Volume 24 (1), pp. 37–48
http://www.orssa.org.za

ORiON
ISSN 0529-191-X
©2008

A minimum incoming weight label method and its application in CPM networks

E Munapo[*] BC Jones[†] S Kumar[‡]

Received: 9 April 2007; Revised: 3 August 2007; Accepted: 23 October 2007

Abstract

An efficient approach towards finding a directed, shortest path or a directed longest path from the source to all other nodes in a directed network is described in this paper. Application of this approach with respect to CPM project networks is also considered. The approach is based on a minimum incoming weight labelling method and determines a critical path for a given project activity network. Although many algorithms exist in the operational research literature that may be used to find critical paths, the method discussed in this paper has an interesting application in determining optimal crash limits for various activities in a CPM network. In case of network topology changes due to any reason, such as that an activity has to be completed in crash duration or the actual duration of an activity takes longer than scheduled, a new critical path and new associated floats can be computed without analyzing the complete network all over again. This is achieved by recycling part of the available information. The algorithm presented in this paper is illustrated by means of numerical examples.

Key words: CPM network, activity crash limit, minimum weight label, shortest path, longest path, protean system, information recycling.

1 Introduction

Many an industrial problem may be modelled as a shortest path problem or one of its variants in a directed network [2, 3, 10–16]. For example, the conventional travelling salesman problem deals with a round trip visiting each place once before returning to some home city. Alternatively, specific attributes may be associated with each city and

[*]Department of Applied Mathematics, National University of Science and Technology, Bulawayo, Zimbabwe and also Department of Information and Communication Technology and Electronics, Faculty of Engineering Sciences, Chinhoyi University of Technology, Private Bag 7724, Chinhoyi, Zimbabwe.

[†]Department of Applied Mathematics, National University of Science and Technology, Bulawayo, Zimbabwe.

[‡]Corresponding author: Department of Mathematics and Statistics, University of Melbourne, Parkville, Victoria, Australia and also School of Computer Science and Mathematics, Victoria University, PO Box 14428, Footscray Park Campus, Melbourne, MC 8001, Australia, email: skumar@csm.vu.edu.au

one may be interested in a tour that covers a given set of attributes in the least amount of time or cost, as discussed in [3]. A large number of methods are available for finding shortest/longest paths in networks [1]. Project scheduling is one such case, where a project may be represented as a weighted, directed network with network node connections (arcs) denoting the various project activities. Arc directions represent activity precedence, while arc weights usually represent durations or some time attribute. Interest in such networks is centered on finding longest paths, which determine the set of critical activities and consequently the duration of the project [1, 4, 6, 18]. For these calculations, one typically finds associated floats on various activities and activity crash schedules so as to meet the target completion date. When activities have to be undertaken in crash duration, cost is also an important consideration. A major assumption in *C*ritical *P*ath *M*ethods (CPMs) is typically that the durations of all activities are precisely known. Similarly, in the *P*rogram *E*valuation *R*eview *T*echnique (PERT), an implied assumption is that the three-time estimates for each activity are precisely known. However, these assumptions may not be reasonable or realistic due to project interactions with external activities over which project managers have very little or no control. Consequently, changes in activity durations may prolong completion of the project, but may also change the network topology, thus requiring recalculation with respect to this altered information. Such changing networks are called *protean networks* [7, 8].

A new algorithm for finding shortest paths in networks is introduced in this paper. This algorithm also generates free floats associated with various activities. It is efficient with respect to the determination of the crash duration of any desired activity. Thus, when changes are experienced, the concept of information-recycling is applied. The new approach directly finds the free floats for each activity. The information-recycling concept is explained in §2, and it has been applied by one of the authors for linear, quadratic and geometric programming problems [8, 9], and for a reliability problem in [17]. This paper addresses, in a protean environment, the following:

1. A minimum weight label approach to find a shortest path.
2. Determining the set of critical activities, free floats associated with various activities and the critical paths.
3. Finding an optimum crash schedule of a given activity.

The minimum incoming weight label approach is explained in §3. Consideration of crash activities is presented in §4 and some computational experiments are presented in §5.

