A Successive Shortest Path Algorithm For The Semi-Assignment Problem

by

S.O. Duffuaa
Department of Industrial and Operations Engineering
University of Michigan
Ann Arbor, MI 48109
and
M.S. AL-Ghassab
Saudi Petrochem Co.
Jubail, Saudi Arabia

Technical Report # 94-15
April 1994

A Successive Shortest Path Algorithm For The Semi-Assignment Problem

by

S.O. Duffuaa*
Department of Industrial and Operations Engineering
University of Michigan
Ann Arbor, MI 48109
E-mail: SALIH.ENGIN.UMICH.EDU

and

M.S. AL-Ghassab
Saudi Petrochem Co.
Jubail, Saudi Arabia

In this paper a successive shortest path algorithm has been developed for the semi-assignment problem. The algorithm can be viewed as a dual method. It maintains dual feasibility and complementary slackness and achieves primal feasibility by solving a sequence of auxiliary shortest path problems. The algorithm has a worst case bound $0(n^3)$, where n is the number of destinations in the semi-assignment problem. Through empirical testing on standard semi-assignment problem it has been shown that the implementation of the proposed algorithm dominates a specialized primal simplex code and an Out-of-Kilter implementation.

Subject classification: Networks/graph; Shortest path problem; algorithm for semi-assignment problem.

---

* On Sabbatical leave from King Fahd University of Petroleum and Minerals, Dhahran, 31261, Saudi Arabia

The semi-assignment problem deals with assigning optimally m tasks to n agents such that each task is assigned to exactly one agent. It is a bipartite network whose demand constraints are the same as those of the classical assignment problem and whose supply constraints are the same as those of the transportation problem. Semi-assignment problems arise in numerous applications of scheduling, project planning and man power planning and assignment [4]. In new car design at General Motors (GM) semi-assignment formulations are used to assign microprocessors to tasks or functions. Here the number of microprocessors are much less than the tasks in the car. The mathematical programming formulation of the semi-assignment problem is as follows:

Minimize
$$\sum_{(c_{ij})\in E} c_{ij}x_{ij} \tag{1}$$

Subject to
$$\sum_{(i,j)\in FS(i)} x_{ij} = b_i, i \in I = (1,2,3,\ldots,m) \tag{2}$$

$$\sum_{(i,j)\in RS(j)} x_{ij} = 1 \ , j \in J = (1,2,3,\ldots,n) \tag{3}$$

$$x_{ij} \geq 0 \text{ for all } i, j \tag{4}$$

where E denotes the set of arcs in the network, FS(i) is the forward star of node i, RS(j) is the backward star for node j, c_{ij} is the cost of assigning processor i to task j, b_i denotes the number of tasks a processor can handle, and x_{ij} = 1, implies processor i is assigned to task j. It is clear the model is feasible if and only if $\sum_i b_i = n$.

The semi-assignment problem can be viewed as a minimum cost flow problem and therefore algorithms for solving pure networks such as specializations of primal simplex, dual simplex, Out-of-Kilter and shortest augmenting path (successive shortest path methods) can be employed for solving it. Specialization of the simplex method and the use of efficient data structures has resulted in codes for the minimum cost flow network which are 100 times faster than solving network problems with general purpose linear programming codes. For software implementations for the minimum

cost flow problem see [2,10]. For a complete development of the network simplex and its computational complexity see Ford and Fulkerson [9], Cunningham [5], Kennington and Helgason [14], Papadimitriou and Steiglitz [16], Tarjan [18], Rockefellar [17] and Ahuja, Magnanti and Orlin [1].

A major problem the simplex method specialization for the assignment problem has, is the phenomenon of degenerate pivots. This lead Barr, Glover and Klingman [3] to develop the alternating basis algorithm (AB) for the assignment problem. Later Barr, Glover and Klingman [4] generalized the AB algorithm to the semi-assignment problem which is the first specialized algorithm designed for the semi-assignment problem. The AB algorithm is also discovered in [5].

The shortest augmenting path algorithms or otherwise known as successive shortest path algorithms (SSP) are basically dual methods. Specialization of SSP methods to the assignment problem are done by Tomizawa [19], Hung and Rom [12], Enquist [8], and Jonker and Volgenant [13]. Extensive theoretical studies of shortest path algorithms for solving minimum cost flow problems are in Papadimitriou and Steiglitz [16], Edmonds and Karp [7], Hu [11], and Ahuja, Magnanti and Orlin [1].

