OPSEARCH, VOL. 40, No. 4, 2003

0030-3887/00 $ 5.00+0.00

New Algorithms For Multi Objective Shortest Path Problem

V. N. Sastry

Institute for Development and Research in Banking Technology (IDRBT)
Road No.1, Castle Hills, Masab Tank. Hyderabad-500 057. Andhra Pradesh, India.

T. N. Janakiraman

Department of Mathematics and Computer Applications
National Institute of Technology, Tiruchirappalli - 620 015. Tamil Nadu, India.

And

S. Ismail Mohideen

Post Graduate and Research Department of Mathematics
Jamal Mohamed College, Tiruchirappalli - 620 020. Tamil Nadu. India.

Abstract

In recent years there has been an increase in research activity on multi-objective network optimization problems. Network optimization models can be obtained from a large number of application domains such as transportation systems, communication systems, pipeline distribution systems, fluid flow systems and neural decision systems. The primary aim of these network models is to optimize the performance with respect to pre defined objectives. Multiple objectives such as optimization of cost, time, distance, delay, risk, reliability, quality of service and environment impact etc. may arise in such problems. Many real life applications, dealing with above networks, require the computation of best or shortest paths from one node to another, called Shortest Path Problem (SPP). In this paper, three new algorithms for Multiple Objective Shortest Path Problem (MOSPP) and an algorithm to detect negative cycle in a network are proposed. MOSPP in a cyclic and acyclic network having weights either positive or negative or both can be solved using the proposed algorithms. Maximum number of Pareto optimal paths of a MOSPP in a network, is very much useful in finding the maximum number of iterations and the complexity of a particular algorithm. We prove here, the maximum number of Pareto optimal paths of any MOSPP in a completely connected network, in the worst case, is $1+(n-2)+(n-2)(n-3)+\ldots+(n-2)!+(n-2)!$ and it lies between $2[(n-2)!]$ and

3[(n-2)!]. The computational complexities of the proposed algorithms have been analyzed. All proposed algorithms are illustrated with examples of cyclic and acyclic network.

Key words

Networks, multiple objectives, shortest path, weighted graph, Pareto minimum.

1. Introduction

Many real life problems can be represented as networks, such as transportation networks, communication networks, pipeline distribution networks, fluid flow (blood, plasma) networks and neural networks. The primary aim of these network models is to optimize the performance with respect to pre defined objectives. Multiple objectives such as optimization of cost, time, distance, delay, risk, reliability, quality of service and environment impact etc. may arise in such problems. Many applications, dealing with above networks, require the computation of best or shortest paths from one node to another, called Shortest Path Problem (SPP). When only one objective (criteria) is considered in the network, SPP is called a Single Objective Shortest Path Problem (SOSPP). SOSPP appears in many applications as a sub problem [3] for which several efficient algorithms are available in literature [12]. A Multiple Objective Shortest Path Problem (MOSPP) in a network consists of more than one objective.

In a single objective case, the best path indicates the optimal path (shortest or longest). This concept of optimality is replaced by Pareto optimality [8] in a multiple objective case, due to a conflicting nature of objectives, that is, the optimal solution according to one objective function can be different from that of another objective function. A Pareto Optimal Solution (POS), or non-dominated solution, for a multiple objective optimization problem is one for which no objective function can be improved without a simultaneous detriment to at least one of the other objectives. Therefore, there are several POSs. If we compute all POSs for a given network, a final solution can be selected from it, which suits best the real world problem requirement.

Many algorithms available in the literature fall under one of the two categories, namely Label Setting Method (LSM) [4] and Label Correcting Method (LCM) [6]. In LSMs, there are two types of nodes, permanently labeled nodes and temporarily labeled nodes. Once a node is permanently labeled (marked), its shortest distance from the source node is known. In every iteration, one node gets permanently labeled. In this method, once the label of a node is set as permanent, it will not change in future iterations. This method was first proposed by Dijkstra [4]. In LCMs, no node label is considered permanent, until the algorithm terminates. An LCM works, even if there are edges with negative weights, whereas an LSM works, only if the weights of the edges are non-negative [4]. An LCM was first proposed by Ford [6].

In the literature, SOSPP have been studied extensively. Bellman [1] gave one of the first algorithms, for determining shortest path from a source node to all other nodes of a network, with no restriction on the arc weights, using dynamic programming approach. The computational complexity of this algorithm is $O(n^3)$. Dijkstra [4] proposed an algorithm to solve SOSPP in a network, it works only when all the arc weights are non-negative. This is the most popular algorithm, for finding shortest path from a source node to all other nodes of a network, with complexity $O(n^2)$. For detailed survey of SOSPP, one can refer Deo and Pang [3], who gave a listing of many references. Dreyfus [5] briefly described some algorithms in his survey paper. The problem of detecting negative circuits in SOSPP of a network is discussed in Yen [11].

Hansen [7] defined a series of bi-criterion path problems in directed graph and examined their computational complexities. He proved that a family of problems exists, for which, any path between a given pair of nodes is a non-dominated path. Hence, any algorithm to solve an MOSPP is exponential in the worst-case analysis. That is, no polynomial time algorithm can guarantee the determination of all non-dominated solutions in polynomial time. Also he stated that no systematic study of bi-criterion path problems has yet been published. Martin and Santose [9] have analyzed the labelling algorithm for MOSPP and have modified Hansen's network [7]. They have showed that labeling algorithm can also determine an exponential number of dominated labels to compute a single optimal solution.

Corley and Moon [2] have given an algorithm to solve MOSPP using dynamic programming approach, which requires $O(mn^{2n-3} + mn^n)$ elementary operations, where m is the number of objectives and n is the number of nodes.

This motivates us to try for a better algorithm, which reduces the computational complexity of the existing algorithms. Four algorithms with their computational complexities are proposed in this paper. Algorithm 1 is proposed in section 3. It is a modification of Yen's [11] algorithm. Using this algorithm repeatedly, we can detect negative cycles, (if any) of an MOSPP in a network, in polynomial time. Maximum number of Pareto optimal paths of an MOSPP, in a network, is very much useful in finding the maximum number of iterations and the complexity of a particular algorithm. In section 4, we prove, the maximum number of Pareto optimal paths of any MOSPP in a completely connected network, in the worst case, is $1+(n-2)+(n-2)(n-3)+\ldots+(n-2)!+(n-2)!$ and it lies between $2[(n-2)!]$ and $3[(n-2)!]$. Also we develop, in section 4, a recurrence relation to find the maximum number of Pareto optimal paths of any MOSPP in the worst case as $p(n) = (n-2)\ p(n-1)+1$, $p(2) = 1$. Algorithm 2 is a modification of Dijkstra's algorithm and is a generalization to multiple objective cases, which is proposed in section 5. This algorithm finds the values of all Pareto minimum paths between any specified starting node s and all other nodes of an MOSPP in a network. Algorithm 3, proposed in section 6, is a generalization of Yen's algorithm [11]. This algorithm proceeds forwards and backwards alternatively, and, finds the values of all Pareto minimum paths from starting node 1 to all other nodes in atmost (n-1) iterations, where n is the number of nodes. A new algorithm is proposed in section 7, which requires less number of

iterations compared to algorithms 2 and 3. Computational complexities of all four algorithms are analyzed. Comparative analysis of the algorithms of an MOSPP and empirical observations are presented in section 8. Section 9 includes concluding remarks.

