A Genetic Algorithm for Smooth Scheduling in Slotted WDM Network

Jingjing Zhang, Yaohui Jin*, and Weisheng Hu

State Key Lab of Advanced Optical Communication System and Network,
Shanghai Jiao Tong University, Shanghai 200030, China

{zjj,jinyh,wshu}@sjtu.edu.cn

Abstract

In this paper we study the smooth scheduling problem in slotted wavelength division-multiplexing (WDM) ring network. In such a network, scheduling has to satisfy conflict-free constraints in both sending and receiving nodes. In this paper we consider the two important performance criteria of throughput and jitter existing in real applications. Firstly, the mathematical formulation of the smooth scheduling problem is presented, which is a combinational optimization problem with permutation property. Then the problem specific genetic algorithm is proposed to solve it. Finally, simulating results show that the proposed algorithm can achieve good performance in throughput and jitter.

Keyword: Genetic Algorithm; Smooth Scheduling; Slotted WDM

I. Introduction

Slotted wavelength-division-multiplexing (WDM) ring network, combining WDM and time-division-multiplexing (TDM) together in virtue of tunable laser with fast tuning speed, has received great attention recently [1]. First, it enjoys great potential to provide larger bandwidth in order to satisfy the requirement of explosive internet traffic. Second, the deployed ring architecture has lots of advantages such as simple routing policy, simple control and management of network resources [1]. Third, synchronization of nodes in the ring can be achieved and impact introduced by different propagation delays can be ignored adopting the scheme proposed in [2] [3].

Nowadays, real time applications in metro area network (MAN) such as video and audio applications are increasing with high speed. They are sensitive to delay variance termed as jitter. Jitter leads to undesirable burst for the receiving end, which in turn requires large buffer. The buffer size is crucial especially for optical networks because buffers are constituted by expensive optical delay lines. Large optical buffers imply high cost, power consumption and management complexity. Hence the communication system needs to guarantee a jitter bound for these applications. In this paper we directly set high throughput and low jitter as the objects of the scheduling algorithms. It's proved in the following that small delay and fairness can be indirectly achieved.

In quasi-static traffic pattern, traffic demands of the node pairs in a time period named frame are known ahead. They are described by a matrix T, where $T_{i,j}$ denotes the number of slots needed to schedule traffic from node i to j. The aim is to obtain a schedule timetable S, where $S_{i,t}$ denotes the

receiving node of sending node i in slot t. S is desired to achieve good performances in terms of jitter and throughput. Throughput is inversely proportional to the bandwidth requirement. High throughput requires short schedule table. Short overall delay is concordant with minimum bandwidth requirement since short delay is achieved for scheduling traffic demand with short schedule table. On the other hand, low jitter requires the traffic for any entry(i,j) be distributed evenly in the timetable. Even distribution for all entries implies fairness among nodes. In order to avoid the quadratic computation of delay variance, we adopt peak-to-peak delay difference to compute jitter. It's defined as the difference between maximum and minimum delay [7].

The schedule table S has two meanings of permutations. First, owning to the conflict free constraints, each column of the schedule table S is a permutation (or partial permutation) of receiving nodes. This permutation is defined as inner permutation. Second, S is constituted by a permutation of these inner permutations. It's defined as outer permutation. Throughput is determined by the total number of inner permutations. Jitter is determined by both permutations. In solving it by genetic algorithm, we move the small bandwidth object into the constraint list to reduce the two-object optimization problem into one-object problem. Then the schedule table is directly set as the chromosome code. There are three important characters of our problem. Firstly, the lengths of chromosomes vary from each other. Two crossover positions need be chosen for two individuals respectively. Secondly, crossover and mutation operation cannot break all the constraints. Adjustment is needed to make the final individuals valid. Thirdly, crossover and mutation operations should be efficient enough to modify both inner and outer permutations so as to search for fitter genes.