Recently Kennington and Wang [15] developed a shortest augmenting path algorithm for the semi-assignment problem which was inspired by work of Jonker and Volgenant [13]. Their algorithm is a generalization of the algorithm in [13] for the assignment problem and their computational results indicate that their algorithm is the best known. In this paper we develop an algorithm for the semi-assignment problem based on successive shortest path approach. Our algorithm generalizes the SSP algorithm developed by Enquist for the classical assignment problem. Empirical testing and comparisons with CAPNET (a specialized simplex codes developed by Barr, Glover and Klingman [3] and TRANS (a network code in the Statistical Analysis Software (SAS) package) have shown that the developed SSP algorithm is uniformly

better on all sizes of tested problem. The test problems are generated using NETGEN and are used previously in [4].

The rest of the paper is organized as follows: Section 2 describes the SSP algorithm for the semi-assignment and Section 3 presents the steps of the algorithm and its theoretical properties. Section 4 discusses implementation issues for the algorithm and Section 5 presents the comparative computational testing with other algorithms. Section 6 concludes the paper.

2. Description Of The SSP Algorithm For The Semi-Assignment Problem

Given a semi-assignment problem as defined in equations (1-4), we say $X = (x_{ij})$ is a tentative solution provided $\exists$ at least one $j \in J$ such that:

$$\sum_{(i,j)\in FS(i)} x_{ij} = b_i, \quad i \in I.$$

Since the semi-assignment network will remain fixed in the following discussion, we denote the semi-assignment problem equipped with a tentative solution X by (m,n,C,b,X).

We say that (m,n,C,b,X) is in the standard form if $c_{ij} \geq 0$ for $(i,j) \in$ E and $c_{ij} = 0$ if $x_{ij} \geq 0$.

In the starting procedure for SSP a tentative solution is defined as follows. First,

$$\hat{c}_i = \min_{(i,p)\in FS(i)} \{c_{ip}\} \tag{5}$$

must be determined for $i \in$ I. $x_{ij} = b_i$ for some j such that $c_{ij} = \hat{c}$. For this X, (m,n,C,b,X) may not be in the standard form. However, the forward start of i may be scaled by setting

$$c_{ip} \leftarrow c_{ip} - \hat{c}_i, \text{ for } (i,p) \in FS(i). \tag{6}$$

The resulting (m,n,C,b,X) is in the standard form.

For a given tentative solution X, we let

$$a_j = \sum_{(i,j)\in RS(j)} x_{ij} \tag{7}$$

The modified semi-assignment problem relative to (m,n,C,b,X) is defined as follows:

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

Subject to
$$\sum_{(i,j)\in FS(i)} x_{ij} = b_i, \quad i \in I \tag{9}$$

$$\sum_{(i,j)\in RS(i)} x_{ij} = a_j, \quad j \in I \tag{10}$$

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

We note that when (m,n,C,b,X) is in standard form, X provides an optimal solution to the modified semi-assignment problem in (8) relative to (m,n,C,b,X).

A destination node j is said to be abundant relative to X when $a_j > 1$. Likewise, j is said to be deficient relative to X when $a_j = 0$, and if $a_j = 1$, node j is said to be neutral. The goal is to transform all abundant and deficient nodes to neutral nodes. This will be accomplished via a successive shortest path methodology.

Suppose (m,n,C,b,X) is in the standard form and d some deficient node with respect to X. Then the shortest path problem relative to (m,n,C,b,X) and d is denoted SP(m,n,b,X,d) and is defined as follows: The nodes of SP(m,n,b,X,d) are arcs (i,j) of the semi-assignment problem with, $x_{ij} > 0$. Such shortest path node is denoted by $\left(i \rightarrow X_{i,j}\right)$ or $\left(i \rightarrow j\right)$. Introduce one more node $(m+1 \rightarrow d)$ for the shortest path network and make it consistent with previous notation by extending X so that x_{m+1}, $d = 1$. For SP(m,n,b,X,d) the root node is $(m+1 \rightarrow d)$, while a node $\left(i \rightarrow X_{i,j}\right)$ is abundant provided node j is abundant relative to X. An arc exist in the shortest path network from $\left(i \rightarrow X_{i,j}\right)$ to $\left(p \rightarrow X_{p,k}\right)$ in case $(p,j) \in E$ and its length c_{pj}, where $j = X_{i,j}$.

