

Various solution algorithms have also been developed for the k-shortest path problem (e.g., Dreyfus, 1969; Martins, 1984). Azevedo et al. (1993) compare these algorithms and present a new one. They prove that the new algorithm is computationally more efficient than the previous ones in terms of solution time. The improved version of this kSP solution algorithm (Azevedo et al., 1994) was used in testing the three algorithms in this article.

One hundred and eighty networks were randomly generated. There were 18 different network sizes based upon the number of nodes and arcs in the networks. Specifically, the networks contained 10 000 nodes, 20 000 nodes, or 40 000 nodes. For each one of these three "node sizes", there were six possible "number of arcs". For the 10 000 node problems, these were 15 000; 25 000; 50 000; 100 000; 150 000; and 200 000 arcs. For the 20 000 node problems, these were 30 000; 50 000; 100 000; 200 000; 300 000 and 400 000 arcs. For the 40 000 node problems, these were 60 000; 100 000; 200 000; 400 000; 600 000; and 800 000 arcs. For each of these 18 network sizes, 10 random networks were generated. Consequently, 180 networks were generated randomly. Each of these networks contains at least one Hamiltonian cycle. This guarantees that the network is connected and that there exists at least one path from each node to every other node in the network. The values of $t_{i,j}$ and $c_{i,j}$ for each arc (i,j) are uniformly distributed random integers in the range [1, 500]. The number of arcs incident to each node is a uniformly distributed random integer in the range [1, # of nodes-1]; where # of nodes = 10 000; 20 000; or 40 000 depending on the problem set. The direction of each arc was also determined randomly.

In addition, a value for T (i.e., the RHS of constraint (3)) was needed for each problem. The relative value of T may greatly influence the difficulty of solving the problem (and therefore the relative efficiency of a particular solution algorithm). For example, if T is relatively large, then the kSP solution for $k = 1$ might be optimal to the cSP. However, if T is relatively "tight", then it may require a very high value of k to identify the optimal cSP solution. As a consequence, T is an important problem input used to determine various parameters used in Algorithm 3. These parameters include w (15) and p (16) used to determine the kSP search direction (14). To evaluate the influence of T on the efficiency of the three algorithms, each of the 180 networks was solved for five values of T. These values were determined for each randomly generated network by "reverse-engineering". That is, 5 values of p were selected ($p = 0.1, 0.2, 0.4, 0.6$, and 0.8) and these values and the appropriate network values were then used in (16) to determine five values of T for the various networks. The tests were executed on a Pentium III @ 1 GHz with 512 K RAM.

Results of the 900 test problems are summarized in Tables 1–5. The average and maximum solution time of the 10 problems for each particular network size and p value are presented in columns 3–5. Algorithm 1 solves a kSP algorithm only for each test problem. Algorithm 2 solves a varying number of shortest path problems (at least 2, to determinate A and B) and a kSP algorithm for each test problem. Algorithm 3 solves 2 shortest path problems (to determine A and B) and a kSP algorithm for each test problem. The average and maximum values of k used by the kSP algorithms and the number of shortest path problems solved (in Algorithms 2 and 3) are presented in columns 7, 9, 10, 12 and 13. Again, these represent the average or maximum value for the 10 random networks for each given network size and p value.