2. Notations, Assumptions And Terminology

2.1. Notations

- $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes or vertices of a network G. For the sake of convenience, it is represented as $\{1, 2, \ldots, n\}$.
- $s \in V$ is the starting node of a network. For convenience, s is taken as 1.
- $E \subseteq VxV$ is the set of edges or arcs of a network.
- $d_{ij} = (d_{ij}^1, d_{ij}^2, \ldots, d_{ij}^m)$ is the vector weight of the edge $(i, j) \in E$.
- F_i^k is the set of all values of the tentative Pareto minimum paths from the starting node s to all other nodes $i \in V$ at k^{th} iteration and is called value of the node i at k^{th} iteration.
- f_{ip}^k is the sum of the vector weights of edges lying on the tentative Pareto minimum path from starting node s to node i at k^{th} iteration and it is the p^{th} component of F_i^k.
- B_i is the set of all nodes other than the starting node s, which succeeds node i.
- Z_i is the set of all nodes other than starting node s, which precedes node i.
- N_i is the set of all nodes which precedes node i and less than i (i.e., $N_i \subset Z_i$).
- M_i is the set of all nodes which precedes node i and greater than i (i.e., $M_i \subset Z_i$).
- i and j are indices varying from 1 to n.
- k is the iteration number.
- S is the set of all paths between any two nodes of a network.
- $w(p) = (w^1(p), w^2(p), \ldots, w^m(p))$ is the value of a path $p \in S$.
- Pmin(•) is Pareto minimum of the set (•)

2.2. Assumptions

- E contains no self-loops, since a loop is not a part of shortest path.
- No restrictions on the signs of the components of d_{ij}.
- Remove all the edges of the network, whose head node is the starting node, as our aim is to find Pareto minimum paths from the starting node to all other nodes.
- Network may be cyclic or acyclic.

2.3. Terminology

2.3.1 Network: A network represented by G = (V, E, d) consists of the set V of n nodes (vertices), the set $E \subseteq V \times V$ of edges (arcs) and a function d defined as d: $E \rightarrow R^m$, that is, each edge (i, j) $\in$ E is associated with a vector weight $d_{ij} = (d_{ij}^1, d_{ij}^2, \ldots, d_{ij}^m)$. Here, m represents the number of objectives. If m = 1, the network is called single objective network and if m >1, it is called multi objective network. If each edge has direction, then it is called directed network.

2.3.2 Path: A path between the nodes i and j is a sequence of continuous edges that connects nodes i and j. We can also represent a path by the sequence $\langle v_i, \ldots, v_j \rangle$ of its nodes. If the sequence of nodes is finite then the path is said to be finite.

2.3.3 Value Of A Path: Value of a path is the sum of the weights of the edges constituting the path. Let S be the set of all paths between any two nodes. The value of a path p $\in$ S of a SOSPP is a function w: S $\rightarrow$ R such that $w(p) = \sum_{(i,j) \in p} d_{ij}$, where d_{ij} is the weight of the edge (i, j). The value of a path p $\in$ S of a MOSPP is a vector function w: S $\rightarrow R^m$ such that $w(p) = (w^1(p), w^2(p), \ldots, w^m(p))$ and each component w^r, r $\in$ {1, 2, ..., m} is defined as a function w^r: S $\rightarrow$ R. If p is a path $\langle v_1, v_2, \ldots, v_n \rangle$, then $w^r(p) = \sum_{i=1}^{n-1} d^r_{i, i+1}$, where r $\in$ {1, 2, ..., m}. That is, value of a path of a MOSPP is a vector, each of its component is the sum of the corresponding component of the vector weights of the edges constituting the path. A path is said to be bounded if and only if w(p) is finite.

2.3.4 Network Optimization Problem: Network optimization problem (NOP) is the problem of determining the optimum solution of a network with respect to the given optimization criteria. If the number of criteria to be optimized is one, then the problem is called Single Objective Network Optimization Problem (SONOP) and if it is more than one, then it is called Multiple Objective Network Optimization Problem (MONOP).

2.3.5 Order Relation On R^m: Let $a = (a^1, a^2, \ldots, a^m)$ and $b = (b^1, b^2, \ldots, b^m)$ be any two vector elements of R^m then $a <_{R^m} b$ if and only if $a^i \leq b^i$ for all i = 1, 2, ..., m and $a^i < b^i$ for at least one i.

2.3.6 Addition On R^m: Let $a = (a^1, a^2, \ldots, a^m)$ and $b = (b^1, b^2, \ldots, b^m)$ be any two vector elements of R^m then $a +_{R^m} b = (a^1 + b^1, a^2 + b^2, \ldots, a^m + b^m)$

2.3.7 Dominance ($<_D$) Of Paths : Let p and q be two paths of S. $p <_D q$ (p dominates q or q is dominated by p) if and only if $w(p) <_{R^m} w(q)$.

2.3.8 Non-Dominated or Pareto Minimum Path: For an MOSPP with S as the set of all paths between two specific nodes of a network, a path p* $\in$ S is said to be Pareto minimum if there does not exist any other path p $\in$ S such that $p <_D p^*$.

2.3.9 Cycle: A path which starts and ends at the same node is called a cycle.

2.3.10 Cyclic And Acyclic Network: If there is a cycle in a network, it is called cyclic network, otherwise it is called acyclic network.

2.3.11 Negative Cycle Of An Mospp: If at least one component of the sum of the vector weights along any cycle, of a multi objective network G, is negative then the cycle is called negative cycle or negative circuit. That is, let C be a cycle in G. C is a negative cycle of an MOSPP if and only if there exist $r \in \{1, 2, \ldots, m\}$ such that $w^r(C) < 0$.

2.3.12 Zero Cycle: C is a zero cycle in an MOSPP if and only if $w(C) = 0_{R^m}$. That is, all components of the sum of the vector weights along the cycle are zero.

2.3.13 Concatenation Of Two Paths: Concatenation of two paths can be defined, only if the terminal node of first path is the initial node of the second. Concatenation of two paths $p = \langle v_1, \ldots, v_i \rangle$ and $q = \langle v_i, \ldots, v_j \rangle$ is $p \oplus q = \langle v_1, \ldots, v_i, \ldots, v_j \rangle$.

If C is the cycle represented by $\langle v_1, \ldots, v_i, v_1 \rangle$ then $C^0 = \langle v_1 \rangle$ and $C^{j+1} = C^j \oplus C$, $j \geq 0$. That is $C^1 = \langle v_1, \ldots, v_i, v_1 \rangle$, $C^2 = \langle v_1, \ldots, v_i, v_1, \ldots, v_i, v_1 \rangle$.

2.3.14 Completely Connected Network: Completely connected network is a network, in which every ordered pair of nodes is having a directed edge.

2.4. Some Important Results

Theorems 1,2,3 are due to Martin and Santos [9], which are reproduced here, for the sake of continuity and completeness in establishing the proof of our proposed algorithms.

Theorem 1: An MOSPP is bounded, if and only if there is no negative cycle in G.

Theorem 2: An MOSPP is finite, if there is no negative or zero cycle in G.

Theorem 3: If there are no negative cycles in the MOSPP, then any non-dominated path is formed by non-dominated sub paths.

3. Detection of Negative Cycle of An MOSPP

From theorem 1 of section 2.4, if a network has a negative cycle, then the problem is unbounded. Corley and Moon's algorithm detects negative cycle, (if exists), only after an exponential number of elementary operations. Here, we propose a method to detect negative cycle of an MOSPP in polynomial time.