2 Information recycling in protean systems

The philosophy of information recycling is similar to recycling in waste management [7, 13, 15]. In waste management, motivation for recycling comes from reducing the bulk of solid waste and converting it to an alternative cheap resource for industry, thereby minimizing damage to the environment. The recycling of information is similarly intended to avoid unnecessary calculations in order to obtain required information from alternatively available information. The notion of recycling information is likely to be more useful for situations where events recur frequently, but with changed input data. For most real-life applications, models are not developed for one single use. Usually, once a model has been developed, the underlying situation may recur with different input variables. The question arises as to why one should carry out analyses each time considering the problem anew when an earlier solution may still provide useful information with respect to the current situation. One of the authors has classified such attempts as *information recycling* methods [8, 9] and systems that are subject to changes as *protean systems* [7, 17].

As examples of protean systems, consider telecommunication networks whose topologies change due to failures and link repairs, or road networks whose traffic flow topologies change due to accidents and routine maintenance. Similarly, CPM networks may also be regarded as protean networks due to changes in activity durations. For example, in duration estimates no provision is made for labour disputes or unexpected delays. When network topologies change, it is desirable to make use of the existing information in order to find the new values of the required unknowns. In this paper, we consider a protean project scheduling approach which employs recycling of information.

3 The minimum incoming weight label algorithm

Many methods are available for finding a shortest path [1, 5, 6, 18] between any given pair of nodes in a (directed) network. However, in this section, a new approach is introduced which is relatively simple to use and particularly suited for CPM networks under the recycling philosophy of a protean network. This new approach is called the *minimum incoming weight label algorithm.*

3.1 The general approach

Our general approach is to alter the weights associated with arcs entering a node by subtracting the minimum weight and by adding the same minimum weight to the weights of all arcs departing from that node, thus leaving the total weight along any path from the source to the sink passing through that intermediate node unaffected.

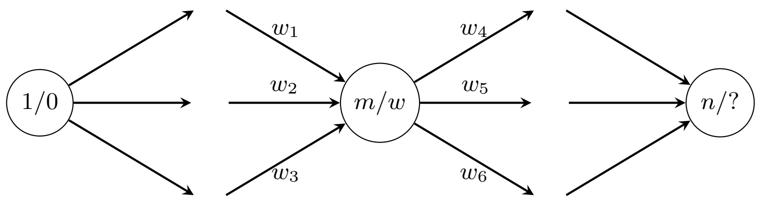

Figure 1: *Minimum incoming weight label method applied to a typical node m.*

For example, let w_1, w_2 and w_3 be the weights associated with the three incoming arcs to a node m and let the weights associated with three departing arcs be denoted by w_4, w_5 and w_6, as shown in Figure 1. Define the minimum weight $w = \min\{w_1, w_2, w_3\}$ and subtract this value from the associated weights of the minimum incoming arcs. Thus the new weights associated with these three incoming arcs are: $w_1 - w$, $w_2 - w$ and $w_3 - w$, respectively; at least one of these modified weights will be zero. The same minimum value, w, is added to the weights of all the departing arcs from this node. The modified weights associated with the three departing arcs from node m are therefore $w_4 + w, w_5 + w$ and $w_6 + w$, respectively. Note that the total weight along any path from the source to the sink, if it passes through the node m, will remain unchanged. Hence the quality of a given path through the node m with respect to its total distance from the source when compared to those of other paths, will remain unaffected. The node m is labelled with the label w. Note that a node can be labelled only if starting nodes of all arcs terminating at that node have already been labelled. To initiate this label algorithm, the source is assumed to have a zero label. The label on a node may be interpreted as the length of a shortest path from the source to that node; hence the label 0 at the source is meaningful.