The steps of the SSP algorithm will be given below after defining the following notations:

R_i = node potential for the i-th origin node,

K_j = node potential for the j-th destination node,

D_i = distance of the i-th shortest path node from the root,

P_i = predecessor of the i-th shortest path node in the shortest path tree.

The use of R is to denote the mapping whose value at i is R_i. The use of K, D and P is similar. When a shortest path problem is solved, denote the first abundant node to be permanently labelled by v, and denote the distance from the root by L, and $S(v)$ the set of nodes on the path from the root node to v. Let $C^o = \left\{ c^o{}_{ij} : (i,j) \in E \right\}$ denote the costs of the origin (original) semi-assignment problem. Then the steps of the SSP are given below.

3. The Steps of the SSP Algorithm and It's Theoretical Properties

In this section the steps of the SSP algorithm are stated and clarified by an example. Then the theoretical properties of the algorithm are given.

0. Define X^1 and transform C^0 *to* C^1 by scaling as described in equation (6) above, so that (C^1, X^1) is in the standard form. Set $k = 1$.
1. Choose a deficient destination node d^k relative to X^k. If no deficient node exist, stop X^k defines an optimal solution.
2. Solve $SP(m,n,C^k,b,d^k)$. The shortest path algorithm is terminated as soon as an abundant node is permanently labelled. If the SSP fails to permanently label an abundant node, stop, the semi-assignment problem is infeasible. Otherwise, the results of this step are D^k, P^k, v^k and L^k.
3. For each permanently labelled shortest path node $(i \to j)$ from step 2, set $R_i^k = D_i^k - L^k$ and $K_j^k = L^k - D_i^k$. For any remaining origins i or destinations j, set $R_i^k = K_j^k = 0$.

4. Set $c_{ij}^{k+1} = c_{ij}^{k} - R_i^k - K_j^k$, for $(i,j) \in E$.

5. Whenever the i-th shortest path node is in $S(v^k)$, $i \neq m+1$, set $X_{i,0}^{k+1} = X_{l,Q}^{k}$, where O is the destination assignment number for node i, Q is the destination assignment number of node l, and $l = P_i^k$. For the abundant node $X_{i,r_i+1}^{k+1} = X_{l,Q}^{k}$, if $X_{i,0}^{k}$ receives more than one unit from node i (i.e., $x_{iX_{i,0}^k} > 1$). For all other origins, i, not in in $S(v^k)$, $X_{i,0}^{k+1} = X_{i,0}^{k}$. Set $k \rightarrow k + 1$ and go to step 1 of the SSP.

In the case of the semi-assignment problem in an intermediate stage we may have a source supplying several destinations. Let the number of destinations be r_i, e.g. $(i \rightarrow X_{i,1})$, $(i \rightarrow X_{i,2})$, ... ,$(i \rightarrow X_{i,r_i})$. This implies in the shortest path problem $SP(m,n,b,X^k,d^k)$ two nodes may have the same origin i, for example $(i \rightarrow X_{i,1})$ and $(i \rightarrow X_{i,2})$. Therefore in solving the $SP(m,n,C^k,b,X^k,d^k)$ a tie may occur when scanning the predecessor of these two nodes. The tie is broken arbitrarily. This does not occur in the case of the assignment problem, since each source supplies only one destination.

In order to fix ideas, we present one iteration of the SSP algorithm on the example showing how SP(m,n,C,X,d) is defined in a particular case. Let the semi-assignment problem be as shown in Figure 1, where arc (i,j) is labelled with cost c_{ij}.

