Bandwidth Sensitive Routing in DiffServ Networks with Heterogeneous Bandwidth Requirements

Jun Wang, Li Xiao, King-Shan Lui, Klara Nahrstedt
Department of Computer Science
University of Illinois at Urbana-Champaign
{junwang3, lixiao, kinglui, klara}@cs.uiuc.edu

***Abstract*— This paper studies the problem of finding optimal routes for premium class traffic in a DiffServ network such that (1) loop-freedom is guaranteed in the entire network under hop-by-hop routing assumption; and (2) the *maximum relative congestion* among all links is minimized. This problem is called the *Extended Optimal Premium Routing* (eOPR) problem, which is proven to be NP-hard.**

We use the *integer programming method* to mathematically formulate the eOPR problem and find the optimal solutions for small scale networks. We also study heuristic algorithms in order to handle large scale networks. Simulation results are compared with the optimal solutions obtained by solving the integer programming models. The results show that the Bandwidth-inversion Shortest Path (BSP) algorithm can be a good candidate to route premium traffic in DiffServ networks.

I. Introduction

To provide better end-to-end Quality of Service (QoS) to applications, the Differentiated Service (DiffServ) scheme has been proposed as a cost-effective solution [1], [2]. In DiffServ networks, traffic is classified into multiple service classes, in which the *premium* class has the highest priority. However, all traffic between each source-destination pair may flow along the same path independent of class of service. The reason is that DiffServ scheme was designed originally and intentionally to be decoupled from IP routing. Without taking service differentiation into consideration during the routing process, some inappropriate paths might be used by both premium and best-effort traffic so that the premium class traffic, due to its higher priority, could impose very negative *inter-class effects* [3] on best-effort traffic, including larger delay, higher packet loss rate and even bandwidth starvation. Therefore, it is important to find optimal routes for premium class traffic under both DiffServ and hop-by-hop IP routing assumptions, which minimize the negative inter-class effects.

The higher ratio of bandwidth is used by the premium traffic on a link, the larger inter-class effects will be. Therefore, a natural way to find better premium paths is to minimize such bandwidth ratio occupied by premium traffic among all links. We call this bandwidth ratio occupied by premium traffic of a link the *relative congestion* ($\mathcal{RC}$) of the link. Then, given a network, the optimal routing algorithm needs to find routes for the premium traffic such that the maximum value of relative congestion in the network is minimized[1]. This routing problem is called the Extended Optimal Premium-class Routing (eOPR) problem[2] and is proven to be NP-hard (see Appendix). The original OPR problem was raised in [3], where all nodes in a DiffServ network were considered as edge nodes and the bandwidth requirements among all nodes were homogeneous. In this paper, we study a more realistic model, in which nodes are classified into edge nodes and interior nodes and premium bandwidth requirements among edge nodes are heterogeneous.

The rest of the paper is organized as follows. Section II presents the system model, our assumptions and the eOPR problem formulation. The approach based on integer programming is introduced in Section III to obtain the optimal solution to the eOPR problem. Two heuristic algorithms are also introduced there. The performance issues are investigated in Section IV. Some previous work is covered in Section V. Finally, Section VI concludes this paper and the Appendix part gives a brief proof of the NP-hardness of the eOPR problem.

II. System model and problem formulation

A. Network model

A DiffServ network is defined as $\mathcal{N} = (G, V_e, C)$, in which $G = (V, E)$ is a directed graph with V as its node set and E as its link set. $V_e \subseteq V$ is the *edge node set* which contains all edge nodes in the network. $C = \{c_{xy} | \forall (x, y) \in E\}$ defines the capacities of all links, where (x, y) represents a link from node x to node y.

A path from the source node v_1 to the destination v_n is written as $p(v_1, v_n) = \langle v_1, v_2, \cdots, v_{n-1}, v_n \rangle$. A path is *simple* or *loop-free* if all nodes from v_1 to v_n are distinct. If $v_1 = v_n$, then $p(v_1, v_n)$ forms a *loop*. Specially, if a path is used by premium traffic, then we call it a *premium path*. If a link is shared by multiple premium paths between different pairs of

This work was supported by NSF Grant under contract number NSF ANI 00-73802 and NSF CISE Grant under contract number NSF EIA 99-72884. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Please address all correspondences to Jun Wang and Klara Nahrstedt at Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, phone: (217) 244-5841, fax: (217) 244-6869.