Table 1
Computational results for $p = 0.1$

Nodes (#n)	Arcs (#m)	Average computer time (s)			Number of SP problems solved and/or k-paths identified								
		Alg. 1 (Av/max)	Alg. 2 (Av/max)	Alg. 3 (Av/max)	Alg. 1			Alg. 2			Alg. 3		
					#NOpt	#k (Av/max)	#n + #m (max)	#SP (Av/max)	#k (Av/max)	#n + #m (max)	#SP	#k (Av/max)	#n + #m (max)
10 000	15 000	0.2/0.2	0.9/1.1	0.6/0.7	0	28.0/188	42 185	3.7/5	2.4/4	25 344	2	1.5/2	25 211
	25 000	1.3/7.6	1.2/1.7	0.7/0.7	1	14 600.1/102 625	6 300 343	4.1/6	3.4/8	35 421	2	3.8/10	35 483
	50 000	6.4/15.2	2.1/2.5	1.1/1.1	4	82 861.3/200 412	13 041 969	4.8/6	3.5/9	60 548	2	3.1/12	60 684
	100 000	20.8/24.4	3.3/3.9	1.8/1.8	9	193 867.9/226 577	23 096 315	5.1/6	4.3/9	110 682	2	8.1/44	113 444
	150 000	33.1/34.9	4.8/5.6	2.5/4.5	10	208 271.4/233 557	34 423 347	5.8/7	6.4/11	160 998	2	6.0/14	161 464
	200 000	41.9/44.2	6.1/7.7	2.7/2.8	10	216 208.5/246 065	41 995 831	6.1/8	12.9/70	217 715	2	28.0/222	234 555
20 000	30 000	0.4/0.4	1.6/2.0	1.2/1.2	0	13.6/60	56 478	3.1/4	2/2	50 293	2	1.1/2	50 258
	50 000	0.5/0.8	2.7/3.5	1.5/1.5	0	484.2/4006	330 087	4.4/6	2.6/4	70 250	2	1.6/4	70 243
	100 000	7.4/14.7	4.5/5.8	2.3/2.3	4	76 272.7/158 825	12 278 040	5.0/7	3.1/6	120 350	2	4.2/14	120 967
	200 000	24.7/25.4	7.0/8.1	3.7/3.8	10	185 753.6/204 781	23 114 979	5.0/6	4.3/11	220 944	2	4.7/21	221 626
	300 000	35.6/37.0	10.2/12.7	4.8/4.9	10	188 919.2/210 722	34 831 158	5.6/7	7.2/22	322 404	2	8.3/31	323 244
	400 000	43.4/43.8	12.5/14.6	5.9/5.9	10	182 827.4/211 594	41 964 396	5.7/7	7.2/23	423 676	2	14.9/92	432 826
40 000	60 000	0.8/0.8	3.7/4.0	2.4/2.4	0	45.8/214	124 585	3.6/4	2.2/3	100 385	2	1.7/3	100 385
	100 000	1.0/1.1	5.3/7.1	3.0/3.1	0	463.8/2043	286 024	4.3/6	3.2/5	140 346	2	2.5/6	140 447
	200 000	7.9/15.4	9.6/11.2	4.9/5.1	4	67 850.7/160 305	12 287 580	5.2/6	5.2/10	240 708	2	4.9/10	240 708
	400 000	25.9/27.3	15.9/20.4	7.7/7.8	10	165 318.7/184 661	23 221 549	5.3/7	8.2/34	442 884	2	8.0/34	442 884
	600 000	37.9/40.2	23.1/26.5	10.7/10.8	10	180 591.8/192 849	34 079 688	5.8/7	7.3/17	642 222	2	20.5/164	658 468
	800 000	46.2/46.9	27.0/32.4	13.0/13.3	10	165 170.4/183 976	41 975 266	5.6/7	8.9/32	844 091	2	7.8/25	843 241
Global average		18.6/21.1	7.9/9.5	3.9/4.1		107 197/140 192	16 964 203	4.9/6.2	5.2/15.6	233 873	2.0	7.3/39.4	234 103