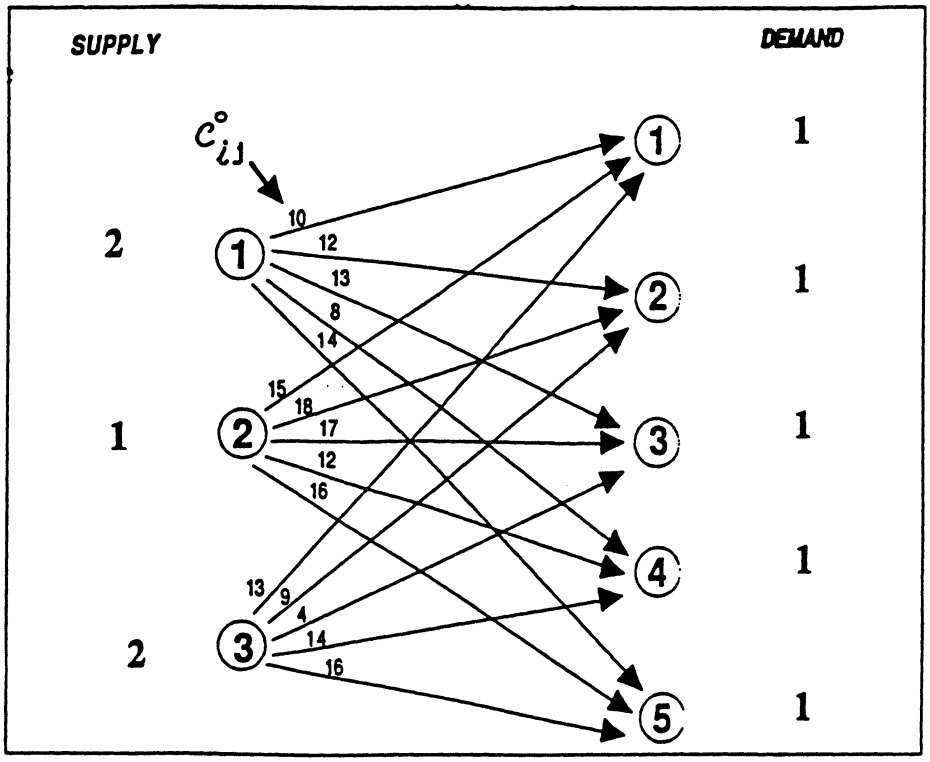

Figure I An Example of a Semi-assignment Problem.

Step 0

Put the problem into standard form:

$$\hat{c}_1 = 8, \quad \hat{c}_2 = 12, \quad \hat{c}_3 = 4$$

$$c^1_{11} = c^0_{11} - \hat{c}_1 = 10 - 8 = 2, \quad c^1_{21} = c^0_{21} - \hat{c}_2 = 15 - 12 = 3$$

$$c^1_{12} = c^0_{12} - \hat{c}_1 = 12 - 8 = 4, \quad c^1_{22} = c^0_{22} - \hat{c}_2 = 18 - 12 = 6$$

$$c^1_{13} = c^0_{13} - \hat{c}_1 = 13 - 8 = 5, \quad c^1_{23} = c^0_{23} - \hat{c}_2 = 17 - 12 = 5$$

$$c^1_{14} = c^0_{14} - \hat{c}_1 = 8 - 8 = 0, \quad c^1_{24} = c^0_{24} - \hat{c}_2 = 12 - 12 = 0$$

$$c^1_{15} = c^0_{15} - \hat{c}_1 = 14 - 8 = 6$$

$$c^1_{25} = c^0_{25} - \hat{c}_2 = 16 - 12 = 4$$

$$c^1_{31} = c^0_{31} - \hat{c}_3 = 13 - 4 = 9, \quad c^1_{32} = c^0_{32} - \hat{c}_3 = 19 - 4 = 5$$

$$c^1_{33} = c^0_{33} - \hat{c}_3 = 4 - 4 = 0, \quad c^1_{34} = c^0_{34} - \hat{c}_3 = 14 - 4 = 10$$

$$c^1_{35} = c^0_{35} - \hat{c}_3 = 16 - 4 = 12$$

Therefore,

$$X^1 = (X_{11}^1, X_{21}^1, X_{31}^1) = (4,4,3) \text{ and}$$

$$a_1^1 = 0, \quad a_2^1 = 0, \quad a_3^1 = 2, \quad a_4^1 = 3, \quad a_5^1 = 0$$

$$k = 1.$$

Iteration 1.

Step 1: Choose a deficient node $d^1 = 1$

Step 2: Solve SP(m,n,C^1,b,X^1,d^1). $m = 3$, $n = 5$.

The network for SP(m,n,C^1,b,X^1,d^1) is shown in Figure 2 where a shortest path node $\left(i \rightarrow X_{i,0}\right)$ is shown as a node with an upper label (i) and a lower label ($X_{i,0}$). The arcs of the shortest path network are labelled with their lengths.