[1] That is, the maximum ratio, among all links, of the total amount of reserved bandwidth for the premium traffic divided by the link capacity is minimized.

[2] More formal definition of the eOPR problem and some examples are presented later in Section II.

0-7803-7802-4/03/$17.00 © 2003 IEEE

edge nodes, then it has to reserve the *sum* of the bandwidth requirements for all premium traffic passing through it.

Since only the edge nodes generate premium traffic, the bandwidth requirements among all the edge nodes are defined as $\mathcal{Q} = \{r_{xy} | x, y \in V_e\}$. As we can see, different routing algorithms may result in different traffic distributions among links. We use $\mathcal{F} = \{f_{xy} | (x, y) \in E\}$ to denote the reserved bandwidth (by the premium traffic) on links. Finally, given $\forall (x, y) \in E$, we define $\mathcal{RC}(x, y) = \frac{f_{xy}}{c_{xy}}$ as the *relative congestion* of the link (x, y). Among all the $\mathcal{RC}$ values in a given network, the maximum one is defined by $Y = \max_{(x,y) \in E} \{\mathcal{RC}(x, y)\}$, which is the optimization target in our eOPR problem.

B. Assumptions

Our formulation of the eOPR problem is based on the following assumptions:

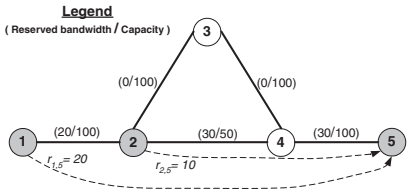
- Each router can obtain the topology information by some link-state routing protocol such as the Open Shortest-Path First Protocol (OSPF).
- Hop-by-hop routing is used. Therefore, routing consistency (or loop-freedom) should be guaranteed [3], [4]. This implies that, for every destination node $v \in V_e$, all paths connecting other edge nodes to v forms a spanning tree.
- Queueing delay at each node along a path accounts for the most significant part of the entire end-to-end delay for that path. This implies that the premium class traffic, with the highest priority traffic, experiences almost no queueing delay [2], [5]. Therefore, choosing fairly longer (in terms of hop-count) paths for the premium traffic does not compromise the delay requirement.
- The current hop count shortest path (SP) algorithm is still used for routing the best-effort traffic in the network, due to its stability and the dominance in volume of the best-effort traffic.

C. Formulation of the eOPR problem

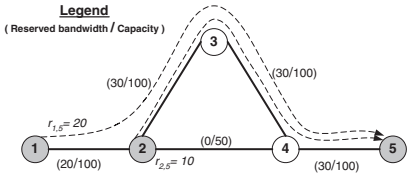
Given a DiffServ network $\mathcal{N}$ and premium bandwidth requirements $\mathcal{Q}$ among the edge nodes, different routing algorithms may choose different paths to route the premium traffic, thus producing different relative congestion values for every link in the network. Therefore, our target is to find a set of routes connecting the edge nodes which minimizes the maximum value of relative congestion among all links. This problem is called the Extended Optimal Premium-class Routing (eOPR) problem. It is NP-hard. The brief proof of the NP-hardness is given later in the Appendix. The optimal routing solution to this problem, $\mathcal{R}^*$, should achieve the optimum relative congestion value:

$$Y^* = min\{Y\} = min\left\{ \max_{(i,j) \in E} \{\frac{f_{ij}}{c_{ij}}\} \right\}$$

Figure 1(a) and 1(b) illustrate a simple example for the eOPR problem. The original topology consists of 5 nodes, 3 of which (node 1, 2, and 5) are edge nodes. The bandwidth requirement



(a) The maximum relative congestion produced by the SP algorithm $Y_{sp} = 0.6$



(b) The maximum relative congestion produced by the optimal algorithm $Y^* = 0.3$