3.2 Assumptions on the network structure

The CPM network of a project may be assumed to satisfy the following conditions:

1. Each activity (i, j) in the project network is a (directed) arc, the node i indicating the start of the activity and the node j indicating the end of the activity. The weight on activity (i, j) is its duration. The indices i and j satisfy the relation $i < j$, *i.e.* the nodes of the network are assumed to be numbered according to a *topological sort* of the nodes.
2. The nodes 1 and n represent the source (origin) and the sink (destination) respectively. The sink in the CPM also represents completion of the project.
3. All nodes of the network, other than the source and the sink, have at least one incoming and one outgoing activity associated with them. These intermediate nodes are numbered $2, 3, \ldots, n-1$.
4. Each node is assigned two values: a number m representing its sequential position in the project network according to the topological sort mentioned above in 1 and a label w representing the shortest distance from the source to that node.

Assumption 2 may be justified by the following known result, since CPM networks are acyclic.

Theorem 1 *An acyclic, directed network has a sink and a source.*

Proof: Let P be a longest directed path in the network, and suppose it is a u-v path. We claim that u is a source. If it were not, then there would be some arc (w, u). If w is not on P, then adding w to P gives a longer path than P. If w is on P, then a cycle is obtained. Either case leads to a contradiction. Hence u must be a source. By a similar argument v is a sink. ■

Assumption 1 is meaningful because of the known result below.

Theorem 2 *A directed network G has a topological sort if and only if it is acyclic.*

Proof: Suppose that G has a topological sort, but contains a cycle C. Let v_i be the node in C with the smallest subscript index. Since $v_i \in C$ there exists an arc into v_i from a another node $v_j \in C$. However, by the choice of v_i, $i < j$, leading to a contradiction.

The proof of the converse proceeds by induction. Let $P(n)$ be the statement that any directed, acyclic network with n vertices has a topological sort. Then $P(1)$ is trivially true. Suppose, as induction hypothesis, that $m \geq 1$ and that $P(m)$ is true. Let G be a directed, acyclic network with $m + 1$ vertices. Then G contains a sink v by Theorem 1. By means of the induction hypothesis, $G - v$ has a topological sort $v_1, v_2, \ldots, v_m$. Letting $v_{m+1} = v$ gives a topological sort of G, and we conclude that $P(m+1)$ is true by induction. ∎

For a network with n nodes a topological sort, whose existence is guaranteed by Theorem 2, may be found in $\mathcal{O}(n)$ time.

3.3 The algorithm

The steps of the minimum incoming weight label algorithm to determine a shortest path from the source to all other nodes is given below.

Step 1. Set $k \leftarrow 2$, label the source '0'.

Step 2. Find the minimum incoming weight w_k of node k, *i.e.* $w_k = \min\{w_{\ell_1,k}, w_{\ell_2,k}, \ldots, w_{\ell_k,k}\}$ where $w_{\ell_1,k}, w_{\ell_2,k}, \ldots, w_{\ell_k,k}$ are the weights associated with the ℓ_k incoming arcs to node k. Using the minimum weight, modify the weight associated with all the incoming activities to and outgoing activities from node k as explained above in §3.1 and record the weight w_k for node k.

Step 3. If $k < n - 1$, set $k \leftarrow k + 1$ and go to Step 2.

Step 4. All nodes have been labelled except for the sink, n. This means that the minimum weight w_n associated with the sink can be determined. Use this minimum weight w_n to modify weights of the incoming activities to the sink by subtracting this minimum from their weights. This minimum weight is recorded at the sink as the distance along a shortest path from the source to the sink.

Step 5. The actual shortest path may be traced by backtracking the activities through nodes with a modified weight of zero associated with them. Output the weight associated with node n as the duration of a shortest path and also all the corresponding shortest paths by means of backtracking.

If each node has only one activity with a zero modified weight activity entering that node, the shortest path from the source to that node is unique. Otherwise alternative paths exist, and may also be traced during the backtracking process. The algorithm described above clearly has a time complexity of $\mathcal{O}(n)$ for a network with n nodes.

3.4 CPM network application of the algorithm

Instead of shortest paths, one is interested in determining longest paths in a CPM network. This objective may be achieved by multiplying the given activity weights by -1 and applying the minimum incoming weight label algorithm. Note that all node labels will be negative quantities simply because minimum feasible weights at each node will be negative. However, modified weights associated with all activities will be nonnegative.