Table 2
Computational results for $p = 0.2$

Nodes (#n)	Arcs (#m)	Average computer time (s)			Number of SP problems solved and/or k-paths identified								
		Alg, 1 (Av/max)	Alg. 2 (Av/max)	Alg. 3 (Av/max)	Alg. 1			Alg. 2			Alg. 3		
					#NOpt	#k (Av/max)	#n + #m (max)	#SP (Av/max)	#k (Av/max)	#n + #m (max)	#SP	#k (Av/max)	#n + #m (max)
10000	15000	0.2/0.2	0.9/1.2	0.6/0.6	0	27.5/188	42185	3.7/5	2.6/4	25344	2	3.2/5	25448
	25000	0.3/0.5	1.2/1.5	0.7/0.7	0	511.7/4749	303665	4.1/5	2.8/6	35319	2	3.5/10	35526
	50000	4.7/15.2	2.0/2.6	1.1/1.1	3	59825.2/200412	13041969	4.7/6	4.9/11	60699	2	4.7/14	60761
	100000	20.8/24.4	3.4/3.8	1.8/1.8	9	193560.1/226577	23096315	5.1/6	7.2/16	111116	2	7.1/19	111272
	150000	28.2/35.0	5.0/6.2	2.5/4.5	8	174741.5/233557	34423347	6.1/8	11.0/32	163073	2	10.6/54	165138
	200000	38.0/43.4	5.9/7.0	2.7/2.8	9	195889.5/246065	41995831	5.8/7	7.5/32	213980	2	12.4/37	214507
20000	30000	0.4/0.4	1.6/2.0	1.2/1.2	0	13.6/60	56478	3.1/4	2.1/3	50374	2	2.3/5	50597
	50000	0.5/0.8	2.7/3.5	1.5/1.5	0	451.7/4006	330087	4.4/6	3.8/11	70714	2	4.1/16	71041
	100000	4.1/14.7	4.3/5.2	2.3/2.3	2	38281.5/158825	12278040	4.8/6	3.4/5	120286	2	4.2/17	121010
	200000	17.6/25.0	6.9/8.1	3.7/3.8	6	129162.7/204781	22921261	5.0/6	6.6/14	221031	2	8.1/21	221713
	300000	32.5/36.9	9.8/10.8	4.8/4.9	9	171834.5/210722	33816393	5.3/6	7.2/14	321429	2	6.0/13	321407
	400000	40.3/50.9	12.9/15.1	5.9/5.9	9	166348.3/211594	41964396	5.9/7	7.7/18	422563	2	11.7/32	424538
40000	60000	0.8/0.8	3.8/4.8	2.4/2.4	0	23.4/80	109293	3.8/5	2.5/4	100504	2	2.8/7	100876
	100000	1.0/1.1	5.2/6.1	3.0/3.1	0	328.0/2 043	286024	4.2/5	3.9/6	140394	2	4.0/6	140394
	200000	5.2/15.4	9.7/11.4	4.9/5.1	2	39128.9/160305	12287580	5.2/6	5.1/11	240671	2	6.3/15	241088
	400000	22.4/27.0	16.8/20.4	7.7/7.8	7	140264.4/184661	22707356	5.6/7	9.7/25	442162	2	12.4/54	444703
	600000	29.4/38.5	21.8/25.0	10.7/10.9	7	135655.7/188822	34079688	5.3/6	6.6/17	642002	2	6.9/22	642809
	800000	42.8/46.9	28.2/32.5	13.1/13.6	9	151757.3/183976	41975266	5.7/7	8.6/21	842716	2	13.5/38	845820
Global average		16.1/21.0	7.9/9.3	3.9/4.1		88767/134524	14493773	4.9/6.0	5.7/13.9	233893	2.0	6.9/21.4	234024

Table 3
Computational results for $p = 0.4$

Nodes (#n)	Arcs (#m)	Average computer time (s)			Number of SP problems solved and/or k-paths identified								
		Alg. 1 (Av/max)	Alg. 2 (Av/max)	Alg. 3 (Av/max)	Alg. 1			Alg. 2			Alg. 3		
					#NOpt	#k (Av/max)	#n + #m (max)	#SP (Av/max)	#k (Av/max)	#n + #m (max)	#SP	#k (Av/max)	#n + #m (max)
10000	15000	0.2/0.2	0.9/1.2	0.6/0.6	0	7.2/26	27437	3.8/5	3.0/7	25605	2	2.7/6	25536
	25000	0.2/0.3	1.2/1.5	0.7/0.7	0	25.4/178	45377	4.0/5	2.9/5	35278	2	3.0/9	35504
	50000	1.8/12.2	2.0/2.2	1.1/1.1	1	20101.8/165989	10190307	4.5/5	4.2/8	60446	2	7.2/22	61075
	100000	2.0/8.1	3.4/3.8	1.8/1.8	0	13864.8/73295	7178463	5.2/6	5.5/12	110974	2	7.0/24	111897
	150000	11.0/35.0	4.8/5.3	2.5/4.5	3	65519.7/233557	34423347	5.6/6	6.1/16	161542	2	23.8/164	175287
	200000	12.1/44.4	5.9/7.1	2.7/2.8	2	58616.3/246065	41993291	5.7/7	4.7/16	211977	2	7.4/21	212553
20000	30000	0.4/0.4	1.6/2.0	1.2/1.3	0	13.6/60	56478	3.1/4	2.4/6	50721	2	3.1/8	50934
	50000	0.5/0.8	2.5/3.0	1.5/1.5	0	410.7/4006	330087	4.1/5	4.2/7	70521	2	4.7/12	70812
	100000	0.8/1.2	4.3/5.5	2.3/2.4	0	677.6/4733	498171	4.6/6	3.6/9	120547	2	3.3/12	120775
	200000	10.9/25.0	7.3/8.6	3.7/3.8	4	75822.5/204283	22921 261	5.2/6	6.6/14	221298	2	10.0/30	222676
	300000	12.5/36.1	10.1/13.3	4.9/5.0	3	58715.1/186556	33516419	5.2/7	5.0/10	321259	2	4.1/10	321259
	400000	15.0/43.8	14.0/16.8	5.9/6.0	3	60035.1/211594	41943601	6.0/7	4.0/9	421300	2	11.4/64	428990
40000	60000	0.8/0.8	3.8/4.8	2.4/2.4	0	13.2/80	109293	3.8/5	2.8/4	100522	2	3.0/6	100710
	100000	1.0/1.0	5.3/6.4	3.0/3.2	0	47.6/308	160910	4.2/5	3.4/9	140589	2	3.7/14	140983
	200000	1.7/2.3	9.8/11.8	4.9/5.0	0	1479.8/7235	864726	5.0/6	3.8/8	240492	2	4.2/10	240604
	400000	7.1/26.3	16.8/19.4	7.8/8.0	1	30915.1/161634	22288272	5.4/6	6.5/12	441042	2	6.8/16	441231
	600000	8.0/36.5	25.0/27.6	10.9/11.2	1	22905.9/170747	31862403	5.8/6	5.4/17	642072	2	6.1/13	641321
	800000	9.3/46.7	32.7/39.2	13.4/14.0	1	19254.0/169378	41918173	6.1/7	3.2/4	840917	2	11.2/40	845137
Global average		5.3/17.8	8.4/10.0	4.0/4.2		23801/102207	4088667	4.9/5.8	4.3/9.6	233761	2.0	6.8/26.7	234012