Fig. 1. An example for the eOPR problem

set $\mathcal{Q}$ is composed of 2 requirements: $r_{1,5} = 20$ and $r_{2,5} = 10$. The numbers associated with every link represent *(Reserved bandwidth f / Link capacity c)* on that link. For instance, in Figure 1(a), since link $(2, 4)$ is used to carry the premium traffic from both node 1 and 2 to the destination node 5, the reserved bandwidth on $(2, 4)$ is $f_{2,4} = r_{1,5} + r_{2,5} = 30$. Figure 1(a) uses the SP routing and Figure 1(b) uses the optimal routing. From Figure 1(a), we can see that the maximum relative congestion, which the SP algorithm ($\mathcal{R}_{sp}$ in short) produces, is $Y_{sp} = 30/50 = 0.6$ with link $(2, 4)$ as the bottleneck link. While, as shown in Figure 1(b), the optimal routing algorithm $\mathcal{R}^*$ can achieve $Y^* = 30/100 = 0.3$,[3] which is definitely smaller than Y_{sp}.

III. Integer programming optimization and heuristic algorithms

A. Integer programming optimization

To obtain the optimal solution to the eOPR problem, it is mathematically formulated as an integer programming problem as presented below. Given a DiffServ network $\mathcal{N} = (G(V, E), V_e, C)$[4] and the bandwidth requirement set $\mathcal{Q} = \{r_{ik} | i, k \in V_e\}$[5], we define the following two sets of decision variables in Equations 1 and 2, where T_k denotes a spanning tree rooted at node k which covers all the edge nodes.

$$x_{ijk} = \begin{cases} 1, & (i, j) \in E, \quad k \in V_e, \quad (i, j) \in T_k \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

[3] Since node 1 chooses the path $\langle 1, 2, 3, 4, 5 \rangle$ to node 5, and node 2 is on this path, then node 2 has to choose $\langle 2, 3, 4, 5 \rangle$ as its own path to 5. Otherwise, inconsistency will occur at node 2.

[4] Recall that $V = [1, \cdots, n]$ is the entire set of nodes, $V_e \subseteq V$ is the edge node set, and $C = \{c_{ij} | \forall (i, j) \in E\}$ is set of link capacities.

[5] Note that if node i or k is not an edge node, then $r_{ik} = 0$.

$$f_{ijk} = \quad \text{bandwidth reserved on link } (i,j) \text{ for destination } k, \quad \text{for } \forall (i,j) \in E, \text{ and } k \in V_e \,. \tag{2}$$

Note that x_{ijk} are binary variables, and $f_{ijk} = 0$ when $x_{ijk} = 0$. Finally, the eOPR problem is formulated as follows:
Find variables x_{ijk}, f_{ijk} which satisfy

$$Y^* = \text{MIN} \left\{ \max_{(i,j) \in E} \left\{ \frac{\sum_{k \in V_e, k \neq i} f_{ijk}}{c_{ij}} \right\} \right\}. \tag{3}$$

subject to

$$\sum_{j=1}^{n} x_{ijk} = 1, \quad i \in V_e, (i,j) \in E, k \in V_e, i \neq k \tag{4}$$

$$\sum_{j=1}^{n} x_{ijk} \leq 1, \quad i \notin V_e, (i,j) \in E, k \in V_e, i \neq k \tag{5}$$

$$\sum_{j=1}^{n} f_{ijk} - \sum_{j=1}^{n} f_{jik} = r_{ik}, \quad i \in V, \quad k \in V_e \tag{6}$$

$$r_{ik} \cdot x_{ijk} \leq f_{ijk} \leq c_{ij} \cdot x_{ijk} \quad i \in V, j \in V, k \in V_e \tag{7}$$

$$x_{ijk} = 0 \text{ or } 1, \quad i \in V, j \in V, k \in V_e \tag{8}$$

Basically, the eOPR problem we are addressing in this paper is somewhat similar to the *Capacitated Miminal Spanning Tree* problems in [6], [7], [8] from the viewpoint of integer programming, except that (1) we are looking for a different target - minimizing the maximum relative congestion among all links, given exactly in Equation 3; (2) in the eOPR problem, nodes are divided into two categories: *edge nodes* which must be covered by the spanning tree, and *interior nodes* which can but do not necessary be on the resultant tree; and (3) the bandwidth requirements between edge nodes in our problem are heterogeneous.