Let us assume that each edge of an MOSPP in a network is assigned with m number of objectives. Consider the objectives of any edge (i, j) as $(d_{ij}^1, d_{ij}^2, \ldots, d_{ij}^m)$, where i and j are tail and head nodes of the edge (i, j). If the value of the r^{th} component of atleast one edge (i, j), (say d_{ij}^r), is negative, then consider the value of the r^{th} component of each edge as the weight of the corresponding edge, and apply algorithm 1 of section 3.1 to detect a negative cycle in polynomial time. In the worst case, we have to apply algorithm 1 (SOSPP) m times. If there exists no negative cycle, then we can apply the proposed algorithms to solve MOSPP; otherwise we can indicate the existence of negative cycle in polynomial time.

3.1. Algorithm 1

Yen's [11] algorithm is modified here, to detect negative cycles, (if exist) of an SOSPP in a network. Here, F_i^k is the value of the shortest path from starting node 1 to all other nodes $i \in V$ at k^{th} iteration.

Step 1:

For i = 2, 3,..., n: let $d_{1i} = \infty$, if (1, i) is not an edge. Let $F_1^k = 0$, for all k. For i = 2, 3, ..., n : let $Z_i = \{x: x \in V-\{1\}$ and (x, i) is an edge$\}$, let $N_i = \{x: x \in Z_i$ and $x < i\}$, let $M_i = \{x: x \in Z_i$ and $x > i\}$, let $F_i^0 = \{d_{1i}\}$. Let k = 1.

Step 2:

For i = 2, 3, ..., n if $N_i = \{\}$, then $F_i^{2k-1} = F_i^{2k-2}$. Otherwise, compute

$$F_i^{2k-1} = \text{Min}\ \{ \bigcup_{j \in N_i} (F_j^{2k-1} + d_{ji}) \cup F_i^{2k-2} \}.$$

If k = 1, go to step 3. If $F_i^{2k-1} = F_i^{2k-2}$ for all i, then there is no negative cycle, terminate. If k = (n+1)/2, then negative cycle exists, terminate. Otherwise, go to step 3.

Step 3:

For i = n, n-1, ..., 3, 2 if $M_i = \{\}$, then $F_i^{2k} = F_i^{2k-1}$. Otherwise, compute

$$F_i^{2k} = \text{Min}\{ \bigcup_{j \in M_i} (F_j^{2k} + d_{ji}) \cup F_i^{2k-1} \}.$$

Step 4:

If $F_i^{2k} = F_i^{2k-1}$, for all i, then there is no negative cycle, terminate. If k = n/2, then negative cycle exists, terminate. Otherwise, let k = k+1 and go to step 2.

3.2 Computational Complexity

We consider only additions and comparisons as elementary operations for computing the complexities of the proposed algorithms, as the number of additions and comparisons dominates all other number of elementary operations. The worst-case analysis involves, in finding an instance of a specific size on which the algorithm is ill behaved. Consider a completely connected network with n nodes. This algorithm detects negative cycle, (if exists), only after n iterations, that is, 2k-1 = n or 2k = n (considering forward as one iteration and backward as another iteration). The computational complexity of this algorithm is $O(n^3)$.

4. Maximum Number of Pareto Optimal Paths

Maximum number of Pareto optimal paths of an MOSPP in a network is very much useful in finding the maximum number of iterations and the complexity of a particular algorithm. We prove, here, that the maximum number of Pareto optimal paths of any MOSPP in a completely connected network, in the worst case, is 1 + (n-2) + (n-2)(n-3) + ... + (n-2)! + (n-2)! and it lies between 2[(n-2)!] and 3[(n-2)!]. Also we develop a recurrence relation, to find the maximum number of Pareto optimal paths of any MOSPP in the worst case, as p(n) = (n-2) p(n-1)+1, p(2) = 1.

Theorem 4. The maximum number of Pareto minimum paths from node 1 to any other node t, of any MOSPP, in a completely connected network with out multiple edges and self-loops is between 2[(n-2)!] and 3[(n-2)!], where n is the number of nodes.

Proof:

Consider a completely connected directed network G = (V, E), where V = {1, 2, ..., n}is the set of n nodes and E $\subseteq$ V $\times$ V is the set of edges. Without loss of generality, we assume that every loop-less path from node 1 to any other node t is Pareto minimum. Therefore, to generate all Pareto minimum paths it is enough to generate all distinct loop-less paths form node 1 to node t. As our aim is to find all loop-less paths from initial node 1 to all other node t, all edges whose head node is 1 are removed from the network.

Paths from node 1 to node t with only one edge, that is, paths not passing through any intermediate node is ⟨1, t⟩. Hence, number of paths with only one edge is 1. Paths with two edges, that is, paths passing through only one intermediate node are ⟨1, 2, t⟩, ⟨1, 3, t⟩, ..., ⟨1, n, t⟩. Hence, number of paths with two edges, is nothing but permutation of (n-2) nodes (excluding 1 and t) taken one at a time, is $P_{n-2,\,1}$ = (n-2). Similarly, number of paths with three edges, (that is, number of paths passing through two intermediate nodes), which is nothing but permutation of (n-2) nodes taken 2 nodes at a time, is $P_{n-2,\,2}$ = (n-2)(n-3). Similarly, number of paths with (n-2) edges, is the permutation of (n-2) nodes taken (n-3) nodes at a time, is $P_{n-2,\,n-3}$ = (n-2)(n-3)...(3)(2) = (n-2)!. Lastly, the number of paths with (n-1) edges is the permutation of (n-2) nodes taken (n-2) nodes at a time, is $P_{n-2,\,n-2}$ = (n-2)(n-3)...(2)(1) = (n-2)!.

Therefore, total maximum number of distinct loop-less paths from node 1 to node t is
= $P_{n-2,\,0} + P_{n-2,\,1} + P_{n-2,\,2} + \ldots + P_{n-2,\,n-2}$

= 1 + (n-2) + (n-2)(n-3) + ... +(n-2)! +(n-2)!

= (n-2)!{1/(n-2)! + 1/(n-3)! + ... +1/3! +1/2! + 1 + 1}

= (n-2)!{1/(n-2)! + 1/(n-3)! + ... +1/3! +1/2! + 2}, where n ≥2.

The sum of the terms inside the bracket lies between 2 and 3. Hence, maximum number of Pareto minimum paths from node 1 to any other node t lies between 2[(n-2)!] and 3[(n-2)!].

Theorem 5. The above result in theorem 4 obeys the recurrence relation, p(n) =(n-2) p(n-1)+1 and p(2) = 1, where p(n) is the maximum number of distinct loop-less paths from initial node 1 to any other node t and n is the number of nodes of the completely connected network.

Proof: We can easily prove this theorem by induction.

NOTE: Using this result recursively, we can determine the maximum number of distinct loop-less paths from a specific starting node to any other node of a completely connected network.

5. Algorithm 2: Generalization of Modified Dijkstra's Algorithm

This algorithm computes the values of Pareto minimum (Pmin) paths from a specified starting node to all other nodes of a MOSPP. It is useful for both SOSPP and MOSPP of cyclic and acyclic networks having weights either positive or negative or both. To determine Pmin of step 4, the method of Sastry and Ismail Mohideen [10] is used.

Remarks:

1. Before applying algorithms of section 5, 6 and 7 to solve MOSPP, we have to ascertain the non-existence of negative cycle of the MOSPP, using the algorithm 1 of section 3.
2. All the three algorithms in section 5, 6 and 7 give the values of the Pareto minimum paths. The actual Pareto minimum path can be easily constructed by working backwards from the specified node such that we go to that predecessor whose Pareto minimum value differs exactly by the weight of the connecting edge. A tie indicates more than one Pareto minimum path.