The results of this step are:

$$D_1^1 = 2, \quad P_1^1 = 4, \quad v^1 = \begin{pmatrix} 1 \\ 4 \end{pmatrix} \text{ and } L^1 = 2$$

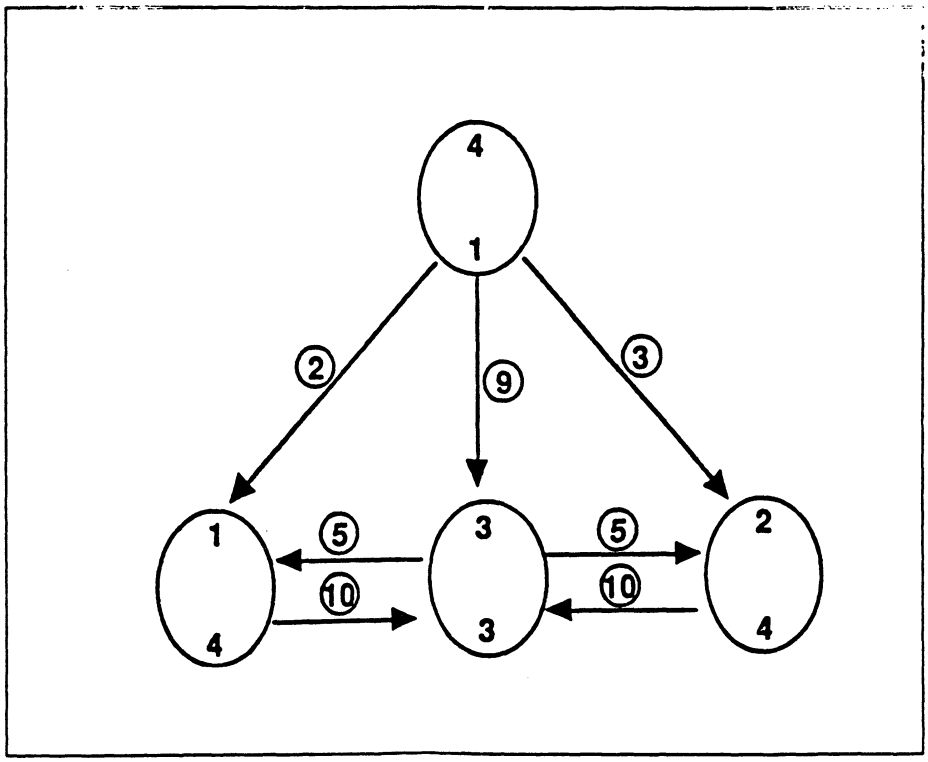

Figure 2 An Example of SP(C,X,d)

Step 3

$$R_1^1 = D_1^1 - L^1 = 2-2 = 0, \quad K_4^1 = L^1 - D_1^1 = 2-2 = 0$$

$$R_2^1 = R_3^1 = K_1^1 = K_2^1 = K_3^1 = K_5^1 = 0$$

Step 4

$$c_{11}^2 = c_{11}^1 - R_1^1 - K_1^1 = 2-0-0 = 2, \quad c_{13}^2 = c_{13}^1 - R_1^1 - K_3^1 = 5-0-0 = 5$$

$$c_{12}^2 = c_{12}^1 - R_1^1 - K_2^1 = 4-0-0 = 4, \quad c_{14}^2 = c_{14}^1 - R_1^1 - K_4^1 = 0-0-0 = 0$$

$$c_{15}^2 = c_{15}^1 - R_1^1 - K_5^1 = 6-0-0 = 6$$

$$c_{21}^2 = c_{21}^1 - R_2^1 - K_1^1 = 3-0-0 = 3, \quad c_{23}^2 = c_{23}^1 - R_2^1 - K_3^1 = 5-0-0 = 5$$

$$c_{22}^2 = c_{22}^1 - R_2^1 - K_1^1 = 6-0-0 = 6, \quad c_{24}^2 = c_{24}^1 - R_2^1 - K_4^1 = 0-0-0 = 0$$

$$c_{25}^2 = c_{25}^1 - R_2^1 - K_5^1 = 4-0-0 = 4$$

$$c_{31}^2 = c_{31}^1 - R_3^1 - K_1^1 = 9-0-0 = 9, \quad c_{33}^2 = c_{33}^1 - R_3^1 - K_3^1 = 0-0-0 = 0$$