Specifically, in the integer programming formulation of eOPR, equalities in Equation 4 ensure that exactly one link is going out from each non-root edge node for each spanning tree. On the other hand, if a node is an interior node, it could be on a tree or not, which is described by Equation 5. The equation in 6 describes flow conservation at each node. Together with the coupling constraints in Equations 7 and 8, it can guarantee that all the edge nodes are covered by a spanning tree (the interior nodes are not necessarily covered by the tree, though). The formal proof of the acyclic property of the tree was provided by Gavish in [9].

However, there is a potential problem in the above formulation. That is, there exist possibly some independent cycles among those free **interior** nodes. Since these cycles are not included in the spanning tree, they cannot be simply eliminated by the above constraints. We call this the *free cycle* problem. The simple solution to this problem, is to pretend that all interior nodes are edge nodes, but with negligibly tiny amount of bandwidth requirements. In this way, the resultant tree will cover all the nodes, thereby preventing cycles in the entire graph, while the target function of minimizing the relative congestion still remains unaffected (because the additional requirements added for those interior nodes are negligible).

To find the optimal solutions, we use the CPLEX [10] as the solver. Although the CPLEX is a strong solver to integer/linear programming problems, it still takes huge amount of time to search for the optimal solutions if the given networks contain more than 40 nodes. This is because the eOPR problem is NP-hard in nature.

B. Heuristic algorithms

Although we present the integer programming method to obtain the optimal solutions to the eOPR problem in the previous subsection, it is impossible to find such optimal solutions within polynomial time due to its NP-hardness, unless $P = NP$. Therefore, we need heuristic approximation algorithms for large-scale networks. In this paper, we study two heuristic algorithms: the *Widest Shortest Path (WSP)* algorithm and the *Bandwidth-inversion Shortest Path (BSP)* algorithm. Both algorithms are based on the Dijkstra's algorithm. They were studied in [3]. However, in this paper, we will apply them to the new eOPR problem. In the next section, we perform simulations to study their performance in this new scenario and compare the results with the optimal ones by solving the corresponding integer programming problems.

The WSP algorithm is the simplest heuristic algorithm which can achieve better relative congestion than the basic SP algorithm. The WSP first uses SP to find shortest path(s) between two given edge nodes. When a tie occurs, the WSP always chooses the widest path among the set of shortest paths between the given pair of edge nodes.

The BSP algorithm is basically a shortest-path algorithm with the distance or weight function defined as

$$w(i,j) = \frac{1}{c_{ij}} \quad \text{for } (i,j) \in E$$

and for a path $p = \langle v_1, v_2, \cdots, v_n \rangle$

$$w(\langle v_1, v_2, \cdots, v_n \rangle) = \sum_{i=1}^{n-1} \frac{1}{c_{v_i v_{i+1}}}$$

The BSP finds the "shortest" paths among edge nodes with respect to the weight function given above. The consistency issues of both WSP and BSP have been studied in [3], [4], [11].

IV. Optimization and simulation result analysis

To evaluate the performances of WSP and BSP in the eOPR scenario, simulations are conducted on different randomly generated network topologies. Simulation results are studied by comparing them with the optimal results.

A. Solve the integer programming problem

In order to get the optimal solution for a given network, its corresponding integer programming model should be solved. We use the ILOG CPLEX [10] as our optimization tool. The results are used as our benchmark to evaluate the real performances of the heuristic algorithms.

B. Simulation topologies and metric of interest

We use the topology generator BRITE [12] to randomly generate our network topologies, where the Waxman model is used and the nodes are placed according to the heavy-tail distribution. Link capacities are randomly generated between the interval $[10, 1024]$. The edge nodes are randomly selected from the entire node set. The bandwidth requirements are also randomly generated among all edge nodes.

Given a network and a routing algorithm, the maximum relative congestion value, Y, is evaluated as the performance metric and compared to the optimal result Y^*.