Table 4
Computational results for $p = 0.6$

Nodes (#n)	Arcs (#m)	Average computer time (s)			Number of SP problems solved and/or k-paths identified								
		Alg. 1 (Av/max)	Alg. 2 (Av/max)	Alg. 3 (Av/max)	Alg. 1			Alg. 2			Alg. 3		
					#NOpt	#k (Av/max)	#n + #m (max)	#SP (Av/max)	#k (Av/max)	#n + #m (max)	#SP	#k (Av/max)	#n + #m (max)
10000	15000	0.2/0.2	0.9/1.2	0.6/0.6	0	7.1/26	27437	3.8/5	3.5/7	25605	2	3.8/10	25931
	25000	0.2/0.3	1.2/1.5	0.7/0.7	0	6.4/28	36672	4.0/5	3.2/6	35341	2	3.4/9	35504
	50000	1.6/12.1	2.0/2.2	1.1/1.1	1	16648.7/165989	10190307	4.4/5	4.1/10	60448	2	7.2/31	61587
	100000	0.7/1.1	3.4/3.8	1.8/1.8	0	826.9/4589	550197	5.2/6	5.2/12	110851	2	5.5/14	111003
	150000	3.5/19.7	4.9/6.3	2.6/4.7	0	16925.6/120885	18988958	5.7/7	7.1/16	161542	2	21.1/144	173865
	200000	5.8/48.2	5.9/7.1	2.8/2.9	1	20355.2/200757	41993291	5.5/7	7.6/26	213773	2	12.9/38	214727
20000	30000	0.4/0.4	1.7/2.4	1.2/1.2	0	9.0/60	56478	3.2/5	2.8/5	50539	2	3.8/13	51403
	50000	0.5/0.5	2.6/3.0	1.5/1.5	0	5.0/24	71691	4.2/5	3.1/6	70425	2	3.0/5	70401
	100000	0.8/0.8	4.4/5.5	2.3/2.4	0	121.2/988	190313	4.7/6	4.3/11	120599	2	4.8/13	120848
	200000	2.1/8.5	7.2/8.8	3.8/3.8	0	6 396.9/55892	6923700	5.0/6	3.9/9	220935	2	5.0/11	221139
	300000	7.1/34.8	10.2/13.3	4.9/5.1	1	29558.3/186556	31714666	5.1/7	3.7/8	321133	2	4.9/12	321330
	400000	2.7/5.2	13.2/18.0	6.0/6.1	0	2928.4/15400	3441454	5.6/8	3.0/4	420984	2	13.1/92	434176
40000	60000	0.8/0.8	3.8/4.8	2.4/2.4	0	9.5/80	109293	3.8/5	3.0/6	100725	2	3.7/17	102013
	100000	1.0/1.0	5.2/6.4	3.1/3.2	0	36.0/308	160910	4.1/5	2.9/6	140461	2	3.0/8	140630
	200000	1.6/1.6	9.7/11.8	5.0/5.1	0	46.9/286	264757	4.9/6	3.8/9	240666	2	4.8/9	240706
	400000	2.6/2.8	16.3/20.1	8.0/8.2	0	215.7/1048	575336	5.0/6	4.7/11	440963	2	6.5/18	441834
	600000	4.0/7.7	25.1/28.0	11.2/11.6	0	2376.2/20278	4356715	5.7/6	4.0/10	641109	2	10.9/54	646652
	800000	8.6/46.6	32.6/39.3	13.8/14.4	1	16998.2/169378	41918173	5.8/7	4.5/16	842596	2	20.9/82	853058
Global average		2.4/10.7	8.4/10.2	4.0/4.3		6304/42955	1292850	4.8/5.9	4.1/9.9	233784	2.0	7.7/32.2	234189

