Delay Coerced Multi Constrained Quality of Service Routing Algorithm

PRAKASH P.S
Department of Computer Science and Engineering
AnnaUniversity of Technology, Coimbatore
Sri Ramakrishna Engineering College, Coimbatore 641 022
INDIA
prakashpsrajan@rediffmail.com

SELVAN.S
Principal
Anna University, Chennai 600 025
Alpha College of Engineering, Chennai 602 107
INDIA

Abstract: - IP networks are evolving from data communication infrastructure into many tight-constraint applications such as video conferencing, IP telephony and require stringent Quality-of-Service (QoS) requirements. A rudimentary issue in QoS routing is to find a path between source-destination pair that satisfies two or more end-to-end constraints. A difficulty in multi constrained routing is that it is intractable. In this context, a multi constrained QoS routing algorithm, Delay Coerced Multi Constrained Routing (DCMCR) is proposed. It approximates (K-1) constraints while coercing one of the constraints. DCMCR is (1+α) (K-1) approximation algorithm and it finds a feasible solution whose first path weight is bound by the first constraint and approximating remaining (K-1) constraints. The proposed algorithm is applied where one of the constraints is strictly satisfied and it performs well by choosing appropriate values of α and constraint bounds. The variety of experimental validations is carried out on different scenario to analyze the performance of the proposed scheme.

Key-Words: - QoS routing, coerce, multi constraint, approximation algorithm, Additive, Intractable.

1 Introduction

Determining a feasible path that satisfies a set of constraints such as delay, cost and reliability of path is a challenging issue in QoS routing [3,5,14] . A difficulty in multi constrained routing is that it is intractable [2,15,24]. QoS metrics are broadly divided into three different categories namely additive, multiplicative and concave metrics. Additive metric is the summation of all links constituting the path e.g. delay, hop-count, delay variance, cost [1], [25,18,20,12]. Multiplicative metric is the product of all links constituting the path. Reliability (1-loss rate) is an example of multiplicative metric. If an edge weight represents the reliability of the edge then the corresponding path weight is the product of the weight associated with the edges on the path. Since the logarithm of the product of N positive numbers is the sum of the logarithm of the N positive numbers. Hence QoS metric such as reliability is known as additive metric i.e. multiplicative metric can be converted into additive metric [17]. Another kind of QoS metric is known as concave (bottleneck) metric where the corresponding weight of a path is the smallest of the weight of the edges on the path e.g. bandwidth i.e. consider sub graphs with only those edges whose weights are greater than or equal to a particular chosen value. The concave metric of a path is the maximum or the minimum of the metric overall the links in the path. This metric is usually dealt with a preprocessing step called topology filtering, wherein all the links that do not satisfy the constraint are pruned and not considered further in the path selection process. The metrics considered should be orthogonal to each other so that there is no redundant information among the metrics [13].The QoS routing problem with a single metric can be

solved in polynomial time such as the widest path or least delay path, or least cost path problem etc. However, multi-constrained QoS routing problems where more than one additive parameter is involved, such as least delay and cost are intractable [11, 7, 16]. The MCP problem has been studied extensively. Most of the existing works concentrate on K=2 where K is the number of constraints, known as the delay constrained least cost problem where two edge weights are cost and delay and one seeks for a least cost path under the constraint that the delay of the path is within a given delay constraint. Ergun et al [5] presented an fully polynomial approximation scheme for the case of acyclic graphs. Chen and Nahrstdt [4] studied the decision version of the DCLC problem where the path that satisfies both the delay constraint and the cost constraint. Goel et al [6] presented an approximation algorithm for the single source all destination delay sensitive routing problem. The authors of [8, 23] proposed to use a linear combination of the two weights and presented simple algorithms for finding a good linear combination of the two weights. In this paper MCP problem with K≥2 is presented where the first constraint is enforced while other K-1 constraints are approximated.

In this paper, a multi constrained routing algorithm DCMCR is proposed which coerces one constraint while approximating (K-1) remaining constraints and it is (K-1) (1+α) approximation algorithm. This algorithm is found to be useful where one of the constraints is strictly satisfied. When the number of constraints is reduced to two this algorithm produces (1+α) approximation with better performance than that of many other algorithms designed for this purpose. The proposed algorithm is simulated through experiments on arbitrary and other network topologies and found that it outperforms other QoS routing schemes in the same suite. In this paper, the quality of service such as throughput and bandwidth is taken into account which is not considered in our earlier work. The effect of size of packet is also considered.

The rest of the paper is organized as follows. Section 2 describes DCMCR problem definition. The proposed algorithm is explained in section 3. Proof of the algorithm is described in section 4. Example calculation and experimental results are described in section 5 and 6 respectively. Finally section 7 concludes the paper.

2 DCMCR Problem Definition

A network is defined by an edge weighted graph G={V,E}, where V is the set of 'v' vertices and E is the set of 'e' edges. Each edge 'e' associated with K weights representing QoS constraints and $w_k(e)\geq 0$ is the K^{th} weight of edge 'e', $\forall e \in E$, $1\leq k\leq K$. In this problem, for each edge 'e' belongs to graph G, $w_1(e)$ is denotes the delay of e and $w_k(e)/W$, $2\leq k\leq K$ denotes the cost of 'e'. It is to seek the least cost s-d path in G with path delay no more than the delay constraint bound W_1. In this work, D(e) is used rather than $w_1(e)$ to denote the delay and C(e) rather then $w_k(e)/W$ to denote cost. Δ_d is used rather than W_1 to denote delay constraint bound and Δ_c rather than $w_k(e)/W$, $2\leq k\leq K$ to denote cost constraint bound. For a path 'p_{DMR}' in G, the K^{th} weight of path p_{DMR} is denoted by $w_k(p_{DMR})$, is the sum of K^{th} weights over the edges on p_{DMR} i.e. $w_k(p_{DMR}) = \sum_{e\in p_{DMR}} w_k(e)$.

DCMCR (G,s,d,Δ_d,Δ_c,D,C): An edge-weighted directed graph G=(V,E,D,C) where each edge $e \in E$ is associated with a delay D(e) and a cost C(e). Assume both delay and cost is non-negative real values. Δ_d is the delay constraint and Δ_c is the cost constraint for the source-destination pair. The objective is to find a path 'p_{DMR}' for a given s-d in G such that $C(p_{DMR)=} \sum_{e\in p_{DMR}} C(e)$ is minimized subject to the constraint $D(p_{DMR}) = \sum_{e\in p_{DMR}} D(e) \leq \Delta_d$. A source destination path 'p_{DMR}' is called delay constrained path if $D(p_{DMR}) \leq \Delta_d$. This algorithm searches for a least-cost delay constrained path and is denoted by p_{DMR}. In this algorithm, the first constraint is coerced $w_1(p_{DMR})\leq\Delta_d$ in the source-destination pair. DCMCR algorithm needs to solve the instance MCPv2.2.

MCPv2.2(G,s,d,K,Δ_d,Δ_c,C,D): A graph G = {V,E,D,C}with K edge positive integer valued edge weights D(e) and C(e) associated with each edge $e\in E$. A positive delay constraint bound Δ_d and a positive cost constraint bound Δ_c is defined. The objective is to find a source-destination path 'p' such that $D(p)\leq \Delta_d$ and $C(p)\leq\Delta_c$.

3 Algorithm Description

In this section, DCMCR algorithm for K≥2 QoS constraints is presented. The proposed algorithm is shown in Fig.1.

UB	Upper bound
LB	Lower bound
p_{DMR}	Feasible shortest path
p_{aux}	Auxiliary path
α	Approximation factor
c	Smallest constraint
T	Test value
v	Number of vertices
$w_k(p_{DMR})$	k^{th} weight of feasible path p_{DMR}
$w_{aux}(p_{aux})$	k^{th} auxiliary weight of auxiliary path p_{aux}
S	Set of vertices V
Q	Minimum priority queue
s-d	Source-destination pair
γ	Real number to construct auxiliary graph
G_{aux}	Auxiliary graph
(V,E)	Set of vertices and set of edges
(u,v)	Edge
K	Number of QoS constraints

len[ ][ ] (K-1) dimensional array to store the length of the edge.

pred[ ][ ] (K-1) dimensional array to store the predecessor.