The labels and modified weights have the following interpretations:

1. The modulus value of the label on each node represents the length of a longest path from the source to that node.
2. The positive weights associated with various activities represent free floats, as we prove next.

Theorem 3 *The nonnegative weights associated with activities in a CPM network after applying the minimum weight label algorithm to the network in order to find longest paths represent free floats associated with those activities.*

Proof: The free float associated with a critical activity is 0 and the free float, $F(i,j)$, associated with a noncritical activity (i,j) is given by

$$F(i,j) = E_j - E_i - w_{i,j}, \tag{1}$$

where E_x is the earliest time by which node x may be reached from the source and where $w_{i,j}$ is duration of the activity (i,j).

The proof proceeds by showing that after executing the minimum incoming weight label algorithm, the nonnegative quantities associated with an activity (i,j) are equal to the right hand side quantity in (1). Now consider a typical node m having entering activities (N_1,m), $(N_2,m),\dots,(N_r,m)$ and let the weights associated with these activities be $-w_{N_1,m}, -w_{N_2,m},\dots,-w_{N_r,m}$. Also let the labels at the nodes from whence these activities emanate be $-E_{N_1}, -E_{N_2},\dots,-E_{N_r}$ respectively. The minimum incoming weight through node m is therefore

$$w = \min\{(-E_{N_1} - w_{N_1,m}), (-E_{N_2} - w_{N_2,m}),\dots,(-E_{N_r} - w_{N_r,m})\} = -E_m.$$

This minimum weight will be subtracted from the weights of all activities entering node m and it will be added to the weights of all outgoing activities from node m and this minimum weight will serve as the label for node m. After subtraction, the modified weights associated with the entering activities are

$$\begin{aligned}\text{Activity 1 weight} &= -E_{N_1} - w_{N_1,m} - (-E_m) = E_m - E_{N_1} - w_{N_1,m},\\ &\vdots\\ \text{Activity } r \text{ weight} &= -E_{N_r} - w_{N_r,m} - (-E_m) = E_m - E_{N_r} - w_{N_r,m}.\end{aligned} \tag{2}$$

Note that right hand sides of (1) and (2) are the same; hence the minimum weight label approach results in free floats associated with the activities. (The zero weights indicate the set of critical activities.) ■

3.5 A numerical example

Consider the small CPM network shown in Figure 2 and suppose longest paths are sought from the source to all other nodes. The upper triangular matrix in Table 1 represents the weight matrix of the network in the Figure 2.

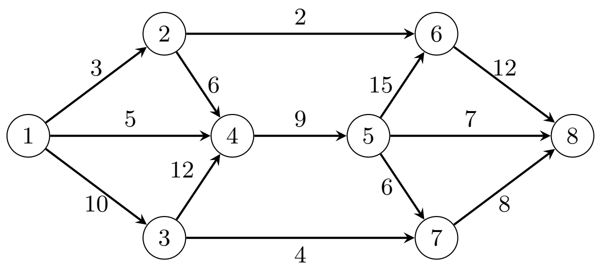

Figure 2: *A small CPM network.*

$i\backslash j$	2	3	4	5	6	7	8	label
1	−3	−10	−5	∗	∗	∗	∗	
2	—	∗	−6	∗	−2	∗	∗	
3		—	−12	∗	∗	−4	∗	
4			—	−9	∗	∗	∗	
5				—	−15	−6	−7	
6					—	∗	−12	
7						—	−8	

Table 1: *Tabular representation of the network in Figure 2 after every activity (i, j) duration has been multiplied by –1. A diagonal entry '—' indicates a zero distance from a node to itself. An asterisk means that no direct link exists between two nodes.*

Using the minimum weight label step through node 2, minimum weights associated with the incoming activity to node 2 is -3. Hence -3 is subtracted from the weight of activity $(1, 2)$, yielding $w_{1,2} = -3 - (-3) = 0$, and is added to the weight of activities $(2, 4)$ and $(2, 6)$, yielding $w_{2,4} = -6 + (-3) = -9$ and $w_{2,6} = -2 + (-3) = -5$. Next the minimum weight label algorithm is applied to node 3, and thereafter to all remaining nodes 4, 5, 6, 7 and 8. The resulting modified weights of all activities, after the minimum incoming weight label algorithm has been performed on the CPM network in Figure 2, are listed in Table 2.