Table 5
Computational results for $p = 0.8$

Nodes (#n)	Arcs (#m)	Average computer time (s)			Number of SP problems solved and/or k-paths identified								
		Alg. 1 (Av/max)	Alg. 2 (Av/max)	Alg. 3 (Av/max)	Alg. 1			Alg. 2			Alg. 3		
					#NOpt	#k (Av/max)	#n + #m (max)	#SP (Av/max)	#k (Av/max)	#n + #m (max)	#SP	#k (Av/max)	#n + #m (max)
10000	15000	0.2/0.2	0.9/1.2	0.6/0.6	0	5.9/26	27437	3.7/5	2.6/4	25416	2	4.2/14	26277
	25000	0.2/0.3	1.2/1.5	0.7/0.7	0	3.7/15	35874	4.0/5	3.2/5	35 283	2	3.5/7	35391
	50000	1.6/12.2	1.9/2.2	1.1/1.2	1	16601.8/165989	10190307	4.3/5	3.5/8	60426	2	34.8/321	76856
	100000	0.6/0.6	3.4/3.8	1.8/1.9	0	23.7/161	126763	5.1/6	4.0/13	110901	2	3.2/9	110673
	150000	0.8/1.2	5.0/6.2	2.8/4.9	0	277.0/2 526	515443	5.7/7	4.5/16	161533	2	7.1/26	162803
	200000	5.7/48.1	5.9/6.6	2.8/2.9	1	20086.8/200757	41993291	5.4/6	5.0/16	212201	2	7.8/34	215092
20000	30000	0.4/0.4	1.7/2.4	1.2/1.2	0	8.8/60	56478	3.2/5	3.4/8	50854	2	5.5/29	53149
	50000	0.5/0.5	2.6/3.1	1.5/1.6	0	3.4/17	71225	4.2/5	2.9/7	70471	2	2.9/10	70710
	100000	0.8/0.8	4.4/5.5	2.4/2.4	0	17.2/93	127195	4.8/6	4.3/9	120478	2	4.4/8	120587
	200000	1.3/1.7	7.3/8.9	3.9/4.7	0	477.1/3669	659600	5.0/6	5.4/13	221 053	2	5.7/19	221619
	300000	5.2/34.9	10.0/12.4	5.0/5.2	1	19696.0/186556	31714666	5.0/6	4.3/10	321 215	2	11.1/57	328244
	400000	2.1/2.3	13.0/17.0	6.2/6.3	0	114.0/1059	630496	5.4/7	4.3/9	421984	2	8.3/41	427429
40000	60000	0.8/0.8	3.8/4.8	2.4/2.4	0	9.2/80	109293	3.8/5	3.4/9	101042	2	5.6/35	104068
	100000	1.0/1.0	5.2/6.4	3.1/3.2	0	35.4/308	160910	4.1/5	3.5/6	140461	2	3.7/7	140445
	200000	1.6/1.6	9.5/11.0	5.1/5.2	0	41.5/286	264757	4.8/6	4.5/9	240610	2	4.7/10	240691
	400000	2.6/2.7	16.3/20.3	8.3/8.4	0	36.4/144	458980	5.0/6	8.6/21	441707	2	5.8/11	441320
	600000	3.8/6.3	25.4/29.4	11.7/12.0	0	1 331.4/13229	3065620	5.6/7	3.7/8	641036	2	6.4/32	644320
	800000	4.3/4.3	31.8/36.1	14.4/14.9	0	3.6/21	845948	5.5/6	2.6/5	840833	2	11.2/53	849633
Global average		1.9/6.7	8.3/9.9	4.2/4.4		3265/31944	727767	4.7/5.8	4.1/9.8	233780	2.0	7.6/40.2	234126

The algorithm (Azevedo et al., 1994) used in these tests is the fastest known algorithm for the kSP problem. However, it increases the network dimensions (i.e., adds nodes and arcs) as it solves the problem. As a result, the number of nodes ($\#n$), the number of arcs ($\#m$), and computer memory requirements (measured as the sum of $\#n + \#m$) increase as k increases. Due to constraints on computer RAM, the algorithm was terminated if it had not identified the optimal solution before $\#n$ reached 2 000 000, or $\#m$ reached 40 000 000, in the augmented network. These values were chosen as they approximate the upper limit of RAM capacity of the Pentium III @ 1 GHz with 512 K RAM computer used to solve the test problems. Larger values for the augmented network would require the use of "virtual" memory which would require the reading/writing of data on a hard disk and result in excessive solution times.