Δ_d	Delay constraint
Δ_C	Cost constraint
C_{aux}	Auxiliary real value cost
C	Real value cost
D	Real value delay

```
1.  void main( ) {
2.      int c, C[], D[], C_aux[], V, S,E,Q, k, K,i=0;
3.      int LB[], UB[], len[][], pred[][];
4.      int α, a, T, Δ_d , Δ_C , γ;
5.      string p_DMR, p_aux, e, u, v, b, s,;
6.      find_smallest_constraint( ) {
/* initialize the set S of vertices and minimum priority queue Q */
7.          S=0;
8.          Q=V(G);
9.          while(Q!=0) {
10.             Q=V-S;
11.             c=Q;}
/* initialize a lower bound and upper bound such that any s-d path 'p_DMR' with D(p_DMR)<=Δ_d */
12.         D(p_DMR) =null; C(p_DMR)=null;
13.         for(e_k=1;e_k<=p_DMR;e_k++){
14.             D(p_DMR) = D(p_DMR) + D(e_k);
15.             C(p_DMR) = C(p_DMR) + C(e_k); }
16.         if(D(p_DMR)<=Δ_d && C(p_DMR)>= c) {
17.             LB[0]=c;
18.     UB[0]=c*v;}}//endfind_smallest_constraint()
/*construct an auxiliary graph G_aux which is an instance of graph G to find the optimal value x_opt of DCMCR */
19.     construct_auxgraph( ) {
20.         γ = (v-1)/LB*α;
21.         C_aux(e) = γ* C(e);}
22.         a = log(v);
23.         α = pow(a,2);
/* perform testing procedure to refine the value of lower bound LB and upper bound UB */
24.     while (UB[i] >=2(1+ α)LB[i]) {
25.         T=sqrt ((LB[i]*UB[i])/(1+α));
26.     if(D(p)<=Δ_d && C_aux(p)<= (v-1)/α) {
27.         String Test(T,α)= "yes"; }
28.         else { Test(T,α)= "no"; }
29.         if(Test(T,α) = = "yes") {
30.             UB[i+1] = T( 1+α);
31.             LB[i+1] = LB[i]; }
32.         else { UB[i+1]= UB[i];
33.             LB[i+1] = T; } }
34.     i++;
35.   UB= UB[i];
36.   LB = LB[i];} //end construct_auxgraph( )
37.     mcpv2.2( ) {
```

```
/*initialize (K-1) dimensional array to store length and predecessor of edges*/
38.                    K=3;
39.                    for(k=2;k<=K;k++) {
40.                          len[v,ck] = ∞;
41.                          pred[v,ck] = null;
42.                          len[s,ck] = 0;  }
                  /* update array values */
43.              for(k=2;k<=K;k++) {
44.                for( (u,v)=1;(u,v)<=E;(u,v)++) {
45.                if(len[v,ck]  > len[u,bk] +w1(u,v) ) {
46.              len[v,ck]  = len[u,bk] +w1(u,v);
47.                              pred[v,ck] = u;  }}}
/ * find the s-d path paux such that D(paux) < =  Δd and Caux(paux) <=c */
48.                    if(len[d,c] < = Δd ) {
49.                    pDMR =paux;
50.       printf("Feasible path is returned %s",pDMR);
51.                              break ( ); }
52.              else {
53.          printf("No feasible path found");
54.       break ( );}}// end mcpv2.2()
55.       }//end main
```

Fig.1. DCMCR Algorithm

Line 1-5: The pseudo code begins with the initialization process that the DCMCR algorithm initializes all parameters.

Line 6-11: DCMCR performs a function called *find_smallest_constraint ( )* that is to find smallest constraint 'c' for graph G. To accomplish this operation DCMCR algorithm uses a conventional shortest path algorithm.

Line 12-18: DCMCR algorithm initializes the lower and upper bounds to compute the optimal value x_{opt}. To accomplish this DCMCR finds the delay and cost of each edge along the path p_{DMR}. Then compare the delay of path $D(p_{DMR})$ with delay constraint Δ_d and cost of the path $C(p_{DMR})$ with the smallest constraint 'c'. If the delay and cost satisfies the condition then DCMCR initializes the lower and upper bounds to compute the optimal value x_{opt}. DCMCR uses the following equation to compute lower and upper bounds such that LB= c and UB = c*v where 'v' is the number of vertices in graph G. LB and UB denote the sequence of lower and upper bound pair. The value of x_{opt} lies between lower and upper bound values i.e. $LB \leq x_{opt} \leq UB$.

Line 19-23: DCMCR uses *construct auxgraph ( )* to construct an auxiliary graph $G_{aux=}(V,E,D,C_{aux})$, which is an instance of graph G=(V,E,D,C). The auxiliary graph is same as G except that the edge cost weighting function C is changed to C_{aux} such that $C_{aux}(e) = [C(e)*\gamma]$ for every $e \in E$.

Line 24-36: DCMCR refines the upper and lower bound values until the condition $UB \leq 2(1+\alpha)LB$ satisfies as explained in *while* loop. DCMCR uses the testing procedure to refine the lower and upper bound values. In testing procedure, DCMCR uses a positive real number γ. An auxiliary graph $G_{aux=}(V,E,D,C_{aux})$, an instance of graph G=(V,E,D,C), which is same as G except that the edge cost weighting function C is changed to C_{aux}, such that $C_{aux}(e) = C(e)*\gamma$ for every $e \in E$. For given real numbers $T > 0, \alpha > 0$, TEST(T,α) = "yes", if $MCPv2.2(G_{aux},s,d,\Delta_d,(v-1/\alpha),D,C_{aux})$ has feasible solution where $\gamma = (v-1/T.W.\alpha)$ and define TEST(T,α) = "no", if MCPv2.2 $(G_{aux},s,d,\Delta_d,(v-1/\alpha),D,C_{aux})$ has infeasible solution. It can be proved that TEST(T,α) = "yes" implies that $x_{opt} < T(1+\alpha)$ and TEST(T,α) = "no" implies $x_{opt} > T$. Let x_{opt} is the optimal value of DCMCR. Using testing procedure DCMCR algorithm refines the lower and upper bound values such that if TEST(T,α) = "yes" then $UB = T(1+\alpha)$ and if TEST(T,α) = "no" then LB=T. If $UB \leq 2(1+\alpha)LB$ then DCMCR decide the lower bound and upper bound values for further calculations.

Line 37-47: DCMCR applies *mcpv2.2( )*, an instance of DCMCR to compute an source-destination feasible path for auxiliary graph. *mcpv2.2( )* initializes the (K-1) dimensional arrays such as len[ ][ ] and pred[ ][ ] to store the length and predecessor of each edge along the auxiliary graph. DCMCR uses the edge relaxation method to update the length and predecessor of each edge. Thus, it updates the length of entire source-destination pair. Here, MCPv2.2 checks whether or not the length of edge 'v' is greater than the summation of path length of edge 'u' and edge weight of edge (u,v). If this condition satisfies, it updates each edge along the path.

Line 48-54: DCMCR checks whether the length of s-d path len[d,ck] is less than or equal to auxiliary delay constraint bound Δ_d such that $len[d,c_k] \leq \Delta_d$. If the length is less then MCPv2.2 returns the path p_{aux} is feasible path for DCMCR, otherwise not.

4 Proof of MCPv3 Algorithm

It is to be proved that the path p_{aux} found by MCPv2.2 algorithm is guaranteed to be (1+ α) approximation of DCMCR.This algorithm, for any given value $\alpha > 0$ returns a path 'p_{DMR}' for a source-destination pair s-d that is an (1+α) approximation of DCMCR (G, s, d, K, Δ_d, D, C). Find bottleneck edge cost 'c' such that an s-d path p_{DMR}, with $D(p_{DMR}) \le \Delta_d$ and $C(e) \le c$ for all $e \in p_{DMR}$. Then any source-destination path 'p', $D(p) \le \Delta_d$ must contain at least one edge 'e' $C(e) \ge c$.It is known that $c \le x_{opt} \le c.v$, where 'v' is the number of vertices and 'c' is the smallest constraint.