The rest of this paper is organized as follows. In section 2 we present the model of smooth scheduling problem in slotted WDM ring network and review related works. In section 3 we describe our genetic algorithm in terms of coding method, fitness function, crossover as well as mutation. Then, simulation results that show the performance of our designed genetic algorithm are presented in section 4. The last section 5 provides some conclusion remarks.

II. Problem Formulation

A. Smooth Scheduling Model

The following fig.1 shows slotted WDM ring network architecture. Each node is equipped with a notch filter whose wavelength is defined as the home wavelength of the node. A tunable laser switches the wavelength of the output light to the home wavelength of the destination node. Data with the same destination is encapsulated into a fixed-length packet, which consists of a sequence of data. Packets for different destination nodes are then put into different queues in the transmitter to avoid head of line blocking. In [5], a demonstration was performed on a 3-node 70-km ring network prototype carrying 10-Gb/s uniform traffic among network nodes encapsulated in 4.8-μs data packets.

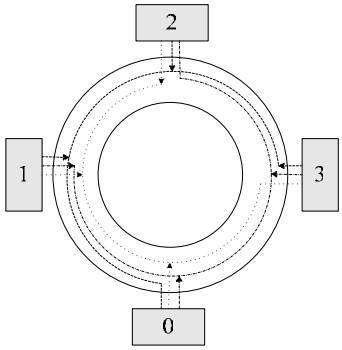

Fig. 1. Slotted WDM ring network architecture

Let T denote the traffic demand. The diagonal elements of T are zero since a node doesn't need to transmit to itself. A three dimensional matrix R is adopted to describe its schedule, where $R_{i,j,t}$ denotes whether slot t is taken by traffic from i to j. Suppose the number of nodes is N, the total slots also the bandwidth requirement used for the traffic demand is L, then $i \in [1,N]$, $j \in [1,N]$, $t \in [1,L]$. One of the objects is to achieve minimum bandwidth requirement.

$$\min (L) \tag{1}$$

Owning to the case that only one tunable transmitter and one fixed receiver is deployed for each node, conflicts appear when two or more packets are simultaneously transmitted or received, mathematically:

$$\sum_{j=1}^{N} R_{i,j,t} \le 1, \forall i \in [1,N], \forall\, t \in [1,L] \tag{2}$$

$$\sum_{i=1}^{N} R_{i,j,t} \le 1, \ \forall j \in [1,N], \forall\, t \in [1,L] \tag{3}$$

$$R_{i,j,t} \in \{0,1\}, \forall j \in [1,N], \forall j \in [1,N], \forall\, t \in [1,L] \tag{4}$$

To ensure lossless of the total traffic, R should satisfy the following constraint (5).

$$\sum_{t=1}^{\mathrm{L}} R_{i,j,t} = T_{i,j}, \ \forall i \in [1,N], \forall\ j \in [1,N] \tag{5}$$

To compute jitter, we firstly educe scheduling slots for all entries. There are totally $T_{i,j}$ schedule positions for traffic from i to j. The kth schedule slot denoted as $d_{i,j}^{k}$ is the kth t at which $R_{i,j,t}$ equal with 1 (see (6)). Then schedule interval between two adjacent schedules for traffic from i to j $interval_{i,j}^{k}$ is computed by (7).

$$d_{i,j}^{k} = t, if\ R_{i,j,t} = 1\ and\ \sum_{l=1}^{t-1} R_{i,j,t} = k-1, \forall i \in [1,N],\ j \in [1,N] \tag{6}$$

$$\mathrm{interval}_{i,j}^{k} = (\mathrm{d}_{i,j}^{k \bmod T_{i,j}+1} - \mathrm{d}_{i,j}^{k}) \bmod L, k \in [1,...T_{i,j}] \tag{7}$$

Average jitter of node pairs can be obtained using the following equation (8). As the diagonal elements in T are zero, the average operation is performed on N*(N-1) node pairs. Then the second object of smooth scheduling problem can be described by (9).