$i\backslash j$	2	3	4	5	6	7	8	label
1	0	0	17	∗	∗	∗	∗	1/(0)
2	—	∗	13	∗	41	∗	∗	2/(−3)
3		—	0	∗	∗	23	∗	3/(−10)
4			—	0	∗	∗	∗	4/(−22)
5				—	0	0	20	5/(−31)
6					—	∗	0	6/(−46)
7						—	13	7/(−37)
min	− 3	−10	−22	−31	−46	−37	−58	8/(−58)

Table 2: *Modified activity weights after the minimum incoming weight label algorithm has been applied to the CPM network in Figure 2.*

The length of a longest path is 58 units and it may be traced by means of the zero entries in Table 2. The sink, 8, is approached from node 6, as the modified weight associated with activity (6,8) is zero. Similarly, node 6 is approached from node 5, and so forth. Thus the required longest path from the source to the sink is: $1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 8$. The critical activities are: $(1,3)$, $(3,4)$, $(4,5)$, $(5,6)$, $(6,8)$. If required, longest paths from the source to all other nodes may be traced out easily in similar fashion.

The positive values in Table 2 are free floats associated with those activities. For example, the free float associated with activity $(1,4)$ is 17, according to the table. It may easily be verified that the free float on activity $(1,4)$ is $E_4 - E_1 - w_{1,4} = 22 - 0 - 5 = 17$. Similar interpretations with regard to weights on all other activities hold.

4 Activity crash durations and compression limits

Once the set of critical activities and the corresponding critical paths in a CPM network are known, the project duration (the length of any longest path) is also known. If, due to external considerations, one is forced to reduce the project duration by an activity crash scheme, one has to gather the necessary information on activity crash durations and associated costs for that time reduction. Suppose a cost-time activity analysis suggests that an activity should be selected which seems promising in terms of reducing the overall project duration. At this stage the question of immediate interest is how to find the compression limit of the selected activity. The compression limit refers to reduction in duration of an activity at higher cost (typically by work undertaken outside the normal working hours). This aspect of the problem is discussed in this section.

Assume that activity (i,j) is the selected activity and that its duration is to be reduced by k units of time, where k is an unknown value to be determined. Recall that each activity duration was multiplied by -1; hence reduction in activity duration is achieved by adding a positive value k to the existing negative weight associated with this activity. The algorithmic procedure for finding the compression limit is as follows.

Step 1. In the final table, after applying the minimum incoming weight label algorithm, add k to the selected critical activity (i,j) and to all other activities that terminate at the same node j.

Step 2. Apply the minimum incoming weight label algorithm starting from node j. Note that all entering activities at node j will have non-negative weights associated with them (*i.e.* the weights are zero for critical activities and positive for non-critical activities). The new minimum at node j will therefore be a positive quantity k and the label at node j will also change to k more than its original value. Note that these new label values will still be non-positive. This means that the positive value k added to the existing negative labels, causes a decrease in the length of a longest path. This new minimum is added to all activities departing from node j.

Step 3. All the incoming activities to nodes m $(\geq j)$ have to satisfy the conditions $F(i,j) - k \geq 0$, where $F(i,j)$ is the free float calculated earlier.

Step 4. Let $k' = \min_{(i,j)}\{F(i,j) - k \geq 0 \mid k \leq \text{the compression limit of the selected activity}\}$. Here k' refers to the compression applied on the selected activity.

Step 5. Replace k by k' and update the network labels from node j onwards.

Reconsider the network shown in the Figure 2. Let the normal durations be the same as those considered earlier and suppose, in addition, that the crash durations are given in

Table 3. These crash durations refer to the shortest possible duration of a specific activity, when faced with a possible time/cost reduction. For example, it would be possible to complete activity $(3, 4)$ in 10 units of time instead of the normal 12 units of time, while activities $(5, 7)$ and $(6, 8)$ cannot be completed in a shorter duration of time.