$LB[0] \le x_{opt} \le UB[0] \le v.LB[0]$ (Since LB=c and UB=c.v) (1)

Let p_{DMR} denote optimal solution of DCMCR(G,s,d,K,Δ_d,D,C) that is p_{DMR} is an s-d path such that,

$$D(p_{DMR}) \le \Delta_d,\ C(p_{DMR}) \le x_{opt} \quad (2)$$

Construct an auxiliary graph G_{aux}, having an auxiliary edge weight, $C_{aux}(e)=[C(e).\gamma]$. Let $\gamma = \frac{(v-1)}{LB \cdot \alpha}$.

Here, construct an auxiliary graph G_{aux}, an instance of graph G to find the optimal value x_{opt} of algorithm DCMCR. To find a x_{opt}, a testing procedure defined in this algorithm is performed. Hence,

$$C_{aux}(p_{DMR}) = \sum_{e \in p_{DMR}} [(C(e) \cdot \gamma) + 1]$$

$$\le \sum_{e \in p_{DMR}} [(C(e) \cdot \gamma) + (v-1)]$$ [p_{DMR} has at most (v-1) edges].

$$\le (v-1) + \gamma \sum_{e \in p_{DMR}} C(e)$$

$$\le (v-1) + \gamma C(p_{DMR}) \quad (3)$$

$$\le (v-1) + C(p_{DMR}) \frac{(v-1)}{LB \cdot \alpha}$$

Substitute the value of $C(p_{DMR})$ from Equation (2) ,

$$C_{aux}(p_{DMR}) = (v-1) + x_{opt} \cdot \frac{(v-1)}{LB \cdot \alpha}$$

Substitute the value of x_{opt} from Equation (1),

$$C_{aux}(p_{DMR}) = (v-1) + \frac{UB \cdot (v-1)}{LB \cdot \alpha} \quad (4)$$

The value of $C_{aux}(p_{DMR})$ is scaled down to nearest lower integer. From Equation (4), the path p_{DMR} is feasible solution of MCPv2.2 $(G_{aux}, s, d, \Delta_d, \left\lfloor \frac{UB \cdot (v-1)}{LB \cdot \alpha} \right\rfloor + (v-1), D, C_{aux})$

Therefore, find a path p_{aux} (from auxiliary graph) and this path p_{aux} may be different from p_{DMR}. If p_{aux} is feasible, it is guaranteed to return a feasible path in DCMCR.

Now, it is to prove that path p_{aux} found by algorithm MCPv2.2 is guaranteed to be an (1+α)-approximation of DCMCR. Since p_{aux} is computed by the algorithm,

$$D(p_{aux}) \le \Delta_d \text{ and}$$

$$C_{aux}(p_{aux}) \le \left\lfloor \frac{UB \cdot (v-1)}{LB \cdot \alpha} \right\rfloor + (v-1) \quad (5)$$

and $C_{aux}(p_{aux}) \le C_{aux}(p_{DMR})$ (6)

It is known that, $C_{aux}(e) = \lfloor C(e) \cdot \gamma \rfloor$

$C_{aux}(p_{aux}) = \lfloor C(p_{aux}) \cdot \gamma \rfloor$,

$$C(p_{aux}) = \frac{1}{\gamma} \cdot C_{aux}(p_{aux})$$

From Equation (5),

$$C(p_{aux}) = \frac{1}{\gamma} \cdot C_{aux}(p_{DMR})$$

From Equation (3),

$$C(p_{aux}) = \frac{1}{\gamma} \cdot [\gamma \cdot C(p_{DMR}) + (v-1)]$$

[since $C_{aux}(p_{DMR}) \le (v\text{-}1) + \gamma.C(p_{DMR})$]

It is known that $C(p_{DMR}) \le x_{opt}$, therefore,

$$C(p_{aux}) = \frac{1}{\gamma} \cdot [\gamma \cdot x_{opt} + (v-1)]$$

$$\le x_{opt} + \frac{(v-1)}{\gamma}$$

$$\le x_{opt} + \frac{(v-1)}{\frac{(v-1)}{LB \cdot \alpha}}, \quad [\because \gamma = \tfrac{(v-1)}{LB \cdot \alpha}].$$

$$\leq x_{opt} + LB \cdot \alpha$$

$$\leq x_{opt} + x_{opt} \cdot \alpha, \qquad [\because LB \leq x_{opt}]$$

$$C(p_{aux}) \leq x_{opt}(1+\alpha) \qquad (7)$$

Equation (7) shows that p_{aux} is an $(1+\alpha)$ approximation of DCMCR.This proves that p_{aux} is an $(1+\alpha)$ approximation to DCMCR. Here delay is coerced and remaining constraints are approximated when $k \geq 2$.

5 Example Calculation

A feasible path P_{DMR} such that $C(P_{DMR}) = \sum_{e \in P_{DMR}} c(e)$ is minimized subject to the constraint $D(P_{DMR}) = \sum_{e \in P_{DMR}} D(e) \leq \Delta_d$ where Δ_d=10, $D(e)=w_1(e)$ and $C(e)= \max\left(\frac{w_k(e)}{W}\right)$, $2 \leq k \leq K$.

Arbitrary topology shown in Fig. 2 is considered to compute the feasible paths.

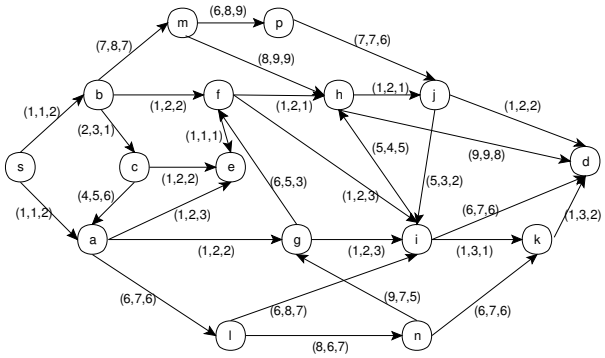

Fig.2. Arbitrary Topology

For each edge D(e) and C(e) are to be found as follows:

D(s,b)	=	1	D(k,d)	=	1
D(b,f)	=	1	D(b,c)	=	2
D(f,h)	=	1	D(c,a)	=	4
D(h,j)	=	1	D(c,e)	=	1
D(j,d)	=	1	D(a,e)	=	1
D(s,a)	=	1	D(f,e)	=	1
D(a,g)	=	1	D(g,f)	=	6
D(g,i)	=	1	D(f,i)	=	1
D(i,k)	=	1	D(i,h)	=	5
D(j,i)	=	5	D(b,m)	=	7
D(m,p)	=	6	D(m,h)	=	8
D(p,j)	=	7	D(a,l)	=	6
D(l,n)	=	8	D(l,i)	=	6
D(n,k)	=	6	D(n,g)	=	9
D(i,d)	=	6	D(h,d)	=	9

Delay of Each path is to be calculated as shown below:

1)	D(sbfhjd)	=	5
2)	D(sbfikd)	=	5
3)	D(sbfihjd)	=	10
4)	D(sbfhjikd)	=	11
5)	D(sagikd)	=	5
6)	D(sagihjd)	=	10
7)	D(sagfikd)	=	11
8)	D(sagfhjd)	=	11
9)	D(sagfhjikd)	=	17
10)	D(sagfihjd)	=	16
11)	D(sbcagikd)	=	11
12)	D(sbcagihjd)	=	16
13)	D(sbcagfikd)	=	17
14)	D(sbcagfhjd)	=	17
15)	D(sbcagfihjd)	=	22
16)	D(sbcagfhjikd)	=	23
17)	D(sbmpjd)	=	22
18)	D(sbmhjd)	=	18
19)	D(sbmhd)	=	25
20)	D(sagid)	=	9
21)	D(salnkd)	=	22
22)	D(salikd)	=	15
23)	D(salid)	=	19

For each edge find $C(e)= \max\left(\frac{w_k(e)}{W}\right)$, $2 \leq k \leq K$

The cost of each edge is as follows:

C(s,b)= max(1/22, 2/20)= max(0.05, 0.1)=0.1
C(b,f)= max(2/22, 2/20)=max(0.09,0.1)=0.1
C(f,h) = max(,2/22,1/20)= max(0.09,0.05)=0.09
C(h,j) = max(2/22,1/20) = max(0.09,0.05)=0.09
C(j,d)= max(2/22,2/20)= max(0.09,0.1)= 0.1
C(s,a) = max(1/22,2/20) = max(0.045,0.1)=0.1
C(a,g)= max(2/22,2/20)= max(0.09,0.1)= 0.1