This limit was reached only by Algorithm 1. The number of times that this occurred in Algorithm 1 for each set of 10 random networks for a given network size and p value is presented in column 6 (#NOpt) of Tables 1–5. The other results for Algorithm 1 (e.g., average solution time, average and maximum values of k) reported in Tables 1–5 are, in effect, lower bounds for the problem sets which had one or more test problems terminated before the optimal solution was identified because they exceeded the upper limit on computer memory.

An underlying assumption in the development of Algorithm 3 is that one can improve the efficiency of the kSP solution approach (i.e., reduce the value of k needed to identify the optimal cSP solution) by redefining the search direction (14) of the kSP algorithm according to the definitions given in (15) and (16). The results presented in Tables 1–5 support the validity of this assumption. The average and maximum values of k for Algorithm 3 were less then the corresponding values for Algorithm 1 in every problem category. For most categories, the reductions were striking. For example, the "global averages" (i.e., for all 180 test problems) for the average and maximum k values for $p = 0.1$ were 7.3 and 39.4, respectively for Algorithm 3 versus 107 197 and 140 192 for Algorithm 1.

Algorithm 3's improved performance versus that of Algorithm 1 was most noticeable for the lower values of p. This was expected for two reasons. First, constraint (3) is the "tightest" in these problems as T is relatively lower. In this case, one would expect that higher values of k would be needed to identify the optimal solution using Algorithm 1. Also, the difference between the search directions is greatest for the lower values of p. Algorithm 3's search direction is similar to Algorithm 1's direction when p approaches 1.0.

A major advantage of Algorithm 3 over Algorithm 1 is that the reduced value of k greatly reduced the computer memory required. For example, the average and maximum values of $\#n + \#m$ for $p = 0.1$ were 234 103 and 843 241 respectively for Algorithm 3 versus 16 964 203 and 41 975 266 for Algorithm 1. Algorithm 1 failed to identify the optimal solution due to the memory termination condition in 23% of the 900 test problems versus 0% for Algorithm 3. The termination condition was invoked most often for lower values of p (e.g., 57% and 44% of the problems where $p = 0.1$ and 0.2, respectively).

It is difficult to compare the solution times of Algorithms 1 and 3 because Algorithm 1 terminated before it identified the optimal solution in 23% of the problems. Algorithm 3 performed best vis-à-vis Algorithm 1 in the larger problems sizes and those with the lowest values of p. These are the problem sets where Algorithm 1 most often did not identify the optimal solution. Had Algorithm 1 been allowed to run longer to identify the optimal solution, the solution times would have been even greater. Given this caveat, the ratios of the "global average" solution times (in seconds) for Algorithm 1/Algorithm 3 were: 4.77 (18.6/3.9) for $p = 0.1$, 4.13 (16.1/3.9) for $p = 0.2$, 1.33 (5.3/4.0) for $p = 0.4$, 0.60 (2.4/4.0) for $p = 0.6$ and, 0.45 (1.9/4.2) for $p = 0.8$. The ratios of the "global average" maximum solution times (in seconds) for Algorithm 1/Algorithm 3 were: 5.15 (21.1/4.1) for $p = 0.1$, 5.12 (21.0/4.1) for $p = 0.2$, 4.24 (17.8/4.2) for $p = 0.4$, 2.49 (10.7/4.3) for $p = 0.6$ and, 1.52 (6.7/4.4) for $p = 0.8$. As these values demonstrate, Algorithm 3's average worse case performance was better than that of Algorithm 1 for every value of p.

Computer memory requirements were very similar for Algorithms 2 and 3. For example, the average and maximum values of $\#n + \#m$ for $p = 0.1$ were 234 103 and 843 241, respectively for Algorithm 3 versus 233 873 and 844 091 for Algorithm 2. Both of these Algorithms identified the optimal solution to every problem well within the RAM termination values.