Project activities	Normal duration as used in §3	Crash duration
(1,2)	3	2
(1,3)*	10	8
(1,4)	5	4
(2,4)	6	6
(2,6)	2	1
(3,4)*	12	10
(3,7)	4	3
(4,5)*	9	6
(5,6)*	15	14
(5,7)	6	6
(5,8)	7	6
(6,8)*	12	12
(7,8)	8	7

Table 3: *Data on the normal and crash durations of activities in the CPM nerwork in Figure 2. Asterisks indicate critical activities and the crash durations in the last column are assumed (given) values for illustration.*

Suppose activity $(4, 5)$ is the activity selected for crashing, based on a cost-time analysis. The objective is to find the compression limit for this activity. The consequences of crashing activity $(4, 5)$ are shown in Table 4.

$i\backslash j$	2	3	4	5	6	7	8	label
1	0	0	7	$*$	$*$	$*$	$*$	1/(0)
2	$-$	$*$	13	$*$	41	$*$	$*$	2/(−3)
3		$-$	0	$*$	$*$	23	$*$	3/(−10)
4			$-$	$0+k$	$*$	$*$	$*$	4/(−22)
5				$-$	$0+k$	$0+k$	$20+k$	5/(−31)
6					$-$	$*$	0	6/(−46)
7						$-$	13	7/(−37)
min	-3	-10	-22	$-31+k$	$-46+k$	$-37+k$	$-58+k$	8/(−58)

Table 4: *Values taken from Table 2, with the duration of activity* $(4, 5)$ *reduced by* k.

Note that no other activity terminates at node 5 (see column 5 in Table 4). The activities departing from node 5 are (5,6), (5,7) and (5,8). When the minimum incoming weight at node 5 is subtracted, the same minimum will be added to three activities. Thus, the labels of nodes 6, 7 and 8 will change. This will affect the free floats associated with (2,6) and (3,7), as shown in Table 5.

Finally, since all free floats have to be non-negative, we have the following conditions resulting from the affected activities:

1. Free float on activity (2,6): $41 - k \geq 0$.
2. Free float on activity (3,7): $23 - k \geq 0$.

$i\backslash j$	2	3	4	5	6	7	8	label	new label
1	0	0	17	$*$	$*$	$*$	$*$	1/(0)	1/(0)
2	–	$*$	13	$*$	$41-k$	$*$	$*$	2/(−3)	2/(−3)
3		–	0	$*$	$*$	$23-k$	$*$	3/(−10)	3/(−10)
4			–	$0+k$	$*$	$*$	$*$	4/(−22)	4/(−22)
5				–	$0+k$	$0+k$	$20+k$	$5/(-31+k)$	5/(−28)
6					–	$*$	0	$6/(-46+k)$	6/(−43)
7						–	13	$7/(-37+k)$	7/(−34)
min	− 3	−10	−22	$-31+k$	$-46+k$	$-37+k$	$-58+k$	$8/(-58+k)$	8/(−55)

Table 5: *Modified activity weights after the crash considerations of activity* $(4,5)$.

3. Crash consideration: The value of k for the selected activity $(4,5)$ is at most its normal duration less its crash duration, that is $k \leq 9-6=3$.

Therefore $k \leq \min\{3, 23, 41\} = 3$.

5 Computational experiments

Some experiments were conducted on randomly generated CPM networks and activity crash compression limits were calculated using the recycling aspect of the minimum incoming weight label algorithm described in §4. Eighteen classes of different network sizes were considered. For each class 20 networks were generated. The average behaviour of the minimum incoming weight algorithm was determined and the results are presented in Table 6. As a benchmark, a value of 100% denotes the computational effort required to perform computations all over again, as opposed to utilising the recycling method described in this paper. For the purpose of these calculations, the crash activities were also selected at random from among the critical activities.

It is clear from Table 6, that the minimum incoming weight algorithm introduces a significant reduction in computational effort when used to establish the compression limits for the critical activities in CPM networks. More specifically, one may expect to traverse/update around 50% fewer activities using this algorithm, on average.

Acknowledgements:

The authors gratefully acknowledge receiving valuable suggestions for improving the paper from the two anonymous reviewers.