C(g,i) = max(2/22,3/20)=max(0.09,0.15)=0.15
C(i,k) = max(3/22,1/20)=max(0.14,0.05)=0.14
C(k,d)= max(3/22,2/20)=max(0.14,0.1)=0.14
C(b,c)= max(3/22,1/20)=max(0.14,0.05)= 0.14
C(c,e)= max(2/22,2/20) =max(0.09,0.1)= 0.1
C(a,e)= max(2/22,3/20)=max(0.09,0.15)= 0.15
C(f,e)= max(1/22,1/20)=max(0.045,0.05)=0.05
C(f,i) = max(2/22,3/20)=max(0.09,0.15) = 0.15
C(i,d)= max(7/22,6/20)=max(0.318,0.3) =0.318
C(i,h)= max(4/22,5/20)=max(0.18,0.25)=0.25
C(c,a)= max(5/22,6/20)=max(0.23,0.3)=0.3
C(j,i)= max(3/22,2/20)=max(0.14,0.1)=0.14
C(g,f)= max(5/22,3/20)=max(0.23,0.15)=0.23
C(h,d)= max(9/22,8/20)= max(0.41,0.4)=0.41
C(b,m)= max(8/22,7/20)=max(0.36,0.35)=0.36
C(m,h)= max(9/22,9/20)= max(0.41,0.45)=0.45
C(m,p)= max(8/22,9/20)=max(0.36,0.45)=0.45
C(p,j)= max(7/22,6/20)=max(0.318,0.3)=0.318
C (l,n)= max(6/22,7/20)= max(0.27,0.35)=0.35
C(n,k)= max(7/22,6/20)=max(0.318,0.3)=0.318
C(a,l)= max(7/22,8/20)=max(0.32,0.4)=0.4
C(l,i)= max(8/22,7/20)=max(0.36,0.35)=0.36
C(n,g)= max(7/22,5/20)= max(0.318,0.25)=0.318

The arbitrary topology with appropriate edge cost values are shown in Fig.3.

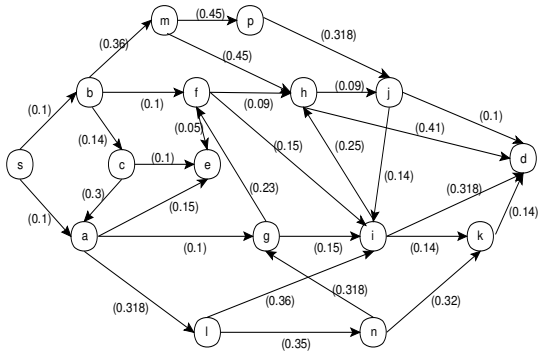

Fig. 3. Arbitrary Topology with DCMCR Cost Values

Cost of each path is calculated as summation of edge weights of the paths as shown below:

1) D(sbfhjd) = 0.48
2) D(sbfikd) = 0.63
3) D(sbfihjd) = 0.79
4) D(sbfhjikd) = 0.8
5) D(sagikd) = 0.63
6) D(sagihjd) = 0.79
7) D(sagfikd) = 0.86
8) D(sagfhjd) = 0.71
9) D(sagfhjikd) = 1.03
10) D(sagfihjd) = 1.02
11) D(sbcagikd) = 1.07
12) D(sbcagihjd) = 1.23
13) D(sbcagfikd) = 1.30
14) D(sbcagfhjd) = 1.15
15) D(sbcagfihjd) = 1.46
16) D(sbcagfhjikd) = 1.47
17) D(sbmpjd) = 1.328
18) D(sbmhjd) = 1.1
19) D(sbmhd) = 1.32
20) D(sagid) = 0.668
21) D(salnkd) = 1.228
22) D(salikd) = 1.058
23) D(salid) = 1.096

Then, to find the feasible path(s) using DCMCR Algorithm

1) Consider p(sbfhjd) path
   D(p) = 5 ≤ 10 (i.e. it satisfies the delay constraint)
   C(p) = 0.48
2) Consider p(sbfikd) path
   D(p) = 5
   C(p)=0.63
3) Consider p(sbfihjd) path
   D(p) =10
   C(p)= 0.79
4) Consider p(sagikd) path
   D(p) =5
   C(p)= 0.63
5) Consider p(sagihjd) path
   D(p) =10
   C(p)= 0.79
6) Consider p(sagid)
   D(p)= 9
   C(p)= 0.668

These six paths are selected in the 23 total available paths. In other paths, the first constraint is not satisfied and hence those paths are not selected.

DCMCR returns the following path as optimum as its cost is least in all six feasible paths.

P_{DMR} (sbfhjd) = D(P) =5
C(P)= 0.48

From the Fig. 4 it is observed that the region below the plane offers the feasible paths.

5.1 Testing Procedure

DCMCR uses the testing to refine the lower and upper bound values. For testing procedure construct an auxiliary graph $G_{aux}(G,s,d,K.\Delta_d, (V-1/\alpha),D,C_{aux})$ and define Test (T, α) =YES if MCPv2.2(G_{auxs},s,d,K.Δ_d,(V-1/α),D,C_{aux}) has feasible solution, otherwise define Test(T, α) =NO where T and α are real numbers i.e. find an source-destination path P_{aux} in G_{aux} such that D(P)= Δ_d and C(P)≤ (V-1)/ α.

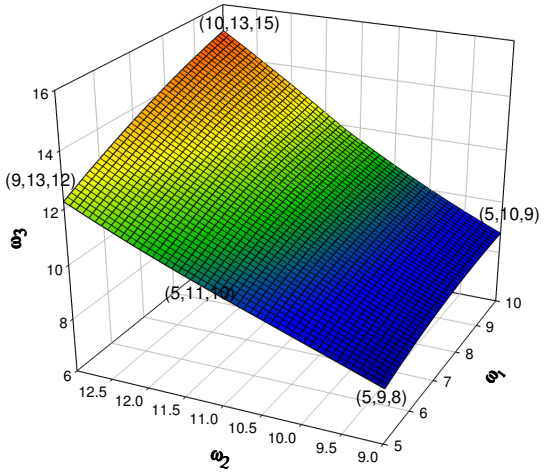

Fig. 4. Optimal Path Weights of DCMCR on 3D Plane

In MCPv2.2 cost constraint is defined as C_{aux} and $C_{aux}(e)=C(e).\gamma$ where $\gamma = \left(\frac{V-1}{T.W.\alpha}\right)$. In the arbitrary topology the following values are defined. The value of γ determines the auxiliary graph i.e. based on Testing values, weight bound and approximation factor γ is computed.

N=16, T=1, α=1 and W=10.

$$\gamma = \left(\frac{V-1}{T.W.\alpha}\right) = 1.5$$

Consider the path p(sbfhjd)

$$C(p)= \max\left(\frac{w_k(p)}{W}\right) \text{ where } 2\le k\le K.$$

C_{aux} (s,b) =C(s,b).γ
C_{aux} (s,b) =[0.1*1.5] = 0.15
C_{aux}(s,b) =[0.1*1.5] = 0.15
C_{aux} (b,f) =[0.1*1.5] = 0.15
C_{aux} (f,h) =[0.09*1.5] = 0.135
C_{aux} (h,j) =[0.09*1.5] = 0.135
C_{aux} (j,d) =[0.1*1.5] = 0.15
C_{aux} (s,a) =[0.1*1.5] = 0.15
C_{aux} (a,g) =[0.1*1.5] = 0.15
C_{aux} (g,i) =[0.15*1.5] = 0.225
C_{aux} (i,k) =[0.14*1.5] = 0.21
C_{aux} (k,d) =[0.14*1.5] = 0.21
C_{aux} (a,e) =[0.15*1.5] = 0.225
C_{aux} (b,c) =[0.14*1.5] = 0.21
C_{aux} (c,e) =[0.1*1.5] = 0.15
C_{aux} (c,a) =[0.3*1.5] = 0.45
C_{aux} (g,f) =[0.23*1.5] = 0.345
C_{aux} (h,d) =[0.41*1.5] = 0.615
C_{aux} (f,i) =[0.15*1.5] = 0.225
C_{aux} (i,h) =[0.25*1.5] = 0.375
C_{aux} (j,i) =[0.14*1.5] = 0.21
C_{aux} (f,e) =[0.05*1.5] = 0.075
C_{aux} (a,l) = [0.318*1.5]=0.477
C_{aux} (l,n) = [0.35*1.5]= 0.525
C_{aux} (l,i) = [0.36*1.5]= 0.54
C_{aux} (n,g) = [0.318*1.5]= 0.477
C_{aux} (n,k) = [0.32*1.5]= 0.48
C_{aux} (b,m) = [0.36*1.5]=0.54
C_{aux} (m,p) = [0.45*1.5]=0.675
C_{aux} (m,h) =[0.45*1.5] = 0.675
C_{aux} (p,j) =[0.318*1.5]= 0.477
C_{aux} (i,d) =[0.318*1.5]= 0.477

Fig.5. represents the auxiliary graph with new weights.