$$jitter = \sum_{i=1}^{N}\sum_{j=1}^{N} \{\max interval_{i,j} - \min interval_{i,j}\} / \left[N*(N-1)\right] \tag{8}$$

min (jitter) (9)

B. ***Related Works***

The constraints of the above scheduling issue are the same with that of input queued switch. C. S. Chang et al. proposed a minimum-bandwidth Birkhoff Von-Neumann (BV) decomposition scheme [6]. BV decomposition can guarantee minimum bandwidth requirement which is calculated using (10).

$$B_{\min} = \max\{\max(\sum_{i=1}^{N} T_{i,j}, \forall j), \max(\sum_{j=1}^{N} T_{i,j}, \forall i)\} \quad (10)$$

However, it does not take jitter into account. I. Keslassy et al. proposed a low jitter decomposition scheme in which a set of new constraints are added besides those referred in BV decomposition [7]. They try to decompose the traffic demand into a combination of orthogonal permutation matrices to reduce similarity between neighbor schedule matrices. This LJ ILP is proved NP-complete [7] [8].

Both of the above two schemes divide the scheduling issues into two steps: decomposing traffic demand into permutation matrices and arrange these decomposed matrices. The first step is to obtain inner permutation and the second step is to decide out permutation. Bandwidth requirement is determined by the first decomposition step and jitter is determined by two steps together. In order to compute jitter directly, we combine the two steps together and set jitter as the optimization object in this paper

III. Genetic Algorithm

Genetic algorithm (GA) proposed by Holland et al. in 1975 is a random search and optimization method [9][10]. It is based on natural selection theory and genetic mechanism of the living beings. In practice, GA is an iterative search process as shown in fig.2. G and M are the generation number and the maximal generation number in the evolution process. In each generation, fitter individuals survive with greater probability and weaker ones are more likely to be eliminated. As long as we adopt proper evaluation function and genetic operations the population will eventually converge to an optimal or near optimal result. We then describe each step in detail.

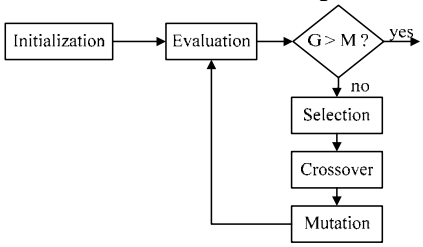

Fig. 2. Flow chart of genetic algorithm

A. ***Coding***

For the above referred three dimensional matrix *R*, we can obtain an equivalent two dimensional schedule table *S* owing to constraints (2)(3)(4).

Because of constraints (2)(4), only one *j* is assigned to $S_{i,t}$; because of constraint (3)(4), there are no two same nonzero elements in the same column. Each column of S corresponds to a permutation matrix. $S_{i,t}$ is the destination node of source node *i* in slot *t*.

$$S_{i,t} = \begin{cases} j, & if\ R_{i,j,t} = 1 \\ 0, & otherwise \end{cases} \quad (11)$$

S is the chromosome code. Each column of *S* corresponds to an allele. The length of chromosome is the number of columns in *S*. The length isn't fixed but may vary from each other. Such variation introduces inconvenience of designing genetic operators.

B. Fitness Function and Selection

There are two objects of the issue and both are of critical importance. The two-object optimization problem is reduced to one object problem by moving the minimum bandwidth object into the constraint list. Minimum bandwidth $B_{\min}$ can be derived by (10). We set an upper bound of required bandwidth in our simulations. Suppose the upper bound is $B_{\min} + \Delta B$, then $L \in [B_{\min}, B_{\min} + \Delta B]$ is acceptable (see (12)).

$$B_{\min} \le L \le B_{\min} + \Delta B \tag{12}$$

jitter(i) is the jitter performance of an individual chromosome *i*. It can be counted by (6) (7) (8) together. Individuals with low jitter are more adaptable for the environment. Thus fitness function is set to be (13).

$$Fit(i) = \frac{1}{jitter(i)} \tag{13}$$

Finally, a traditional roulette wheel selection policy [10] is employed to select chromosomes to participate in the reproduction process. Every chromosome is likely to be selected with a specific probability.