Step 1:

Let $s \in V = \{1, 2, ..., n\}$ be the starting node. Let $F_s = \{(0, 0, ..., 0)\}$. Let $N = V = V - \{s\}$. For all $i \in V$, let $B_i = \{x: x \in V, x \neq s \text{ and } (i, x) \text{ is an arc}\}$. For all $i \in V$, if $(s, i) \in E$, let $F_i^0 = \{d_{si}\}$; else if $(s, i) \notin E$ let $F_i^0 = \{(\infty, \infty, ..., \infty)\}$. Let $k = 0$.

Step 2:

If $N = \{\}$, then Pareto minimum solution exist, terminate; else if $F_i^k \neq \{(\infty, \infty, ..., \infty)\}$ for at least one $i \in N$, then go to step 3; else if $F_i^k = \{(\infty, \infty, ..., \infty)\}$ for all $i \in N$ then there is no path from node s to node $i \in N$. Terminate.

Step 3:

Select a node y arbitrarily from N such that $F_y^k \neq \{(\infty, \infty, ..., \infty)\}$ and let $N=N-\{y\}$.

Step 4:

If $B_y = \{\}$, go to step 2; else for all $j \in B_y$, compute $F_j^{k+1} = \text{Pmin}\{F_y^k + d_{yj}, F_j^k\}$, where $F_y^k + d_{yj} = \{f_{yp}^k + R^m \quad d_{yj} : \quad f_{yp}^k \in F_y^k\}$. For all $j \in V - B_y$, let $F_j^{k+1} = F_j^k$.

Step 5:

If, for all $j \in B_y$, $F_j^{k+1} = F_j^k$, then $k = k+1$ and go to step 2; else put j in N for which $F_j^{k+1} \neq F_j^k$, let $k = k + 1$ and go to step 3.

Theorem 6. The solution obtained by algorithm 2 is Pareto optimal.

Proof:

As our aim is to find Pareto minimum paths from starting node to all other nodes, N is considered as the set of all nodes except the starting node s. In each iteration, one node is deleted from the set N; hence in (n-1) iteration all the nodes will be deleted, if there is no change in the existing values of the Pareto minimum paths at each iteration. Otherwise, it will be terminated in (n-1)(n-2) + 1 number of iterations at most as there is no negative cycle and any Pareto minimum path is a path without any cycle. In a completely connected network, the number of iterations depends on the number of times each node gets updated, which depends on the number of incoming edges to each node (excluding an edge from initial node as the path from starting node to all other nodes are initialized). Hence, for a network with n nodes, the number of incoming edges to each node is (n-2) and there are (n-1) nodes to be considered. Therefore, we need (n-1)(n-2) iterations to compute all the values of the Pareto minimum paths and one more iteration, to empty the stack of the set N. Hence the algorithm terminates in (n-1)(n-2)+1 iterations at most.

For all $i \in V$, F_i^0 will be $\{(\infty, \infty, ..., \infty)\}$, if there is no direct edge from starting node s to node i and $F_i^k \neq \{(\infty, \infty, ..., \infty)\}$, $(k \geq 1)$, if there is a path from starting node to node i (as each component of F_i^k is the sum of the vector weights of the edges lying on the tentative Pareto minimum paths from node s to node i at k^{th} iteration). Therefore, if $F_i^k = \{(\infty, \infty, ..., \infty)\}$ for all $i \in N$, the algorithm terminates indicating the non-existence of paths from starting node s to node i.

Now, we have to show that at the end of the algorithm F_i^{k+1} is the set of all vector sum of arc weights of Pareto minimum paths from starting node s to node i. As our aim is to find Pareto minimum paths from starting node s to all other nodes, $F_s = \{(0, 0, ..., 0)\}$. The vector weights of the edges whose tail node as starting node and head node as i are stored in the set F_i^0 for all $i \in V$ and if (s, i) is not an edge then $\{(\infty, \infty, ..., \infty)\}$ is stored in F_i^0. At $(k+1)^{th}$ iteration one of the nodes of N, say y, for which $F_y^k \neq \{(\infty, \infty, ..., \infty)\}$, is arbitrarily selected in Step 3, removed from N and its immediate successor nodes (which are stored in B_y) are updated in Step 4. If there is any change in the values of the nodes in B_y, compared to the corresponding values of the nodes at k^{th} iteration, those nodes are again stored in N. The change in the set F_i^{k+1} arises in two ways: (1) There may exist a new non-dominating path from node s to node i or (2) There may exist a new path from node s to node i, which dominates some or all the existing paths. In this way at the end of the algorithm, all paths from initial node to all other nodes are examined and the value of the Pareto minimum paths are stored in F_i^{k+1}.

It is important to remark that while the algorithm does not finish, we cannot guarantee that a path is non-dominated, that is, a non-dominated path can be dominated latter on. Therefore, only at the end of the algorithm we can determine all the non-dominated paths.

Theorem 6: Algorithm 2 terminates in (n-1)(n-2)+1 iterations at most, in an MOSPP of a completely connected network having n nodes.

Proof: We can easily prove this theorem by the method of induction.

Note: This number of iteration is sharp in the sense that for this algorithm this bound cannot be improved, which occurs in the worst case.

5.1 Computational Complexity of Algorithm 2

We consider only comparisons as elementary operations for computing the complexity of the proposed algorithm 2, as the number of comparisons dominates all other elementary operations. The worst case occurs when the network is completely connected, that is, every ordered pair of node is having an edge and every path is Pareto-minimum.

Total maximum number of elementary operations, in the worst case, is

$O((n-1)(n-2)^2((n-2)!)^2)$.

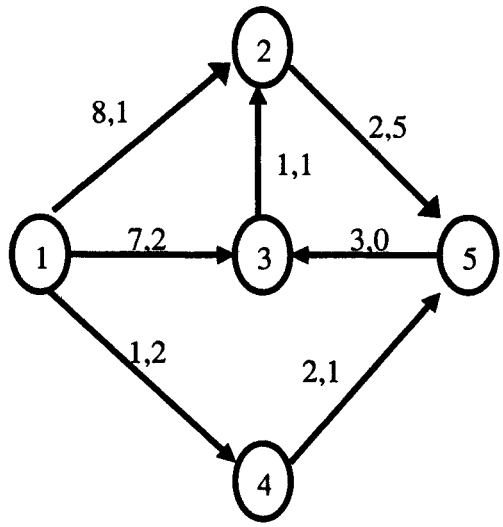
Remark:

Empirical observations of a completely connected network having n nodes

1. In each iteration, (n-2) nodes (excluding initial node and a node which is tentatively permanently marked) are updated.
2. In (n-1) iterations, all the (n-1) nodes (excluding the initial node) are updated (n-2) time each.
3. In $(n-2)^2 + 1$ iterations, all the paths from node 1 to all other nodes are generated. But the algorithm terminates only after $(n-1)(n-2) + 1$ iterations.

5.2. Numerical Example

Consider the following Corley and Moon's cyclic network with two objectives to be minimized.



Using algorithm 2 we compute the following:

Initialization: Let k = 0, starting node s = 1 and F_1= {(0,0)}. B_2 = {5}, B_3 = {2}, B_4 = {5}, B_5 = {3}. Let N = V = {2, 3, 4, 5}. F_2^0 ={d_{12}} = {(8,1)}, F_3^0= {d_{13}} = {(7, 2)}, F_4^0= {d_{14}} = {(1, 2)}, F_5^0 = {(∞, ∞)}.