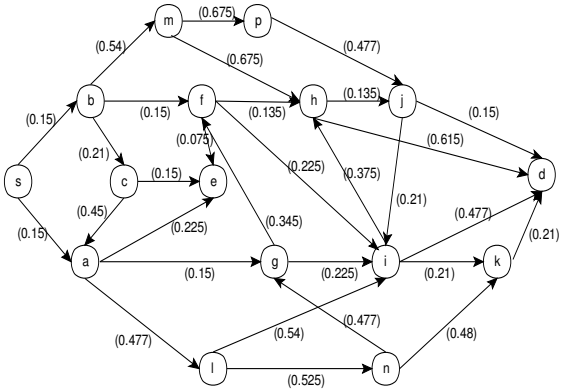

Fig. 5. Auxiliary Graph with New Weights

$D(p(sbfhjd))=D(s,b)+D(b,f)+D(f,h)+D(h,j)+D(j,d)$

$= 1+1+1+1+1= 5 \leq \Delta_d$

$C(p(sbfhjd))=C(s,b)+C(b,f)+C(f,h)+C(h,j)+C(j,d)$

$=0.15+0.15+0.135+0.135+0.15=0.72$

$\leq \Delta_c$

$C_{aux}(p) \leq (V-1)/\alpha=15$, $C_{aux}(p) \leq 15$

$\therefore$ MCPv2.2(G_{aux}s,d, Δ_d , (V-1)/α, D, C_{aux}) has a feasible solution so define Test (T, α) = YES, this implies UB=T(1+ α)

$\therefore$ UB= 1(1+ α)= 2.

In arbitrary topology, the least delay weight ratio is 0.5 and the least cost weight is 0.48. The least delay weight ratio and the least cost weight ratio of ARPANET is 0.6 and 0.73 respectively. Similarly for ANSNET the values are 0.8 and 0.69 as shown in Fig. 6.

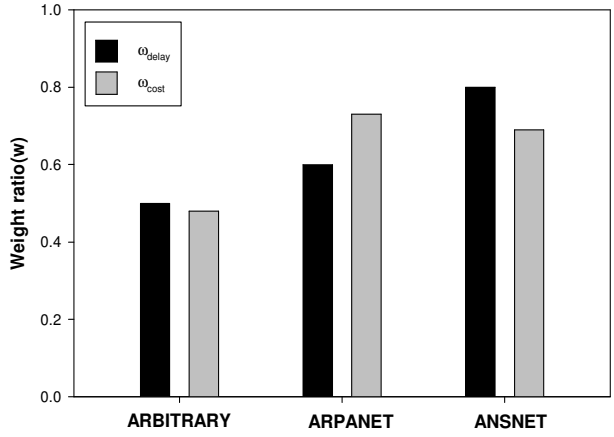

Fig. 6. Ratio of Path Weight of DCMCR Algorithm

6 Experimental Results

MCPv3, K-approximation and greedy algorithms OMCR were implemented and tested with DCMCR for performance evaluation. Arbitrary topology, ANSNET and ARPANET are used for simulation. The edge weights are uniformly selected in the range (1, 10). Constraint bound values are chosen as W=3, 10 and 20. The value of K is chosen as 3. The source-destination pairs are randomly generated so that the minimum hop-count between them is at least two. Simulations were carried out in NS-2 [26]. The experiments are conducted in Core2Duo, 2.4 GHz, 1 GB Memory with Linux operating system computer. The node configuration is shown in Table 1.

6.1 Comparison of MCPv3 and DCMCR

Execution time of DCMCR (Delay Coerced Multi Constrained Routing) is compared with MCPv3. Both algorithms perform almost similar in terms of execution time. However, DCMCR takes a little more time than MCPv3. This is due to coercing one constraint. DCMCR computing path with coercement of one constraint and finds (1+α)(K-1) approximated optimum solution. But, MCPv3 considers all K constraints equally and computes (1+ α) approximated path. In fact, DCMCR is found useful where a particular constraint bound W is to be strictly satisfied e.g. tight constraint applications.

Table 1 Node Configuration

S.No	Number of Nodes	Area in Sq.m
1	40-60	500 × 500
2	80-100	1000 × 1000
3	120	1500 × 1500

Various simulation details are shown in Table 2.

Table 2 Various Simulation Details

S.No	Simulation Parameter	Remarks
1	Placement of Node	Random
2	Topology Chosen	Arbitrary, ARPANET and ANSNET
3	Distance Between Nodes	Euclidean
4	Range of Value of Constraints	Between 1 and 10
5	Nature of Constraints	Additive
6	Packet size	512 bytes

Moreover, whenever topology changes DCMCR adopts edge relaxation. Topology changes should reflect in the node where changes occur and also in its parent node. Some edges or nodes or both are pruned so that the path computed is always feasible and optimum for a particular source-destination pair. MCPv3 also performs edge relaxation methodologies whenever a change in topology found, but running time is less compared to DCMCR as MCPv3 considers all K-constraints equally and is evident from the Fig. 7. However the

two other algorithms i.e. Greedy and K-approximation obviously take longer time to compute the path while comparing to MCPv3 and DCMCR.

Fig.8. shows the effect of running time of MCPv3 and DCMCR with reference to number of nodes. In this case both algorithms perform in similar fashion. When the number of node increases, the execution time also increases and it is negatively correlated with approximation factor (α) for both algorithms.

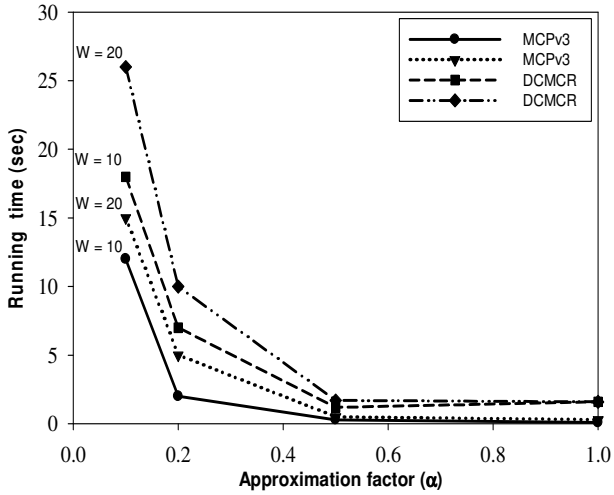

Fig. 7. Running Time of MCPv3 and DCMCR Vs Approximation Factor

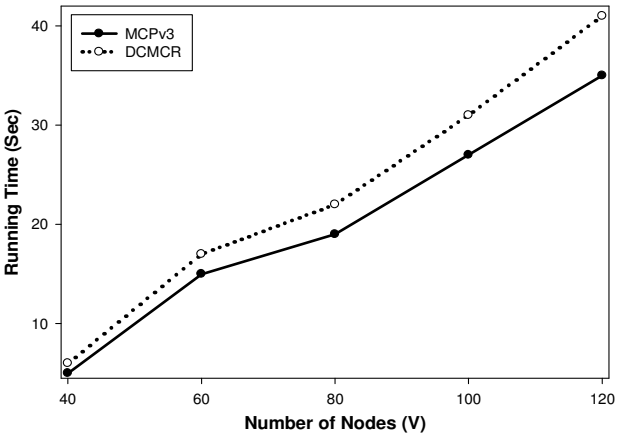

Fig.8 Running Time of MCPv3 and DCMCR Vs Number of Nodes

However, DCMCR takes little more time because of coercing one constraint. This is especially true when the size of the network grows. Because DCMCR has to satisfy the first constraint strictly in all the paths computed. Longer running time is experienced especially when the source and destination nodes are far apart. This is the reason why the time difference is kept increasing when the number of nodes increase. But DCMCR has the performance guarantee, as that of MCPv3 and the ability to bring the feasible path almost in all situations i.e. the error ratio is small. Simulation results revealed that DCMCR and MCPv3 perform almost in similar fashion irrespective of value W and both algorithms find feasible optimum paths.