Topology	Execution Time (ms)			
	SP	WSP	BSP	OPT
$\|V\| = 10$	0.547	0.780	0.653	≈ 2000
$\|V\| = 15$	1.971	2.974	2.398	≈ 884000
$\|V\| = 20$	4.491	6.695	5.323	$> 10^8$
$\|V\| = 25$	10.287	15.596	12.605	
$\|V\| = 30$	18.049	27.767	21.190	

TABLE I

APPROXIMATE EXECUTION TIMES OF DIFFERENT ALGORITHMS

C. Simulation results and analysis

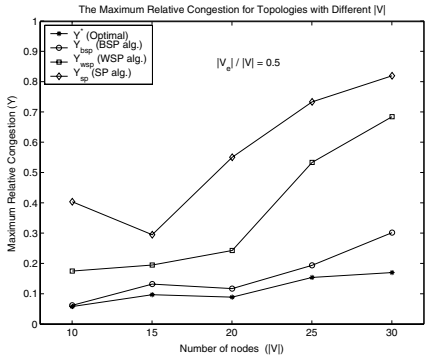


Fig. 2. Maximum relative congestion vs number of nodes ($|V|$)

Figure 2 shows the simulation and optimization results of the maximum relative congestion (Y) with respect to the number of nodes in the topology ($|V|$). We fix the ratio of edge nodes in the network to 0.5, i.e., $\frac{|V_e|}{|V|} = 0.5$. The results show clearly that, within the heuristic algorithms (WSP and BSP), the BSP performs the best. Actually, when $|V|$ is not very large, BSP can achieve results very close to the optimal ones. We also put the results of SP algorithm in the figure just for comparison.

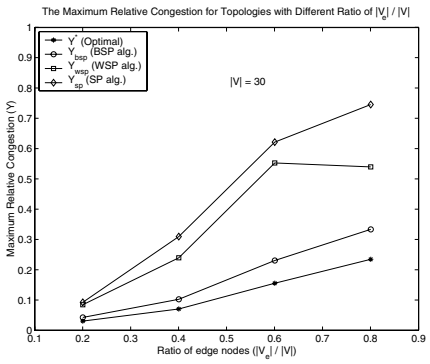


Fig. 3. Maximum relative congestion vs ratio of edge nodes ($\frac{|V_e|}{|V|}$)

Figure 3 shows the results of Y with respect to the ratio of edge nodes in the topology ($\frac{|V_e|}{|V|}$). $|V|$ is fixed to 30 in this case. Again, the BSP performs much better than the WSP, and yields results close to the optimal ones.

Although CPLEX can achieve the optimal results which are always better than the results of heuristic algorithms, its execution times are much larger, as shown in Table I, where we fix $\frac{|V_e|}{|V|} = 0.5$. The real numbers in the table may vary if different hardware platforms are used. But they show clearly that to obtain the optimal solutions may take the order of magnitude longer time than to get sub-optimal solutions by using some heuristic algorithms. Although in practice it is not efficient to use our integer programming model, it provides us a lower bound to evaluate heuristic algorithms.

In summary, both simulation and optimization results suggest that: (1) for small scale networks, we can get the optimal solutions to the eOPR problem by solving the integer programming models; (2) for large-scale networks, however, more efficient approximation algorithms are needed and the BSP algorithm can be a good candidate.

V. PREVIOUS WORK

In [13], S. Chen and K. Nahrstedt did a thorough survey on QoS routing algorithms. Our work in this paper is based on the hop-by-hop routing scheme instead of per-flow consideration. [4] studied hop-by-hop routing algorithm issues, such as isotonicity, search of optimal paths, etc. The author provided an elegant algebra basis to study the QoS routing issues in the Internet. In [11], [14], the authors proposed and evaluated some Dijkstra-based QoS routing algorithms, such as the *shortest-distance path* algorithm which uses the reciprocal of link bandwidth as weight function, similar to the BSP algorithm in this paper. In [15], the authors discussed path selection algorithms to support QoS routes in the context of extensions to the OSPF protocol. However, they focused on QoS routing algorithms for connections or flows.

A similar NP-hard problem, called the *Capacitated Minimum Spanning Tree* problem (CMST), is studied theoretically in [6], [7], [8], [16]. The eOPR problem differs from the CMST problem in the sense that: (1) we consider multiple destination nodes while in the CMST only one destination is taken care of; (2) instead of covering all nodes in CMST, we look for an optimal spanning tree which covers all edge nodes, some interior nodes could be included in the tree if they can help to improve the optimizing target; (3) we have different optimizing target (in CMST, the optimizing target is the minimum cost of all links in the tree).