$$c_{32}^2 = c_{32}^1 - R_3^1 - K_2^1 = 5-0-0 = 5, \quad c_{34}^2 = c_{34}^1 - R_3^1 - K_4^1 = 10-0-0 = 10$$

$$c_{35}^2 = c_{35}^1 - R_3^1 - K_5^1 = 12-0-0 = 12$$

Step 5

$$X^2 = \left(X_{11}^2, X_{12}^2, X_{21}^2, X_{31}^2\right) = (4,1,4,3)$$

$$a_1^2 = 1, \quad a_2^2 = 0, \quad a_3^2 = 2, \quad a_4^2 = 2, \quad a_5^2 = 0$$

let $k = 2$, go to step 1

Next, the theoretical Properties of the SSP Algorithm are given.

Theorem 1. *If SSP does not stop because of the infeasibility test in step 2, it reaches optimality in at most* n - 1 *iterations.*

Proof: We proceed by induction. We have (m,n,C^1,b,X^1) is in standard form with at most n - 1, deficient destinations. If we assume that (m,n,C^k,b,X^k) is in standard form with $q \geq 1$, deficient destinations, it follows that (m,n,C^{k+1},b,X^{k+1}) is in standard form with q - 1 deficient destinations. At each iteration the number of deficient nodes is reduced by one. Therefore the theorem follows.

Theorem 2. *SSP algorithm has a $O(n^3)$ computational bound.*

Proof: When the algorithm does not indicate an infeasible semi-assignment problem, it requires at most n - 1 iterations by Theorem 1. This result, together with the $O(n^2)$ computational bound, where n is the number of nodes in the shortest path tree, for the Dijkstra shortest path algorithm, implies the $O(n^3)$ bound in this case.

Theorem 3. *The original semi-assignment problem is infeasible if and only if the shortest path algorithm fails to label an abundant node of SP(m,n,C^k,b,d^k) on some iteration k.*

Proof: Half the proof follows from Theorem 1. We proceed with the remaining half. Suppose that the algorithm fails to label an abundant node. Let N_1 be the set of origins i such that $(i \rightarrow q)$ is permanently labelled for some q, and let N_2 be the set of destinations j such that $(p \rightarrow j)$ is permanently labelled for some p. All arcs of the original semi-assignment network which terminate in N_2 must originate in N_1 by the way SP(m,n,C^k,b,d^k) is defined. Since N_2 contains one more element than N_1 (namely, d^k) it is clear that the original semi-assignment problem is infeasible.

4. Implementation of the SSP Algorithm for the Semi-Assignment Problem

A Fortran code has been developed for the semi-assignment problem implementing the SSP algorithm developed in this paper and is called SPSN. Some of the details of the implementations are discussed below.

let,

$$R_i^o = \min_{(i,j)\in FS(i)} \{c_{ij}\},\ i \in I, \quad K_j^o = 0,\ j \in J$$

$$R_i^{-m} = \sum_{k=0}^{m-1} R_i^k,\ i \in I, \quad K_j^{-m} = \sum_{k=0}^{m-1} K_j^k,\ j \in J$$

R_i^{-m} and K_j^{-m} are the accumulated node potential for the modified semi-assignment problem relative to X^m.

There is a difference between the statement of the algorithm in section 3 and its implementation. In Step 3 of the algorithm node potentials are defined for all nodes of the semi-assignment network at each iteration, while in the code, the accumulated potentials are maintained. Thus at iteration k, only the node potentials corresponding to permanently labelled shortest path nodes need to be updated. In Step 4 the cost data for all arcs is updated, however, this is not the case in the code. Instead, whenever a cost c_{ij}^k is needed in the solution of SP(m,n,b,X^k,d), it is computed using the relation $c_{ij}^k = c_{ij}^o - R_i^{-k} - K_j^{-k}$. Next we describe how the details of the SSP algorithm is handled in the code.

In Step 1 of the SSP, there may be more than one deficient node to choose from. In this implementation the node with the smallest j such that j is deficient is chosen. Then the following technique is used for creating and solving the shortest path problem.