6.2 Analysis of Success Ratio

Arbitrary networks are generated by using Waxman graph algorithm [22]. Here 'n' nodes are randomly distributed and each node is placed at a location with integer coordinates. The Euclidean metric is then used to determine the distance between each pair of nodes. On the other hand edges are introduced between pairs of nodes s-d with a probability that depends on the distance between them. The edge probability is given by $p(u,v) = \beta\exp(-d(u,v)/\alpha L)$ where $d(u,v)$ is the distance from node 'u' to 'v', L is the maximum distance between two nodes and α, β are parameters in the range between 0 and 1. Larger values of β result in graphs with higher edge densities, while small values of α increase the density of short edges relative to lower ones [9, 21].

Success Ratio (SR) of various algorithms at different nodes with n=40, 80 and 120 is shown in Fig.9. Success Ratio is defined as the percentage of time that the algorithm finds a feasible path, at least one exists [10, 19]. High success ratio could be achieved when the number of nodes is between 80 and 100. MCPv3 achieves better success ratio than greedy and K-approximation algorithms almost in all the three cases. These tests are experimented in randomly generated graphs. Success ratio and number of nodes are positively correlated for MCPv3. Success ratio of DCMCR is little less than MCPv3 due to coercing one constraint.

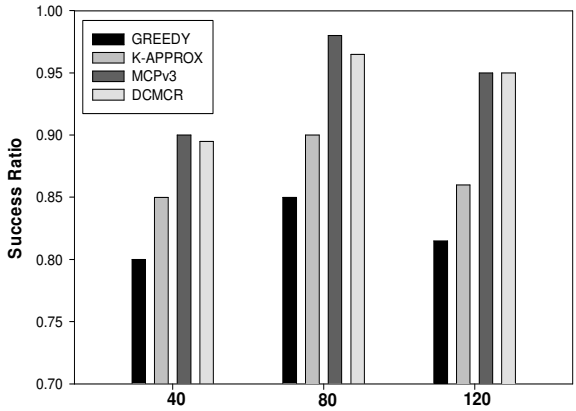

Fig.9. Success Ratio of Various Algorithms

DCMCR algorithm provides approximation solution to restricted shortest path problem and is readily applicable to underlying MCP problem. By coercing one constraint, it is obvious that the number of feasible paths is fewer than MCPv3. However DCMCR is much suited for an application where a constraint is to be strictly satisfied. In greedy and K-approximation algorithms, the success ratio is comparatively lower and moreover time consumed to find the feasible paths are also high.

A routing algorithm behaves well, when its success ratio is high and the cost of the routes is low. Higher success ratios and lower costs result in better 'overall performance'. Fig.10 illustrates the variation of success ratio with the increasing number of feasible paths for different randomly generated graphs whose size is varying from 20 to 120. Success ratio of MCPv3 increases when the value of K=3 for any number of nodes. MCPv3 finds feasible paths irrespective of nature of constraints i.e. tight or loose constraints. This is due to optimum selection of approximation factor and x_{opt}. MCPv3 is able to determine paths irrespective of size of network.

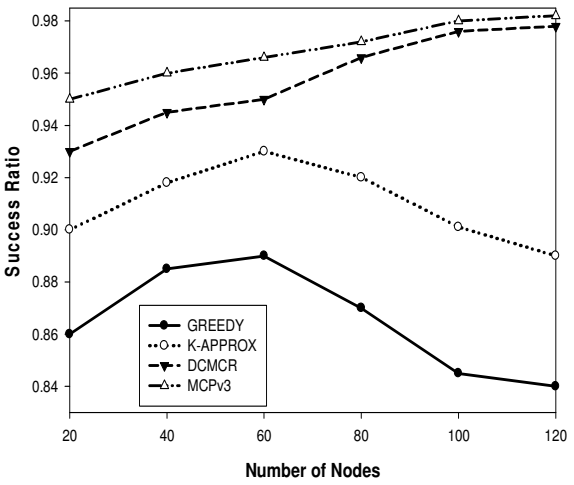

Fig.10. Success Ratio Vs Number of Nodes when K=3 for Various Algorithms

In fact, DCMCR and MCPv3 perform similarly even though DCMCR need to coerce one constraint. DCMCR approximates remaining two constraints and hence its performance is compared with MCPv3. But greedy and K-approximation algorithms are not comparatively better. This is due to absolute selection of weight in greedy algorithm and K-approximation is simple in nature and its time complexity is not linear. Therefore when the number of constraints K≥3 the success ratio is not comparable with MCPv3 and DCMCR. It is particularly true when the link weights are negatively correlated, finding a feasible path becomes hard for K-approximation algorithm.

The effect of success ratio Vs different number of constraints is considered shown in Fig.11. Success ratio and number of constraints are negatively correlated in all the algorithms. However, the performance of Greedy and K-approximation algorithms affected by the number of constraints is more serious than the proposed algorithms. When the number of constraints is more than three, the performance of Greedy and K-approximation algorithms are poor. Greedy algorithm considers the true path length and K-approximation uses upper bound of the path while making decisions. Appropriate testing procedures are conducted to choose x_{opt} in MCPv3 and DCMCR algorithms and hence better performance is achieved.

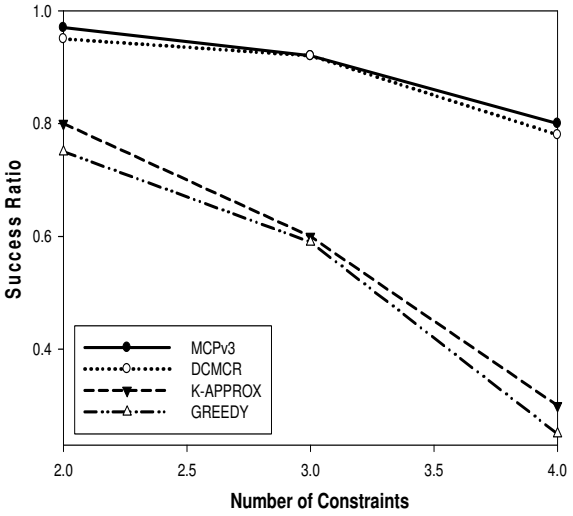

Fig. 11 Number of Constraints Vs Success Ratio

Fig.12 shows effect of success ratio with the increasing number of feasible paths for different number of nodes. It is obvious that the success ratio and feasible paths are positively correlated regardless of number of nodes. It is observed that the success ratio varies sharply especially when number of nodes is between 60 and 120 and the number of feasible paths p≥2. More number of feasible paths can be achieved by better edge relaxation techniques and better average node degree. An average node degree between 7 and 8 is achieved in the generated random graph. In this case also DCMCR and MCPv3 outperform the other algorithms.