In [3], we raised the basic OPR problem for the first time. The basic OPR problem assumed homogeneous bandwidth requirement among all nodes. In this paper, we study the eOPR problem which is much more realistic in the sense that (1) both edge nodes and interior nodes are modeled and (2) bandwidth requirements among edge nodes are heterogeneous.

VI. CONCLUSION

In this paper, we studied the Extended Optimal Premium-class Routing (eOPR) problem, which was proven to be NP-hard. An approach based on integer programming was proposed to obtain the optimal solutions for given small-scale networks. Two heuristic algorithms, WSP and BSP, were introduced and studied as efficient approximation solutions. Their performances, as well as the comparisons to the optimal solutions, were also investigated. Our results suggest that BSP

can be a good candidate algorithm in practice, because it can achieve sub-optimal solutions close to the optimal ones, while it takes significantly smaller amount of execution times.

REFERENCES

[1] S.Blake et. al., "An Architecture for Differentiated Services," *RFC 2475*, December 1998.

[2] K.Nichols, V.Jacobson, and L.Zhang, "A Two-bit Differentiated Services Architecture for the Internet," *RFC 2638*, July 1999.

[3] Jun Wang and Klara Nahrstedt, "Hop-by-Hop Routing Algorithms For Premium-class Traffic In DiffServ Networks," in *Proceedings of IEEE Infocom 2002 (to appear)*, June 2002.

[4] J.L. Sobrinho, "Algebra and algorithms for QoS path computation and hop-by-hop routing in the Inernet," in *IEEE INFOCOM 2001, Anchorage, Alaska*, April 2001, pp. 727 – 735.

[5] J. Wang, Y. Wang, and K. Nahrstedt, "Quantitative Study of Differentiated Service Model Using UltraSAN," *Tech. Report UIUCDCS-R-2001-2237, Department of Computer Science, University of Illinois at Urbana-Champaign*, July 2001.

[6] A. Amberg and W. Domschke, "Capacitated minimum spanning trees: Algorithms using intelligent search," *Combinatorial Optimization: Theory and Practice*, vol. 1, pp. 9–39, Summer.

[7] C.H. Papadimitriou, "The complexity of the capacitated tree problem," *Networks*, vol. 8, pp. 217 – 230, 1978.

[8] Leslie Hall, "Experience with a Cutting Plane Algorithm for the Capacitated Spanning Tree Problem," *INFORMS Journal on Computing*, vol. 8, no. 3, Summer 1996.

[9] B. Gavish, "Formulations and Algorithms for the Capacitated Minimal Directed Tree Problem," *Journal of the ACM*, vol. 30, pp. 118–132, 1983.

[10] "Ilog cplex," in *http://www.ilog.com/products/cplex/*.

[11] Q. Ma and P. Steenkiste, "On path selection for traffic with bandwidth guarantees," in *5th IEEE International Conference on Network Protocols, Atlanta, GA*, October 1997, pp. 191 – 202.

[12] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: An approach to universal topology generation," in *Proceedings of the International Workshop on MASCOTS '01*, Cincinnati, Ohio, August 2001.

[13] S. Chen and K. Nahrstedt, "An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions," *IEEE Network, Special Issue on Transmission and Distribution of Digital Video*, Nov./Dec. 1998.

[14] Q. Ma and P. Steenkiste, "Supporting Dynamic Inter-Class Resource Sharing: A Multi-class QoS Routing Algorithm," in *IEEE Proceedings of INFOCOM '99*, 1999, pp. 649–660.

[15] G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda, and T. Przygienda, "QoS Routing Mechanisms and OSPF Extensions," RFC 2676, Aug. 1999.

[16] R. Ahuja, "Multi-exchange neighborhood structures for the cmst problem," in *Mathematical Programming 91*, 2001, pp. 71–97.

[17] J.K. Lenstra, D. Shmoys, and E. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," in *28th IEEE FOCS*, 1987.

[18] J. Kleinberg, Y. Rabani, and E. Tardos, "Fairness in routing and load balancing," in *40th Annual Symposium on Foundations of Computer Science*, 1999, pp. 568 – 578.