Iteration 1: N = {2, 3, 4, 5}, $F_i^0 \neq$ {(∞, ∞)} for i = 2, 3 and 4. Let y = 2. N = {3, 4, 5}, B_2 = {5}. F_5^1 = PMin{F_2^0 + d_{25}; F_5^0}= PMin{(10, 6); (∞, ∞)}={(10, 6)} ≠ F_5^0. F_2^1 = F_2^0 = {(8,1)}, F_3^1 = F_3^0 = {(7,2)}, F_4^1 = F_4^0 = {(1, 2)}. Let k = 1.

Iteration 2: N = {3, 4, 5}, $F_i^1 \neq$ {(∞, ∞)} for i = 3, 4 and 5. Let y = 3. N = {4, 5}, B_3 ={2}. F_2^2 = PMin{F_3^1 + d_{32}; F_2^1}= PMin{(8, 3); (8, 1)}={(8,1)} = F_2^1. F_3^2 = F_3^1= {(7,2)}, F_4^2 = F_4^1 = {(1, 2)}, F_5^2 = F_5^1 = {(10, 6)}. Let k = 2.

Proceeding as in iteration 1 and 2 we get the following:

Iteration 3: N = {4, 5}, $F_i^2 \neq$ {(∞, ∞)} for i = 4 and 5. Let y = 4. N = {5}, B_4={5}

F_5^3 = {(3, 3)} ≠ F_5^2, F_2^3 = F_2^2 = {(8,1)}, F_3^3 = F_3^2 = {(7,2)}, F_4^3 = F_4^2 = {(1, 2)}.

Iteration 4: N = {5}, $F_5^3 \neq$ {(∞, ∞)}. Let y = 5. N = { }, B_5={3}. F_3^4 = {(6, 3), (7,2)} ≠ F_3^3, F_2^4 = F_2^3 = {(8,1)}, F_4^4 = F_4^3 = {(1, 2)}, F_5^4 = F_5^3 = {(3, 3)}.

Iteration 5: N = {3}, $F_3^4 \neq$ {(∞, ∞)}. Let y = 3. N = { }, B_3={2}. F_2^5 = {(7, 4), (8,1)} ≠ F_2^4, F_3^5 = F_3^4 = {(6, 3), (7,2)}, F_4^5 = F_4^4 = {(1, 2)}, F_5^5 = F_5^4 = {(3, 3)}.

Iteration 6: N = {2}, $F_2^5 \neq$ {(∞, ∞)}. Let y = 2. N = { }, B_2 = {5}. F_5^6 = {(3, 3)} = F_5^5, F_2^6 = F_2^5 = {(7, 4), (8,1)}, F_3^6 = F_3^5 = {(6, 3), (7,2)}, F_4^6 = F_4^5 = {(1, 2)}.

Iteration 7: N = { }, terminate.

Result: Now backtracking from nodes 2, 3, 4 and 5, we get the following Pareto minimum paths from node 1 to nodes 2, 3, 4 and 5: < 1, 2 > with the value of the path as (8, 1), < 1, 4, 5, 3, 2 > with the value of the path as (7, 4), < 1, 4, 5, 3 > with the value of the path as (6, 3), < 1, 3 > with the value of the path as (7, 2), < 1, 4 > with the value of the path as (1, 2) and < 1, 4, 5 > with the value of the path as (3, 3).

6. Algorithm 3: Generalization of Modified Yen's Algorithm

This algorithm proceeds forwards and backwards alternatively and determines the values of all the Pareto minimum paths from starting node 1 to all other nodes of a MOSPP, with no negative cycle. It is useful for both SOSPP and MOSPP of cyclic and acyclic networks having weights either positive or negative or both.

Step 1:

For i = 2, 3,..., n : let d_{1i} = (∞, ∞, ..., ∞), if (1, i) is not an edge. Let F_1^k = (0, 0, ...0) for all k. For i = 2, 3, ..., n : let Z_i = {x: x ∈ V-{1} and (x, i) is an edge}, let N_i = {x: x ∈ Z_i and x < i}, let M_i = {x: x ∈ Z_i and x > i}, let F_i^0 = {d_{1i}}. Let k = 1.

Step 2:

For i = 2, 3, ..., n : if $N_i = \{\}$, then $F_i^{2k-1} = F_i^{2k-2}$. Otherwise, compute

$F_i^{2k-1} = \text{Pmin}\{ \bigcup_{j \in N_i} (d_{ji} + F_j^{2k-1}) \cup F_i^{2k-2} \}$, where $d_{ji} + F_j^k = \{ d_{ji} +_{R^m} f_{jp}^k : f_{jp}^k \in F_j^k \}$.

If k = 1, then go to step 3. If $F_i^{2k-1} = F_i^{2k-2}$, for all i, then go to step 5. If k = n/2, then go to step 5, otherwise, go to step 3.

Step 3:

For i = n, n-1, ..., 3, 2 : if $M_i = \{\}$, then $F_i^{2k} = F_i^{2k-1}$. Otherwise compute

$F_i^{2k} = \text{Pmin}\{ \bigcup_{j \in M_i} (d_{ji} + F_j^{2k}) \cup F_i^{2k-1} \}$, where $d_{ji} + F_j^k = \{ d_{ji} +_{R^m} f_{jp}^k : f_{jp}^k \in F_j^k \}$.

Step 4:

If $F_i^{2k} = F_i^{2k-1}$, for all i, then go to step 5. If k = (n-1)/2, then go to step 5, otherwise, let k = k+1 and go to step 2.

Step 5:

Pareto optimal solution exist, terminate.

Theorem 7. The solution obtained by algorithm 3 is Pareto minimum.

Proof:

Here, total number of nodes is n and our aim is to determine all Pareto minimum paths from node 1 to node i, $2 \le i \le n$. In the worst case, let us assume that every path in the network from node 1 to node i is Pareto minimum. By theorem 4, total maximum number of paths from node 1 to node n, lies between 2(n-2)! and 3(n-2)!

Our algorithm starts initially with paths having one edge. At $(2k-1)^{th}$ iteration, node i is processed using all currently processed nodes 2, 3, ..., (i-1) and at the $(2k)^{th}$ iteration, node i is processed using all currently processed nodes n, (n-1), ..., (i+1). Hence, if the network G is acyclic and having topological ordering, (that is, if $(i, j) \in E$ implies $i < j$), then the algorithm determines all the values of the Pareto minimum paths in the first iteration itself. If G is acyclic, it is possible to sort the set of nodes topologically.

Let us suppose that the network is cyclic and let $< 1, v_1, v_2, \ldots, v_p, i >$ be a Pareto minimum path from node 1 to node i passing through p distinct nodes $v_1, v_2, \ldots, v_p$ of the original network G. The path can be divided in to t homogeneous blocks (Refer the figure given below), that is, the number naming the nodes in each block either form a strictly increasing or decreasing sequence [11].

$$\underset{\text{BLOCK 1}}{|1 < v_1 < v_2 < \ldots < v_{r1}|} \underset{\text{BLOCK 2}}{> v_{r1+1} > v_{r1+2} > \ldots > v_{r2}|} < \ldots \underset{\text{BLOCK t}}{| v_{t+1} \ldots i|}$$

For example < 1, 5, 8, 7, 4, 3, 9, 11, 12 > consists of three homogeneous blocks

$|1 < 5 < 8| > 7 > 4 > 3| < 9 < 11 < 12|$.