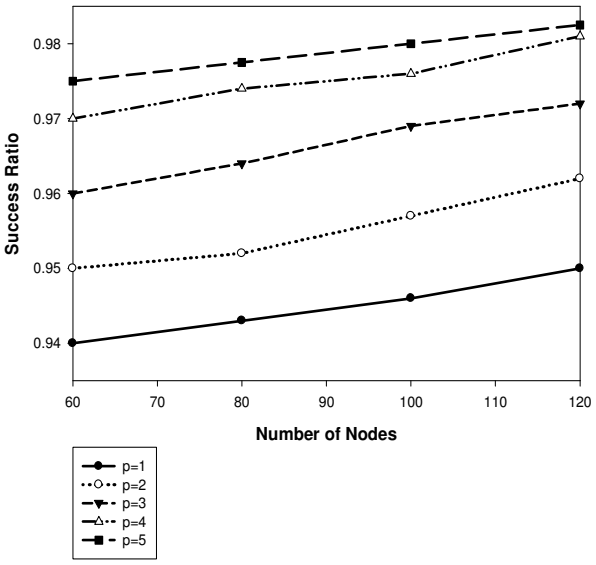

Fig.12. Number of Feasible Paths Vs Success Ratio for Various Number of Nodes

6.3 Effect of Throughput, Delay and Bandwidth on Various Number of Nodes

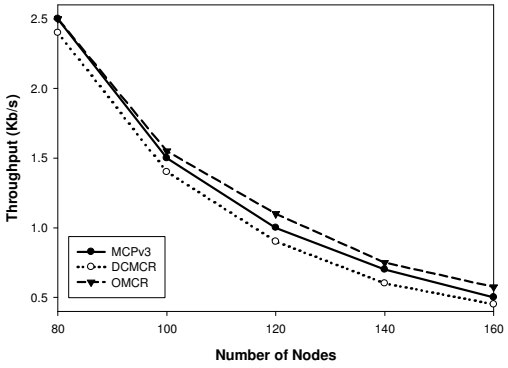

Fig. 13. Throughput Vs Number of Nodes

The effect of throughput for various number of nodes is shown in Fig.13 All the nodes are connected by directed links and each node has a node degree of 3 and above. The delay of each link is between 1 ms and 10 ms. Routing requests are generated randomly. The network throughput starts to decrease when the load of the network increases. In DCMCR, due to the one constraint coercement the throughput starts to decline when the number of nodes is high. When the network size grows, the available throughput decreases due to more number of flows. But, the rate at which the throughput is decreases is lesser in MCPv3 than DCMCR.

However, both algorithms can be applied to distributed network environment. If the number of nodes is high there will be more bottleneck bandwidth and hence the throughput decreases. Nevertheless even when the network load is high the proposed algorithms are more efficient since the throughput is less; because both MCPv3 and DCMCR are path constrained algorithms. Any bottleneck bandwidth may simply be omitted in the process of finding feasible paths. MCPv3 and DCMCR achieve load balancing by using more than one path. Other algorithms drop the packets because of congestion in a single path. Throughput is not affected much by the link quality when there are only a few hops. When the network size increases the proposed algorithm suffer from accumulative link loss.

The effect of delay and throughput is shown in Fig.14. Both algorithms maintain the delay less than 150 msec for almost all the throughput value. In fact, increasing throughput results in increased delays. These effects are more conspicuous during periods of congestion. Higher delay may be experienced when the source and destination are far apart or when the network experiencing congestion.

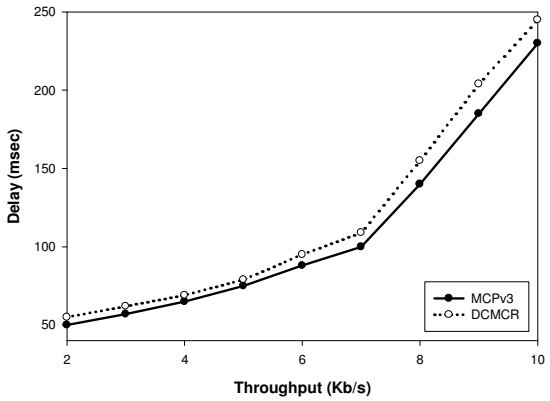

Fig. 14. Delay Vs Throughput

Any good routing algorithm should keep the delay as low as possible for almost all values of throughput. In MCPv3 and DCMCR delay start to rise only when the network size grows beyond 200 nodes, or if the network experience congestion. As long as the network size is below 200 both algorithms experience reasonably acceptable delay limits. DCMCR shows little excess delay due to the delay coercement. Moreover, by choosing

appropriate approximation factor and other parameters MCPv3 and DCMCR are perform better.

The Number of Messages for various bandwidths is shown in Fig. 15. Routing messages are counted that each node sends to its neighbours in order to establish an end-to-end delay-throughput constrained route. The performance of MPOR and DMR in tight bandwidth requirement seems to be worse. This is due to more control messages are to be sent to update the network when compared to OMCR. The performance of DMR and OMCR are almost similar. But MPOR is drastically underperforms; this is due to it produces large amount of packets in the case of network failures. The convergence time is also less because of the less number of packets that are being exchanged in case of network failures. Multiple vectors reduce the overhead in OMCR and hence the available bandwidth is effectively utilized in the proposed algorithms. But in MPOR only one message accommodates in one packet and hence the bandwidth is being wasted in the case of network failure. Due to effective utilization of throughput the convergence time is less in OMCR.

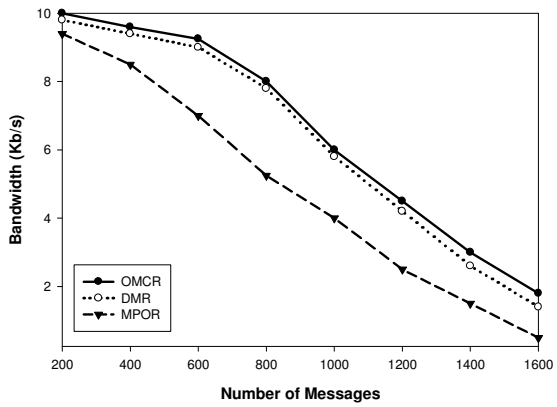

Fig.15. Number of Messages Vs Bandwidth

The effect of throughput on various network load is shown in Fig.16.When the network load increases throughput of all the proposed algorithms are kept decreasing. In the case of OMCR the throughput steadily deceases when the network load increase. MCPv3 and DCMCR normally avoid the links that have little bandwidth by proper edge relaxation. However, at heavy loads and/or the number of nodes of the network increases the throughput declines. In OMCR, due to better vector transformation techniques the throughput is effectively managed. However, whenever the network traffic increases significantly throughput declines. This is especially true when most of the capacity of each link is reserved for guaranteed sessions.

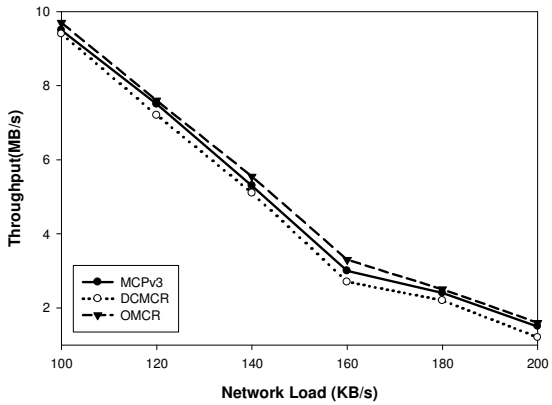

Fig. 16. Network Load Vs Throughput

6.4 Effect of Packet Size

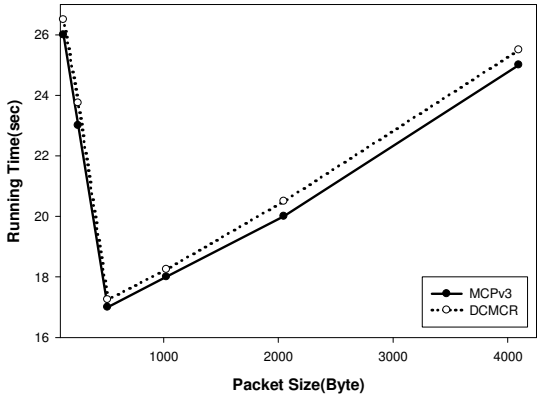

Fig. 17. Packet Size Vs Running Time

The effect of running time for different packet sizes is shown in Fig.17 .The better performance of the proposed schemes is achieved when the packet size is 512 or 1024 bytes. Beyond this range the performance of the proposed algorithms are slightly less. Smaller packets can cause more router overhead in terms of checksum overhead i.e. each packet has to undergo error checking and hence more time is consumed in router. Because of this reason the running time is high when the packet size is 128 bytes. Error checking process is less in the case of 512 or 1024 byte packets. If the packet size is high i.e. 2048 or 4096, the routers may start to

drop the packets because of the bottleneck bandwidth or congestion. Otherwise the packets are to be fragmented, if the packet size is greater than the MTU (Maximum Transmission Unit) of a link e.g. Ethernet the MTU is 1500 bytes and for point to point networks is 532 bytes. This causes time delay in reaching the destination and hence running time of the algorithm is increased.