References

[1] Ahuja RK, Magnanti TL & Orlin JB, 1993, *Network flows: Theory, algorithms and applications*, Prentice Hall, Upper Saddle River (NJ).

[2] Arora H & Kumar S, 1994, $(k+1)^{th}$ *Shortest path from k shortest paths in a network*, pp. 75–84 in Gupta OP (Ed), *Mathematics today*, **12-A,** PC Vaidya Sanman Nidhi Trust, Ahmedabad.

[3] Bansal SP & Kumar S, 1971, *Optimal tour with multiple job facilities at each station*, Indian Journal of Mathematics, **13(1)**, pp. 45–49.

Number of activities	Average number of activities used for compression limit	Reduction in computational effort
10	4	60.0%
20	11	45.0%
30	13	57.0%
40	19	52.5%
50	24	52.0%
60	32	47.0%
70	38	45.7%
80	45	43.7%
90	48	46.7%
100	51	49.0%
120	58	51.7%
140	69	50.7%
160	82	48.8%
200	105	47.5%
250	118	52.8%
300	161	46.4%
350	172	50.9%
400	196	51.0%

Table 6: *Computational experiments.*

[4] Bapoo R & Kumar S, 2003, *Analysis of a project scheduling protean network*, Zimbabwe Journal of Science and Technology, **3(1)**, pp. 32–39.

[5] Henley EJ & Williams RA, 1973, *Graph theory in modern engineering*, Academic Press, Inc., Orlando (FL).

[6] Hillier FS & Lieberman GJ, 1986, *Introduction to operations research*, Holden-Day, Inc., San Francisco (CA).

[7] Kumar S, 1995, *Optimization of protean systems: A review*, pp. 139–146, in Fushimi M & Tone K (Eds), *Proceedings of the Association of Asian Pacific Operational Research Societies 1994 Fukuoka, Japan,* World Scientific Publishers, Singapore.

[8] Kumar S, 2005, *Information recycling mathematical methods for protean systems: A path-way approach*, South African Journal of Industrial Engineering, **16(2)**, pp. 81–101.

[9] Kumar S, 2006, *Information recycling mathematical methods for protean systems: A path-way approach to a geometric program,* South African Journal of Industrial Engineering, **17(2)**, pp. 127–143.

[10] Kumar S & Arora H, 1995, *Shortest paths and connected graphs in a protean network*, pp 125–35, in Kapur PK & Seghal VK (Eds) *Operational research: Theory and practice*, Spaniel Publishers, New Delhi.

[11] Kumar S, Arora H, Chiera C & Raviganesh G, 1999, *Paths in self-healing protean network*, Padma Bhusan Professor BN Prasad Birth Centenary Commemorative Volume, Bulletin of the Allahabad Mathematical Society, **14**, pp. 79–94.

[12] Kumar S & Bansal SP, 1972, *On a set of N nodes network when each node has k sub-nodes*, Opsearch, **9(2)**, pp. 122–126.

[13] Kumar S & Bapoo R, 1999, *Deployment of the refuse vehicles in protean environment*, Opsearch, **36(3)**, pp. 242–259.

[14] Kumar S, Raviganesh G, McPhee J & Nyathi TM, 1999, *Terminal pair reliability in a protean network*, International Journal of Operations and Quantitative Management, **5(2)**, pp. 133–150.

[15] Oladimeji AA, Kumar S, Parikh CT & Bhala MJ, 2002, *Energy from waste: A perspective from developing countries*, pp 21–29 in Grover VI & Hogland W (Eds), *Recovering energy from the waste: Various aspects*, Science Publishers Inc., New Hampshire.

[16] Saksena JP & Kumar S, 1966, *Path through K specified nodes,* Operations Research, **14(4)**, pp. 909–913.

[17] Talukder HM & Kumar S, 2000, *Call routing in protean communication network: An application*, Zimbabwe Journal of Science and Technology, **1(2)**, pp. 79–90.

[18] Winston WL, 1995, *Introduction to mathematical programming: Applications and algorithms*, Duxbury Press, Wadsworth Publishing Company, Belmont (CA).