Regarding solution times, the "global average" solution times for Algorithm 3 were less than those of Algorithm 2 for every network size and p value. The averages for both algorithms were fairly consistent across p values as is demonstrated by the global averages. The ratios of the "global average" solution times (in seconds) for Algorithm 2/Algorithm 3 were: 2.03 (7.9/3.9) for $p = 0.1$, 2.03 (7.9/3.9) for $p = 0.2$, 2.10 (8.4/4.0)

for $p = 0.4$, 2.10 (8.4/4.0) for $p = 0.6$, and 1.98 (8.3/4.2) for $p = 0.8$. The maximum solution times demonstrate a similar pattern with Algorithm 2's average maximum times and maximum times for all problems being about double the values for Algorithm 3. For example, the ratios of the "global average" maximum solution times (in seconds) for Algorithm 2/Algorithm 3 were: 2.32 (9.5/4.1) for $p = 0.1$, and 2.27 (9.3/4.1) for $p = 0.2$.

6. Summary and conclusions

The shortest path problem is a practical and frequently used network routing problem and numerous efficient algorithms have been developed to solve it. Modifications to the shortest path problem are generally more difficult to solve. The constrained shortest path (cSP) problem includes a constraint that establishes an upper limit, say T, on the sum of some other arc weight (e.g., travel time). Handler and Zang (1980) proposed two algorithms (Algorithms 1 and 2) to solve the problem optimally. In essence, these Algorithms identify the optimal solution via a directed search. Algorithm 1 does this by solving a k-Shortest Path (kSP) problem with the same objective function, (1), as that of the underlying cSP problem. Algorithm 2's search is based upon a Lagrangian relaxation of the cSP with the search direction a linear combination, (10), of the original objective function (1) and the additional constraint (3).

This paper introduces a new optimal algorithm (Algorithm 3) to solve the cSP problem. The motivation of this research was to identify a more effective search direction. The underlying assumption of Algorithm 3 is that the kSP search will be more efficient (i.e., identify the optimal solution to the cSP with a lower value of k) if the search direction is orthogonal to the tangent of the convex hull of the cSP solution set where $f_2(X) = T$. Unfortunately, the convex hull of the cSP solution set is unknown in advance of solving the kSP problem. Consequently, this search direction is approximated based upon the relative "tightness" of constraint (3).

The three algorithms were compared on two criteria: solution time and computer memory requirements. The three algorithms were used to solve 900 test problems ranging in size from 10 000 to 40 000 nodes and 15 000 to 800 000 arcs. The tightness of constraint (3) has a direct impact on the difficulty of solving the cSP. For example, if (3) is not binding at the optimal solution, then the problem can be solved very efficiently as a simple Shortest Path problem. The 900 test problems were divided into 5 sets that had differing levels of tightness for constraint (3).

All three of the algorithms require solving a kSP problem. The kSP algorithm (Azevedo et al., 1994) used in these tests is the fastest known one for the problem. It solves the kSP problem by augmenting the original network with additional nodes and arcs. As a consequence, the augmented network can become extremely large as k increases requiring large amounts of RAM. Consequently, the computer memory required was measured in terms of the number of nodes plus the number of arcs in the augmented kSP network. To avoid the use of virtual memory requiring excessive solution times, an algorithm was terminated before it identified the optimal solution if it exceeded the RAM available. Algorithm 1 was terminated by this condition before it identified the optimal solution in 23% of the test problems. This occurred most frequently in the problem sets where constraint (3) is the tightest. Algorithms 2 and 3 were never terminated by this condition.

Algorithms 1 and 3 both solve the cSP directly via a kSP algorithm. The primary difference is in the search direction for the kSP. On average, Algorithm 3 required approximately 25% of the time required by Algorithm 1 for the problem sets where constraint (3) was the tightest (e.g., 3.9 vs. 18.6 s and 3.9 vs. 16.1 s) while Algorithm 1 required about 50% of the time of Algorithm 3 for the problem sets where constraint (3) was the least tight (e.g., 2.4 vs. 4.0 s and 1.9 vs. 4.4 s). However, the times for Algorithm 1 are somewhat misleading as the algorithm was terminated before finding the optimal solution in 23% of the test problems.

Algorithm 3 required considerably less computer memory than did Algorithm 1, especially, in the problem sets where constraint (3) was the tightest. The average memory requirements ranged from 16 964 203 for Algorithm 1 vs. 234 103 for Algorithm 3 in the problem sets where (3) was the tightest, to 727 767 for Algorithm 1 vs. 234 126 for Algorithm 3 in the problem sets where (3) was the least tight.