The shortest path problem is solved using Dijkstra algorithm. The implementation and data structures for Dijkstra is obtained from [6]. The efficiency of

Dijkstra algorithm depends on the maximum arc length in the problem which is the dimension of the array used for address calculation sort. Making a complete path through the arc data to determine the maximum shortest path arc length at each iteration of the SSP would be very inefficient. In the code we simply maintain an upper bound on the maximum shortest path arc length and use this in determining the length of the sort list or the size of the radix. If we let:

$$\bar{c} = \max_{(i,j)\in E}\left\{c_{ij}\right\}$$

and

$$\hat{K}^k = \min_{j\in J}\left\{K_j^k\right\}$$

Then the upper bound on arc length for SP(m,n,b,X^k,d^k) is $\bar{c} - \hat{K}^k$. This upper bound is easily updated along with node potentials K-k.

The assignment of each source node i to a destination node j is stored in an array called X (initially X:=Ø of length m. Each time a destination node j is satisfied from a source node i, it is linked to the list in position i of the array X.

When reversing the assignments in Step 5, a destination node in $S(v^k)$ may change its source node. If the abundant node in the modified semi-assignment problem receiving only one unit from a source node i in $S(v^k)$, then the position of this abundant node is occupied by the destination node of the predecessor of node i in $S(v^k)$. If the abundant node j in the modified semi-assignment problem receiving more than one unit from a source node i in $S(v^k)$, the destination of the predecessor of node i in $S(v^k)$ is supplied from node i and the abundant destination node j is supplied also from node i. For the rest of the path $S(v^k)$ continue by successively examining the predecessors of v until the root node is encountered.

5. Comparative Computation Tests

In this section the design and the result of the computational testing will be reported.

5.1 Design of Experiment

The developed code is in core and the program is written in Fortran IV and compiled using the same compiler and the same computer (AMDHAL 5850). The computer jobs were executed during periods when the machine load was approximately the same, and all solution times are exclusive of input and output. The total time spent solving the problem was recorded by calling a real time clock upon starting to solve the problem and again when the solution was obtained. Each test problem is solved five times and the average is reported. The problems used in the tests were randomly generated using NETGEN. In contrast to the assignment test problems, the specification of the semi-assignment test problems vary greatly in both the number of nodes and arcs. The test problems vary in size from 50 origins and 500 destinations to 400 origins and 4000 destinations. The number of arcs varies from 2000 arcs to 16,000 arcs. The cost data were randomly generated between 1 - 1000 for the first set of test problems. In the second set of test problems the cost data were randomly generated between 1 - 10,000.

5.2 Computational Results

The developed code SPSN is tested and compared with two other codes. The two other codes are CAPNET a specialized primal simplex code developed by Barr, Glover and Klingman at the University of Texas at Austin and the second code is TRANS an implementation of the Out-of-Kilter algorithm part of the Statistical Analysis Software (SAS).

In order to compare the three codes a set of semi-assignment problem generated by NETGEN and also used in the study by Barr et al. [4] were solved. The results of comparisons are shown in Tables 1 and 2. Table 1 shows problem size, density (in terms of number of arcs) and solution times (CPU times) for the set of problems with cost vector between 1 and 100. Table 2 presents the same items for the set of problems with cost vector in the range 1 - 10,000. Upon examining Tables 1 and 2 it is evident that SPSN dominates the other two codes and is the fastest code among the three. In terms of total CPU times for all problems solved SPSN is 1.3 times faster than CAPNET.

6. Conclusion

A new SSP algorithm has been developed for the semi-assiognment problem. The algorithm generalizes the SSP algorithm proposed by Enquist for the assignment problem [8]. The new algorithm is coded and compared with CAPNET, (a specialized primal simplex code) and TRANS an Out-of-Kilter code part of the SAS package. The empirical computational testing showed that the new algorithm dominates CAPNET and TRANS.

7 Acknowlegement

The authors would like to acknowlege the support provided by the department of systems Engineering, King Fhad University of Petroleum and Minerals, Dhahran,31261, Saudi Arabia, for conducting this research

Table 1. Solution Times in Seconds on Semi-Assignment Problems with a Cost Range of 1 - 1000.