In the first homogeneous block, as $1< v_1 < v_2 < \ldots < v_{r1}$, the values of the Pareto minimal paths from node 1 to the nodes $v_1, v_2, \ldots, v_{r1}$ are uniquely determined in iteration 1 using step 2 of the algorithm. In the second homogeneous block, as the Pareto minimum values of the nodes $v_1, v_2, \ldots, v_{r1}$ are already determined in iteration 1 and as $v_{r1} > v_{r1+1} > v_{r1+2} >\ldots> v_{r2}$, the values of the Pareto minimum paths from node 1 to the nodes $v_{r1+1}, v_{r1+2}, \ldots, v_{r2}$ are uniquely determined in iteration 2 using step 2 of the algorithm.

As the number of homogeneous blocks is bounded by (n-1), the algorithm determines the values of all the Pareto minimum paths in at most (n-1) iterations. Therefore, the iterative procedure of the algorithm converges in finite number of steps and the solution thus obtained is Pareto minimum.

6.1. Computational Complexity of Algorithm 3

Here also, we consider only comparisons as elementary operations for computing the complexity of the proposed algorithm 3. The worst case occurs when the network is completely connected. The computational complexity of algorithm 3 is $O((n-2)((n-1)!)^2)$, where n is the number of nodes.

6.2. Numerical Example

Consider the Corley and Moon's example of section 5.2

Using algorithm 3 we compute the following:

Initialization: $Z_2 = \{3\}, Z_3 = \{5\}, Z_4 = \{\}, Z_5 = \{2, 4\}; N_2 = \{\} = N_3 = N_4, N_5 = \{2, 4\}$

$M_2 = \{3\}, M_3 = \{5\}, M_4 = \{\}\ M_5 = \{\}; F_1^k = \{0\}$, for all k.

$F_2^0 = \{(8, 1)\}, F_3^0 = \{(7, 2)\}, F_4^0 = \{(1, 2)\}, F_5^0 = \{(\infty, \infty)\}, k = 1.$

Iteration I (forward): As $N_2 = \{\}, F_2^1 = F_2^0 = \{(8, 1)\}$; As $N_3 = \{\}, F_3^1 = F_3^0 = \{(7, 2)\}$. As $N_4 = \{\}, F_4^1 = F_4^0 = \{(1, 2)\}$. As $N_5 = \{2, 4\}, F_5^1 = PMin\ \{F_2^1 + d_{25}; F_4^1 + d_{45}; F_5^0\} =$ Pmin $\{(10, 6); (3, 3); (\infty, \infty)\} = \{(3, 3)\}$.

Iteration II (backward): As $M_5 = \{\}, F_5^2 = F_5^1 = \{(3, 3)\}$; As $M_4 = \{\}, F_4^2 = F_4^1 = \{(1, 2)\}$ As $M_3 = \{5\}, F_3^2 = PMin\ \{F_5^2 + d_{53}; F_3^1\} = PMin\{(6, 3); (7, 2)\} = \{(6, 3), (7, 2)\}$

As $M_2 = \{3\}, F_2^2 = PMin\ \{F_3^2 + d_{32}; F_2^1\} = PMin\{(7, 4), (8, 3); (8, 1)\} = \{(7, 4), (8, 1)\}$

Iteration III (forward): Proceeding as in iteration I forward, we get.

$F_2^3 = \{(7, 4), (8, 1)\}; F_3^3 = \{(6, 3), (7, 2)\}; F_4^3 = \{(1, 2)\}; F_5^3 = \{(3, 3)\}$

As $F_i^3 = F_i^2$ for all i, terminate.

7. Algorithm 4: New Algorithm

In modified Dijkstra's algorithm, we select a node arbitrarily, it is tentatively permanently marked and all its succeeding nodes are updated. Yen's algorithm proceeds forwards and backwards alternatively. In forward process, node j is updated using all the

currently updated nodes i such that i < j, and in backward process, node j is updated using all the currently updated nodes i such that i > j. Here, in this algorithm, a node is arbitrarily selected and is updated using all its preceding nodes. Hence, in one iteration all the nodes are updated using all the currently updated preceding nodes. Here, the node numbers can be of any order and starting node can be any node, unlike modified Yen's algorithm, which requires node numbers of the form 1, 2, ..., n and starting node 1. This algorithm also determines the values of all the Pareto minimum paths from a specified starting node s to all other nodes of a MOSPP with no negative cycle. It is useful for both SOSPP and MOSPP of cyclic and acyclic networks having weights either positive or negative or both.

Step 1:

Let $V = \{1, 2, \ldots, n\}$ be the set of all nodes and s be the initial node. Let $V = N = V-\{s\}$. For all $i \in V$: let $d_{si} = (\infty, \infty, \ldots, \infty)$, if (s, i) is not an edge, let $Z_i = \{x: x \in V$ and (x, i) is an edge$\}$, let $F_i^0 = \{d_{si}\}$. Let $F_s^k = (0, 0, \ldots 0)$ for all k. Let $k = 1$, $A = \{\}$.

Step 2:

Select a node y arbitrarily from V. Let $V = V-\{y\}$. If $Z_y = \{\}$, then let $F_y^k = F_y^{k-1}$ and go to step 4, otherwise, go to step 3.

Step 3:

Compute $F_y^k = \text{Pmin}\{\bigcup_{\substack{j \in Z_y \\ j \in A}} (F_j^k + d_{jy}) \bigcup_{\substack{j \in Z_y \\ j \in V}} (F_j^{k-1} + d_{jy}) \cup F_y^{k-1}\}$, where

$F_j^k + d_{jy} = \{f_{jp}^k +_{R^m} d_{ji} : f_{jp}^k \in F_j^k\}$.

Step 4:

Put y in A. If $V = \{\}$, then go to step 5. Otherwise, go to step 2.

Step 5:

Let $V = N$. If $F_i^k = F_i^{k-1}$, for all $i \in V$, then Pareto minimum solution exists and terminate. Otherwise, go to step 6.

Step 6:

If $k = (n-2)$, then Pareto minimum solution exists, terminate. Otherwise, let $k = k+1$ and $A = \{\}$. Go to step 2.

Theorem 8. Algorithm 4 determines the values of all the Pareto minimum paths, in an MOSPP of a completely connected network having n nodes, in at most (n-2) iterations.

Proof:

Consider a completely connected network G with n nodes. By theorem 2 of section 2.4, MOSPP is finite, if there is no negative or zero cycle in G.

Assume that every loop-less path from initial node s to node i, ($i \in V-\{s\}$), is Pareto minimum path. Hence, it is enough to prove that the algorithm generates all the loop-less paths from node s to node i, ($i \in V-\{s\}$), in (n-2) iterations at most.

In theorem 4, we have proved that the maximum number of paths from node s to any other node, in a completely connected network, is $P_{n-2,0} + P_{n-2,1} + P_{n-2,2} + \ldots + P_{n-2,n-2}$ and it lies between 2[(n-2)!] and 3[(n-2)!], where $P_{n-2,r}$ means number of paths from node s to any other node passing through r, ($0 \leq r \leq n-2$), intermediate nodes.