Solution time and memory comparisons between Algorithms 2 and 3 are less dramatic but they indicate that Algorithm 3 was superior to Algorithm 2 in terms of solution time in every problem set. Algorithm 3 required approximately half the time to solve the problems as did Algorithm 2. This advantage was generally

the greatest in the problems with the largest networks and the tightest constraint (3). Computer memory requirements were approximately the same for these two algorithms.

Analyses of the solution time and computer memory requirements presented in Tables 1–5 indicate that Algorithm 3 outperforms Algorithm 2 in terms of solution time in all problems and requires essentially the same computer memory. Algorithm 3 performs best vis-à-vis Algorithm 1 in the problem sets where constraint (3) is most constraining and in those with the largest networks. For example, average solution times for Algorithm 1/Algorithm 3 were 18.6/3.9, 16.1/3.9, and 5.3/4.0 in the three problem sets where constraint (3) is the tightest and were 2.4/4.0, and 1.9/4.2 for the two problem sets where constraint (3) is the least tight. Algorithm 3 required less computer memory than did Algorithm 1 in every problem set. This advantage was most dramatic in the problems with large networks and/or where constraint (3) was the most constraining. In general, these are the most difficult cSP problems to solve. Consequently, Algorithm 3 was superior to Algorithm 1 in terms of computer memory and solution times for the problems that are most difficult to solve. However, even in the easiest problems to solve where Algorithm 1 generally solved the problems faster than did Algorithm 3, Algorithm 1 required a maximum solution time of 48.1 s vs. a maximum of 14.9 s for Algorithm 3.

References

Azevedo, J.A., Costa, M., Madeira, J., Martins, E.Q.V., 1993. An algorithm for the ranking of shortest paths. European Journal of Operational Research 69, 97–106.

Azevedo, J.A., Madeira, J., Costa, M., Martins, E.Q.V., Pires, F., 1994. A computational improvement for a shortest paths ranking algorithm. European Journal of Operational Research 73, 188–191.

Bellman, R.E., 1958. On a routing problem. Quarterly of Applied Mathematics 16, 87–90.

Cherkassky, B.V., Goldberg, A.V., Radzik, T., 1996. Shortest paths algorithms: theory and experimental evaluation. Mathematical Programming 73, 129–174.

Current, J.R., ReVelle, C.S., Cohon, J.L., 1984. The shortest covering path problem: an application of locational constraints to network design. Journal of Regional Science 24, 161–185.

Current, J.R., Pirkul, H., Rolland, E., 1994. Efficient algorithms for solving the shortest covering path problem. Transportation Science 28, 317–327.

Dantzig, G.B., 1960. On the shortest route through a network. Management Science, 187–190.

Daskin, M.S., 1995. Network and Discrete Location. John Wiley and Sons, New York.

Deo, N., Pang, C., 1984. Shortest-path algorithms: taxonomy and annotation. Networks 14, 275–323.

Dial, R.B., 1969. Algorithm 360: shortest path forest with topological ordering. Communications of the ACM 12, 632–633.

Dijkstra, E.W., 1959. A note on two problems in connection with graphs. Numerische Mathematik 1, 269–271.

Dreyfus, S., 1969. An appraisal of some shortest path algorithms. Operations Research 17, 345–412.

Evans, J.R., Minieka, E., 1992. Optimization Algorithms for Networks and Graphs. Marcel Dekker, New York.

Floyd, R.W., 1962. Algorithm 97: shortest path. Communications of the ACM 5, 345.

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

Garey, M.R., Johnson, D.S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York.

Handler, G.Y., Zang, I., 1980. A dual algorithm for the constrained shortest path problem. Networks 10, 293–310.

Johsch, H.C., 1966. The shortest route problem with constraints. Journal of Mathematical Analysis 14, 191–197.

Magnanti, T.L., Wong, R.T., 1984. Network design and transportation planning: models and algorithms. Transportation Science 18, 1–55.

Martins, E.Q.V., 1984. An algorithm for ranking paths that may contain cycles. European Journal of Operational Research 18, 123–130.

Moore, E.F., 1959. The shortest path through a maze. Proceedings of the International Symposium on the Theory of Switching. Harvard University Press, pp. 285–292.

Zhan, F.B., Noon, C.E., 1998. Shortest path algorithms on real road networks. Transportation Science 32, 65–73.