No. of Nodes	No. of Arcs	Times (in seconds)		
m x *n*		CAPNET	TRANS	SPSN
50 x 500	2,000	1.247	CNR	0.6332
50 x 500	5,000	2.079	CNR	0.7456
50 x 500	10,000	3.740	CNR	1.568
50 x 1000	4,000	2.515	CNR	1.292
50 x 1000	10,000	5.384	CNR	2.904
50 x 1000	20,000	7.830	CNR	4.502
100 x 1000	4,000	3.064	CNR	1.742
100 x 1000	10,000	5.235	CNR	3.625
100 x 1000	16,000	7.930	CNR	4.370
400 x 4000	10,000	19.56	CNR	13.15
400 x 4000	16,000	25.13	CNR	17.95
Total Solution Times		83.714	—	52.4818

CNR = could not run as a result of memory limitations.

Table 2. Solution Times in Seconds on Semi-Assignment Problems with a Cost Range of 1 - 10,000.

No. of Nodes	No. of Arcs	Times (in seconds)	
m x *n*		CAPNET	SPSN
50 x 500	2,000	1.483	1.113
50 x 500	5,000	2.563	1.498
50 x 500	10,000	4.112	2.448
50 x 1000	4,000	2.752	2.059
50 x 1000	10,000	5.692	4.122
50 x 1000	20,000	8.042	6.751
100 x 1000	4,000	3.137	2.015
100 x 1000	10,000	5.652	6.074
100 x 1000	16,000	8.195	7.969
400 x 4000	10,000	21.30	21.14
400 x 4000	16,000	27.45	26.30
Total Solution Times		90.378	81.489

References

[1]. R.T. Ahuja, T.L. Magnanti and J.B. Orlin. Network Flows: Theory, Algorithms and Applications, Prentice Hall, (1993).

[2]. R.T. Ahuja, T.L. Magnanti and J.B. Orlin. Network Flows. In *Handbooks in Operations Research and Management Science, Volume I:* Optimization, G. Nemhauser, A. Rinnooy Kan and M. Todd (eds.). North-Holland, Amsterdam, (1989), 211-369.

[3]. R.F. Barr, Glover and D. Klingman. The Alternating Basis Algorithm for the Assignment Problems, *Mathematical Programming*, 13, (1977), 1-13.

[4]. R.F. Barr, Glover and D. Klingman. New Alternating Basis Algorithm for Semi-Assignment Networks. In Computers and Mathematical Programming, W. White (ed.) National Bureau of Standards Special Publication, U.S. Government Printing Office, Washington, D.C., (1978), 223-232.

[5]. W. Cunningham. A Network Simplex Method. *Math Prog,* 11, (1976), 105-116.

[6]. R.F. Dial, Glover, D. Karney and D. Klingman. A Computational Analysis of Alternative Algorithms and Labelling Techniques for Finding Shortest Path Trees. Networks 9, (1979), 215-248.

[7]. J. Edmonds and R. Krap. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *J. ACM*, 19, (1972), 248-264.

[8]. M. Enquist. A Successive Shortest Path Algorithm for the Assignment Problem, *IFOR*, 20(4), (1982), 370-384.

[9]. L. Ford and D. Fulkerson. Flows in Networks. Princeton. Princeton University Press, N.J. (1962).

[10]. F.D. Glover, Karney and D. Klingman. A Computational Comparisons of Primal, Dual and Primal-Dual Computer Codes for Network Flow Problems, Research Report CCS136, Center for Cybernetic Studies, University of Texas, Austin, Texas (1973).

[11]. T. Hu. Combinational Algorithms. Addison-Wesley, Reading, Mass (1982).

[12]. M. Hung and W. Rom. Solving the Assignment Problem by Relaxation, Operations Research, 28(4), (1980), 969-982.

[13]. R. Jonker and T. Volgenant. A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems, *Computing*, 38, (1987), 325-340.

[14]. J. Kennington and R. Helgason. Algorithms for Network Programming. John Wiley, New York, (1980).

[15]. J. Kennington and Z. Wang. A Shortest Augmenting Path Algorithm For The Semi-Assignment Problem Operations Research, 40(1), (1992), 178-187.

[16]. J. Papadimitriou and K. Steiglitz. Combinational Optimization: Algorithms and Complexity. Prentice-Hall, Englewood Cliffs, N.J. (1982).

[17]. R. Rockafellar. Network Flows and Monotropic Optimization, John Wiley, New York, (1984).

[18]. R. Tarjan. Data Structures and Network Algorithms. Society for Industrial and Applied Mathematics, Philadelphia, (1983).

[19]. N. Tomizawa. On Some Techniques Useful for the Solution of Transportation Network Problems, Networks 7, (1971), 173-194.