Here, all the paths passing through no intermediate node are all initialized. Each node $i \in V-\{s\}$ is updated, using all its recently updated preceding node $j \in V-\{s, i\}$, (that is, an edge (j, i) is added respectively to all the set of existing paths from node s to node j, stored at node j). Hence, all loop-less paths from node s to node i passing through no intermediate node, all loop-less paths from node s to node i passing through one intermediate node and some of the paths from node s to node i passing through more than one intermediate nodes are generated and they are stored at node i. (For better readability, one can refer section 7.1).

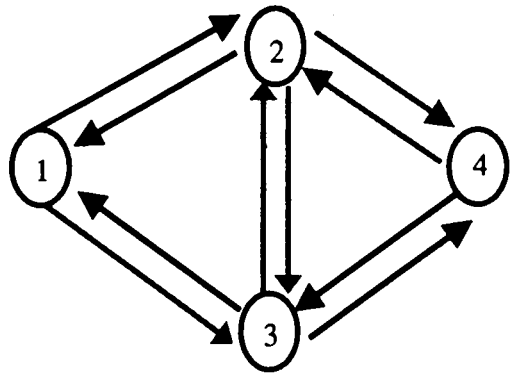
Let us assume that all the loop-less paths from node s to node $i \in V-\{s\}$ containing r, (r = 0, 1, 2, ..., k), intermediate nodes and some of the paths from node s to node i, containing more than k intermediate nodes are all generated at k^{th} iteration and are stored respectively at each node i.

At $(k+1)^{th}$ iteration, each node i is updated using all its recently updated nodes in such a way that: an edge (j, i) is added to all the (n-2) sets (as there are (n-2) preceding nodes, excluding the initial node and node i) of existing paths of the recently updated node j and all the loop-less paths are stored at node i.

Hence at $(k+1)^{th}$ iteration, all the loop-less paths from node s to node $i \in V-\{s\}$ containing r, (r = 0, 1, 2, ..., k+1), intermediate nodes are generated. As any loop-less path in an n node network contain maximum of (n-2) nodes, the algorithm terminates in (n-2) iterations at most.

7.1. Generation of All the Paths Using New Algorithm

New algorithm is illustrated with a completely connected network having 4 nodes:



Note: Algorithm determines only the value of the Pareto minimum paths, for explanation here we are using actual paths.

Initialization: $Z_2 = \{3, 4\}$, $Z_3 = \{2, 4\}$, $Z_4 = \{2, 3\}$. $F_2^0 = \{<1, 2>\}$, $F_3^0 = \{<1, 3>\}$, $F_4^0 = \{<1, 4>\}$; $k = 1$.

Iteration I: $F_2^1 = Pmin\{ F_3^0 + d_{32}, F_4^0 + d_{42}, F_2^0 \} = Pmin\{<1, 3, 2>, <1, 4, 2>, <1, 2>\}$.

All paths from node 1 to node 2, passing through no intermediate node and one intermediate node have been generated. $F_3^1 = Pmin\{F_2^1 + d_{23}, F_4^0 + d_{43}, F_3^0\} = Pmin\{<1,3,2,3>, <1,4,2,3>, <1,2,3>; <1,4,3>; <1,3>\} = \{<1, 4, 2, 3>, <1, 2, 3>; <1, 4, 3>; <1, 3>\}$.

All paths from node 1 to node 3, passing through no intermediate node and one intermediate node have been generated. Also, a path passing through two intermediate nodes has been generated.

$F_4^1 = Pmin\{ F_2^1 + d_{24}, F_3^1 + d_{43}, F_4^0 \} = Pmin\{<1,3,2,4>, <1,4,2,4>, <1,2,4>; <1,4,2,3,4>; <1,2,3,4>, <1,4,3,4>, <1,3,4>; <1,4>\} = \{<1,3,2,4>, <1,2,4>, <1,2,3,4>, <1,3,4>, <1,4>\}$. All paths from node 1 to node 4 have been generated.

For all i, $F_i^1 \neq F_i^0$. Also $k \neq 2$, let $k = 2$

Iteration II: Proceeding as in iteration I, we get

$F_2^2 = Pmin\{ F_3^1 + d_{32}, F_4^1 + d_{42}, F_2^1 \} = \{<1,4,3,2>, <1,3,2>, <1,3,4,2>, <1,4,2>, <1,2>\}$

$F_3^2 = Pmin\{ F_2^2 + d_{23}, F_4^1 + d_{43}, F_3^1 \} = \{<1,4,2,3>, <1,2,3>, <1,2,4,3>, <1,4,3>, <1,3>\}$

$F_4^2 = Pmin\{ F_2^2 + d_{24}, F_3^2 + d_{43}, F_4^1 \} = \{<1,3,2,4>, <1,2,4>, <1,2,3,4>, <1,3,4>, <1,4>\}$

For all i, $F_i^2 \neq F_i^1$. But $k = n-2 = 2$. Terminate. All the five Pareto minimum paths from node 1 to all other nodes have been generated.

7.2. Computational Complexity of New Algorithm

Here also, we consider only comparisons as elementary operations for computing the complexity of the proposed algorithm 4. The worst case occurs when the network is completely connected.

The total maximum number of elementary operations in the worst case is $O((n-1)(n-2)^2((n-2)!)^2)$, where n is the number of nodes.

7.3. Numerical Example

Consider the Corley and Moon's example of section 5.2

Using algorithm 4 we compute the following:

Initialization: $Z_2 = \{3\}$, $Z_3 = \{5\}$, $Z_4 = \{\}$, $Z_5 = \{2, 4\}$; $F_1^k = \{0, 0\}$, for all k.

$F_2^0 = \{(8, 1)\}$, $F_3^0 = \{(7, 2)\}$, $F_4^0 = \{(1, 2)\}$, $F_5^0 = \{(\infty, \infty)\}$, $k = 1$.

Iteration I: As $Z_2 = \{3\}$, $F_2^1 = PMin\{F_3^0 + d_{32}; F_2^0\} = PMin\{(8, 3); (8, 1)\} = \{(8, 1)\}$. As $Z_3 = \{5\}$, $F_3^1 = PMin\{F_5^0 + d_{53}; F_3^0\} = PMin\{(\infty, \infty), (7, 2)\} = \{(7, 2)\}$. As $Z_4 = \{\}$, $F_4^1 = F_4^0 = \{(1, 2)\}$. As $Z_5 = \{2, 4\}$, $F_5^1 = PMin\{F_2^1 + d_{25}; F_4^1 + d_{45}; F_5^0\} = PMin\{(10, 6); (3, 3); (\infty, \infty)\} = \{(3, 3)\}$.

Iteration II: Proceeding as in iteration I, we get:

$F_2^2 = \{(8, 1)\}$; $F_3^2 = \{(6, 3), (7, 2)\}$; $F_4^2 = \{(1, 2)\}$; $F_5^2 = \{(3, 3)\}$

Iteration III: Proceeding as in iteration I, we get:

$F_2^3 = \{(7, 4), (8, 1)\}$; $F_3^3 = \{(6, 3), (7, 2)\}$; $F_4^3 = \{(1, 2)\}$; $F_5^3 = \{(3, 3)\}$

As k = n-2 = 3, terminate.