The effect of success ratio for different packet size is shown Fig.18. .In OMCR success ratio is subdued when the packet size is too small or too big. Small packets cause the router to get congested because of the header processing delay i.e. time taken to read the header checksum. But in larger packets cannot be accommodated due to bandwidth constraints. Either they must be fragmented or to be dropped at router. Fragmentation and reassembly consumes time and hence the success ratio is affected if the packet size is beyond 512 to 1024 bytes.

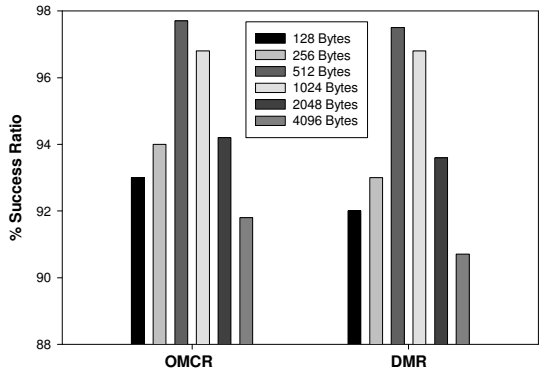

Fig..18. Packet Size Vs % Success Ratio

Generally if the packet size is too low, streams of traffic is broken up into a relatively large number of small packets that adversely affects the performance. On the other hand if the packet size is set too high, it may exceed the networks physical MTU and degrade performance because each packet needs to be subdivided into smaller ones i.e. fragmentation. The default value of maximum packet size is 1500 bytes for broadband connection and 576 bytes for dialup connection.

7 Conclusion

The proposed algorithm is a multi-constrained (1+α) (K-1) approximation algorithm with K≥2 additive constraints. DCMCR approximates (K-1) constraints while coercing one of the constraints. This algorithm finds a feasible solution whose path weight is bound by first constraint and approximating remaining (K-1) constraints. DCMCR is applied where one of the QoS constraints is coerced and remaining constraints are approximated. By choosing appropriate value of α and constraint bounds, the proposed algorithm outperforms its contenders. When the K is equal to 2, this algorithm produces (1+α) approximation with better running time than that of its contenders designed for this purpose. Experimental validation reveals that DCMCR performs better to achieve comparatively reduced running time and delivers well in tight constraint scenario. DCMCR shows better overall performance when the number of nodes is high. The proposed algorithm outperforms in terms of running time, success ratio and scalability by choosing appropriate approximation factor, x_{opt} and effective testing procedures.

References:

[1] Abdelhamid Mellouk, Said Hoceini, and Sherali Zeadally, "Design and performance analysis of an inductive QoS routing algorithm", Computer Communications, Elsevier Publications, Vol. 32, 2009, pp. 1371-1376.

[2] BenAli, N., Belghith, A., Moulierac, J. and Molnar, M., "QoS Multicast aggregation under multiple additive constraints", *Computer Communications, Elsevier Publications,* Vol. 31, 2008, pp. 3564-3578.

[3] Chen Shigang and Nahrstedt Klara, "An overview of quality of service routing for next-generation high speed networks: Problems and Solutions", Journal on IEEE Network, Vol. 12, No. 6, 1998, pp 64-79.

[4] S. Chen and K. Nahrstedt, "On finding multi-constrained paths," In *Proc. IEEE ICC*, 1998, pp. 874–879.

[5] Ergun Funda, Sinha Rakesh and Zhang Lisa, "An improved FPTAS for restricted shortest path", Information Processing Letters, Vol. 83, No. 5, 2002, pp. 287-291.

[6] A. Goel, K. G. Ramakrishnan, D. Kataria, and D. Logothetis, "Efficient computation of delay-sensitive routes from one source to all destinations," In *Proc. IEEE INFOCOM*, 2001, pp. 854–858.

[7] Hua Wang, Xiangxu Meng, Shuai Li and Hong Xu, "A tree-based particle swarm optimization for multicast routing", Computer Networks: The

International Journal of Computer and Telecommunications Networking, Vol. 54 , No. 15, 2010, pp. 2775-2786.

[8] A. Juttner, B. Szviatovszki, I. Mecs, and Z. Rajko, "Lagrange relaxation based method for the QoS routing problem," in *IEEE INFOCOM*, 2001, pp. 859–868.

[9] Keen-Mun Yong, Gee-Swee Poo, and Tee-Hiang Cheng,"Proactive rearrangement in delay constrained dynamic membership multicast", Computer Communications, Vol. 31, 2008, pp. 2566-2580.

[10] Khadivi, P., Samavi, S. and Todd, T.D., "Multi-constraint QoS routing using a new single mixed metrics", Journal of Network and Computer Applications, Elsevier Publications, Vol. 31, 2008, pp. 656-676.

[11] Kwei-Jay Lin, Jing Zhang, Yanlong Zhai and Bin Xu, "The design and implementation of service process reconfiguration with end-to-end QoS constraints in SOA", Journal of Service Oriented Computing and Applications, Vol. 4, No. 4, 2010,pp. 157-168.

[12] Leela, R., Thanulekshmi, N. and Selvakumar, S., "Multi-constraint QoS Unicast Routing Using Genetic Algorithm (MURUGA)", Applied Soft Computing, Elsevier Publications (Accepted for Publication), 2010.

[13] Li, Z., "A distributed approach for multi-constrained path selection and routing optimization", In Proc. of ACM International Conference Proceeding Series, 3rd International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks 2006 (QSHINE "06), Vol. 191, 2006,Article No. 36.

[14] Lorenz, D.H. and Raz, D., "A simple efficient approximation scheme for the restricted shortest path problem", Operation Research Letters, Vol. 28, No. 5, 2001,pp. 213-219.

[15] Ping, Chen Bingcai, Gu Xuemai and Liu Gongliang, "Multi-constraint quality of service routing algorithm for dynamic topology networks", *Journal of Systems Engineering and Electronics, Elsevier Publications,* Vol. 19, No. 1, 2008, pp. 58-64.

[16] Ronghui Hou, King-Shan Lui, Ka-Cheong Leung and Fred Baker, "Approximation algorithm for QoS routing with multiple additive constraints", *In Proc. IEEE International Conference on Communications 2009 (ICC 2009),* Vol. 1, 2009,pp. 1-5.

[17] Sameer Qazi and Tim Moors, "On the impact of routing matrix inconsistencies on statistical path monitoring in overlay networks", *Computer Networks, Elsevier Publications,* Vol.54,No.10, 2010, pp. 1554-1572.

[18] Shuchita Upadhyaya and Gaytri Dhingra, "Exploring issues for QoS based routing algorithms", *International Journal on Computer Science and Engineering*, Vol. 2, No. 5, 2010, pp. 1792-1795.

[19] Teerapat Sanguankotchakorn and Newton Perera, "Hybrid Multi Constrained Optimal Path QoS Routing with inaccurate link state", *In Proc. Ninth International Conference on Networks (ICN),* 2010, pp. 321-326.

[20] Therence Houngbadji and Samuel Pierre, "QoSNET: An integrated QoS network for routing protocols in large scale wireless sensor networks", *Computer Communications, Elsevier Publications,* Vol. 33, No. 11, 2010, pp. 1334-1342.

[21] Tzu-Lun Huang and Lee, D.T., "A distributed multicast routing algorithm for real-time applications in wide area networks", *Journal of Parallel Distributed Computing,* Vol. 67, 2007, pp. 516-530.

[22] Waxman, B.M., "Routing of multipoint connection", *IEEE Journal on Selected Areas in Communications,* Vol. 6, No. 9, 1988, pp. 1617-1622.

[23] G. Xue, "Minimum cost QoS multicast and unicast routing in communication networks," *IEEE Trans. Commun.*, vol. 51, no. 5, 2003, pp. 817–824.

[24] Yan Xing Zheng, Turgay Korkmaz and Wenhua Dou, "Two additive- constrained path selection in the presence of inaccurate state information", *Computer Communications, Elsevier Publications,* Vol. 30, 2007,pp. 2096-2112

[25] Yu Wang, Lemini Li and Du Xu, "Pervasive QoS routing in next generation networks", *Computer Communication, Elsevier Publications,* Vol. 31, 2008, pp. 3485-3491.

[26] Kevin Fall and Kannan Varathan, "The ns Manuals, The Vint Project", *University of California, Berkeley, USA*, pp. 28-130, 2007.