APPENDIX: NP-HARDNESS OF THE eOPR PROBLEM

Actually, the eOPR problem is even harder than the original OPR problem. Therefore, following a similar reduction, it is not difficult to prove that the eOPR is also a NP-hard problem.

As we know, the *Non-uniform Load Balancing problem*, denoted by P0 for short, is NP-hard [17], [18].

P0 - *The Non-uniform Load Balancing problem*

INSTANCE: A set of jobs $J = j_1, j_2, \cdots, j_k$, and a set of machines $M = m_1, m_2, \cdots, m_n$; for each job $j_i \in J$, there is a set $S_i \subset M$ on which j_i can be run; each job j_i has a *requirement* r_i which is equal to either 1 or 2.

QUESTION: Is there an assignment from J to M such that each job $j \in J$ is assigned to a machine $m \in M$ so that the sum of the requirements assigned to each machine is at most 2?

Now we prove that eOPR is also NP-hard by constructing a reduction from P0 to eOPR.

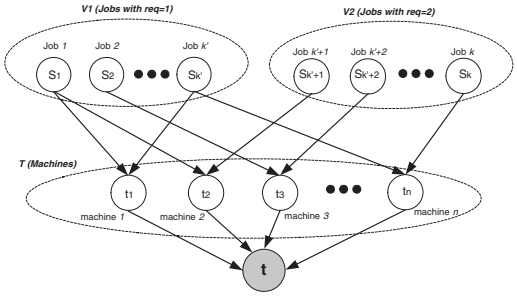


Fig. 4. The NP-hardness reduction

The reduction algorithm begins with an instance of P0. We construct a graph $G(V, E)$ to encode the instance of P0, shown in Figure 4.

For each job j_i in J, we add one node s_i into V. Since we have two types of jobs (with requirement 1 or 2), without loss of generality, we assume that there are k' jobs with requirement 1. Accordingly, we can divide s_i nodes into two groups, denoted by V_1 and V_2 respectively in Figure 4. For each machine m_i in M, we place a node t_i into V as well. We denote these nodes by T. Finally, one additional termination node, t, is added into V. So far, the construction of V is completed, that is, $V = V_1 \cup V_2 \cup T \cup \{t\}$.

For each job j_i in J, since it has a machine set $S_i \subset M$ on which it can run, we add one link (s_i, t_j) into E if machine $m_j \in S_i$. Through these links, we encode the condition that job j_i can only be assigned to a machine in S_i. Then, for each node t_i, a link (t_i, t) is also added into E. We assign all links in E with the same capacity K ($K \geq 2$). This completes our construction of graph $G(V, E)$. Now we have

$$V = \{s_i | i = 1, \cdots, k\} \cup \{t_j | j = 1, \cdots, n\} \cup \{t\} \quad (9)$$

$$E = \{(s_i, t_j) | m_j \in S_i\} \cup \{(t_j, t) | j = 1, \cdots, n\} \quad (10)$$

It is easy to verify that $|V| = k + n + 1$.

Figure 4 illustrates the construction of G. It is clear that such construction of G can be completed within polynomial time. Now the Problem P0 has been transformed into an instance of the eOPR problem, where the constructed $G(V, E)$ is the graph. On top of it, $V_e = \{s_i\} \cup \{t\}$ are all the edge nodes. The link capacities $c_{ij} = K$ ($\forall (i, j) \in E, K \geq 2$). And the requirements are given in Equation 11.

$$r_{ij} = \begin{cases} 1, & i \in V_1, j = t \\ 2, & i \in V_2, j = t \\ 0, & \text{otherwise.} \end{cases} \quad (11)$$

The question now becomes to look for a spanning tree T_t in G rooted at t which (1) covers all the other edge nodes in V_1 and V_2; and (2) satisfies the constraint that the maximum relative congestion among all links should be no larger than $\frac{2}{K}$.

We can show that this transformation of the instance of P0 into the above instance of eOPR is a reduction, because it is easy to verify that (1) a feasible allocation of jobs to machines in Problem P0 always constructs a spanning tree T_t in G; and (2) a solution T_t to the above instance of eOPR problem always implies a feasible allocation to P0 accordingly. (Due to space limitation, we omit the formal proof to this part of claims.)