8. Comparative Analysis

Comparing algorithms 2, 3 and 4, we get:

$9(n-2)[(n-2)!]^2 ((n-1)(n-2) + 1)$: $(9/2)(n-1)^2(n-2)((n-2)!)^2$: $9(n-1)(n-2)^2 ((n-2)!)^2$

Simplifying we get, $(n-1)(n-2) + 1$: $(n-1)^2/2$: $(n-1)(n-2)$

The following table shows, the worst-case complexities of different algorithms, for varying number of nodes n of the MOSPP. Since, we consider only comparisons as elementary operations for our proposed algorithms, the computational complexity of Corley and Moon's algorithm is $O(n^{2n-3})$, when considering comparison as elementary operation.

n	Modified Dijkstra's Algorithm (2)	Modified Yen's Algorithm (3)	New Algorithm (4)	Corley and Moon Algorithm
3	27	18	18	27
4	504	324	432	1024
5	12636	7776	11664	78125
6	435456	259200	414720	10077696
10	8.54469E+12	4.74055E+12	8.42764E+12	1E+17
20	2.27767E+36	1.1986E+36	2.27103E+36	1.37439E+48
50	1.5665E+128	7.9921E+127	1.5658E+128	6.3109E+164
60	9.8724E+162	5.0198E+162	9.8695E+162	1.1059E+208
70	1.7665E+199	8.9606E+198	1.7661E+199	6.0039E+252
80	5.5482E+236	2.8092E+236	5.5473E+236	6.0972E+298
90	2.1343E+275	1.0791E+275	2.134E+275	#NUM!

8.1 Empirical Observations

Following are the observations, when a completely connected network with 5 nodes is considered and algorithms 2, 3 and 4 are applied to generate all the loop-less paths.

1. Algorithm 2 requires 5528 comparisons in (n-1)(n-2)+1 = 13 iterations. But, it generates all the 16 loop-less paths in $(n-2)^2 + 1 = 10$ iterations with 3224 comparisons. Also our worst-case theoretical complexity, which is, total maximum number of comparisons in 13 iterations is 12,636, it is much higher than the exact value 5528.
2. Algorithm 3 requires 2585 comparisons to generate all the 16 loop-less paths in (n-1) = 4 iterations (forward 2 and backward 2). Here also, our worst-case theoretical complexity, which is, total maximum number of comparisons in (n-1) = 4 iterations is 7776, it is much higher than the exact value 2585.
3. Algorithm 4 requires 4191 comparisons to generate all the 16 loop-less paths in (n-2) = 3 iterations. Here also, our worst case theoretical complexity, which is, total maximum number of comparisons in (n-2) = 3 iterations is 11,664, it is much higher than the exact value 4191.

8.2. Remarks

Comparing algorithms 2 and 4, even though both behave alike, in the worst case, we observe that algorithm 2 behaves well in practical cases.

Among the three proposed algorithms, generalization of modified Yen's algorithm (algorithm 3) behaves well.

9. Concluding Remarks

1. The worst-case computational complexity to detect negative cycle of an SOSPP in a network with n nodes, by algorithm 1, is $O(n^3)$, where n is the number of nodes.
2. Using algorithm 1 repeatedly, we can detect negative cycles of an MOSPP in polynomial time, whereas the existing algorithm, like Corley and Moon, detects negative cycles only after exponential number of operations.
3. Hansen considered a particular network and proved that generation of all Pareto optimal solution to a two objective SPP in the worst case becomes intractable. But we have proved that the maximum number of Pareto optimal paths of any MOSOP, in the worst case, is 1 + (n-2) + (n-2)(n-3) + ... +(n-2)! +(n-2)! and it lies between 2(n-2)! and 3(n-2)!.
4. Algorithm 2 determines the values of the Pareto minimum paths from any specific starting node to all other nodes in at most (n-1)(n-2)+1 iterations with computational complexity $O((n-1)(n-2)^2((n-2)!)^2)$, where n is the number of nodes. Also it indicates the nonexistence of a path between a specific node and any other node, if any.
5. Using algorithm 2, we can also find Pareto minimum paths from all nodes to a specific node, by reversing the direction of each arc in the network.

6. Algorithm 3 computes values of all Pareto minimum paths from node 1 to all other nodes in (n-1) iterations at most with computational complexity $O((n-2)((n-1)!)^2)$, where n is the number of nodes.

7. Algorithm 4 in Section 7 computes values of all Pareto minimum paths from any specific starting node to all other nodes in atmost (n-2) iterations with computational complexity $O((n-1)(n-2)^2((n-2)!)^2)$, where n is the number of nodes.

8. If an MOSPP of a network is acyclic, then algorithm 3 and 4 determines all the values of the Pareto minimum paths in the first iteration itself.

9. All the three algorithms 2, 3 and 4 are useful for both SOSPP and MOSPP of cyclic and acyclic networks having weights either positive or negative or both.

10. The computational complexity of the proposed algorithms 2, 3 and 4 are less compared to the other existing algorithm, like Corley and Moon (The complexity of Corley and Moon's algorithm is $O(n^{2n-3})$, considering only comparison as elementary operations). (Refer section 8)

11. Even though algorithm 2 and 4 behaves theoretically alike, we observe that algorithm 2 requires less number of elementary operations compared to algorithm 4 in practical cases.

12. Among algorithms 2, 3 and 4 proposed here, algorithm 3 performs well.

13. The comparative study of the three proposed algorithms is confirmed by the time taken to solve a problem by the algorithms, that is, algorithm 3 outperformed well.

14. All the proposed algorithms determine the values of the Pareto minimum paths. As the number of values of the Pareto minimum paths always less than or equal to the number of Pareto minimum paths (because of the alternative paths), the amount of memory needed to store all the values of the Pareto minimum paths is less than or equal to that of all Pareto minimum paths. Hence, the algorithms proposed here are better than path generating algorithms.

11. Acknowledgment

The authors express their appreciation to Mr. K. Gopinath, M.C.A., Tiruchirappalli, for computer coding of the proposed algorithms. Authors are also grateful to the referees for their valuable comments that have improved the presentation of this research article.

12. References

1. Bellman, R.E., (1958), "On a Routing Problem", Quarterly Journal of Applied Mathematics, Vol.16, pp.87-90.

2. Corley, H.W. and Moon, I.D, (1985), "Technical Note: Shortest Path in Networks with Vector Weights", Journal of Optimization Theory and Applications, 46, 79-86.

3. Deo, N. and Pang, C., (1984), "Shortest Path Algorithms: Taxonomy and Annotations", Networks, 14, 275-323.

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

5. Dreyfus, S.E., (1969), "An appraisal of some shortest path algorithms", Operations Research, 17, 395-412.

6. Ford, L.R., (1956), "Network Flow Theory", The Rand Corporation, pp. 923.

7. Hansen, P., (1980), "Bi-criterion Path Problems", In Lecture Notes in Economics and Mathematical System 177 (Edited by M. Beckmann and H.P. Kunzi), 109-127, Springer, Berlin.

8. Hwang, C. and Masud, A.S.M., (1979), "Multiple Objective Decision Making – Methods and Applications", (Springer, Berlin).

9. Martins, E.Q.V. and Santos, J.L.E., (1999), "The labelling algorithm for the multi objective shortest path problem", CISUC, 1-24.

10. Sastry,V.N. and Ismail Mohideen,S., (1999), "A Modified Algorithm to Compute Pareto Optimal Vectors", Journal of Optimization Theory and Applications, Vol. 103, No.1, 241-244.

11. Yen, J.Y., (1970), "An algorithm for finding shortest routes from all source nodes to a given destination in general networks", Quarterly Journal of Applied Mathematics, 27, 526-530.

12. Zhan, F.B. and Noon, C.E., (1998), "Shortest Path Algorithm: An Evaluation using Real Road Networks", Transportation Science, Vol. 32, No.1, pp. 65-73.