C. Initialization

The purpose of initialization is to produce individuals of the first generation which constitute the initial searching space. To diversity the original population, individuals are created randomly to enable different kinds of gene information exist in the population. The following shows the steps involved in producing one individual chromosome. The whole population is generated by repeating the process.

Step 1. randomly generate a permutation or partial permutation matrix P whose element is nonzero only if the element with the same position in traffic demand T is nonzero too. Then let P denote the scheduling of nodes in the slot.

Step 2. modify the traffic demand under the following rule:

$$\begin{cases} T_{i,j} = T_{i,j} - 1, \ if \ \ P_{i,j} = 1 \\ T_{i,j} = T_{i,j}, \ \ otherwise \end{cases} \tag{14}$$

Step 3. move on to next slot when there still exist nonzero elements in *T* and then go to step 1.

Chromosomes are randomly generated slot-by-slot in which matching output nodes for given input nodes are chosen randomly.

D. Crossover

Crossover operation produces new individuals by exchanging gene information of two parent individuals [9] [10]. Different from conventional crossover in which only a crossover point is needed, two crossover points must be generated for two parents respectively. If simply exchanging two parts of parents to generate individuals, it may generate invalid solutions

which violate the constraints. Adjustment is needed to make the individuals valid. The following describes seven steps involved:

Step 1. randomly select two crossover positions for two parents respectively.

Step 2. exchange the first part of two parents to produce two schedule tables.

Step 3. check whether schedules for a node pair equal with its traffic demand. If it equals, go to step 6; if it is larger than traffic demand, go to step 4; if it's less than traffic demand, go to step 5.

Step 4. randomly select allocations from the allocated slots for this entry, denote them as redundant allocations and cut them. Go to step 6.

Step 5. check free slots in the schedule table for the sending node. When there is no such element in the column of the slot, allocate the slot to the entry. If free spaces are not enough for the unscheduled traffic, append new slots at the end of the chromosome. Go to step 5.

Step 6. look for redundant columns in which nonzero elements can be inserted into other columns without breaking the constraints, then add them to other columns and cut them.

Step 7. move on to the next entry and go to step 3 until schedules for all node pairs have been checked.

For example, A and B are two randomly generated schedule table for the following traffic demand T. Then the steps addressed in the above crossover process are as follows:

$$T=\begin{bmatrix}0&2&1\\2&0&3\\2&2&0\end{bmatrix}\quad \mathcal{A}=\begin{bmatrix}2&0&2&3&0\\3&1&3&1&3\\1&2&1&2&0\end{bmatrix}\quad \mathcal{B}=\begin{bmatrix}3&2&2&0&0\\1&3&1&3&3\\2&1&0&1&2\end{bmatrix}$$

Firstly, we choose crossover position i=4, j=3. After exchanging the first three columns of A with the first two columns of B, we get the two new schedule table:

$$\mathcal{A}=\left[\begin{array}{ccc|cc}2&0&2&3&0\\3&1&3&1&3\\1&2&1&2&0\end{array}\right]\Longrightarrow \mathcal{A}'=\begin{bmatrix}3&2&3&0\\1&3&1&3\\2&1&2&0\end{bmatrix}$$

$$\mathcal{B}=\left[\begin{array}{cc|ccc}3&2&2&0&0\\1&3&1&3&3\\2&1&0&1&2\end{array}\right]\qquad \mathcal{B}'=\begin{bmatrix}2&0&2&2&0&0\\3&1&3&1&3&3\\1&2&1&0&1&2\end{bmatrix}$$

Next we take A' for example to explain the modification process.

$$\mathcal{B}'=\begin{bmatrix}2&0&2&2&0&0\\3&1&3&1&3&3\\1&2&1&0&1&2\end{bmatrix}\xrightarrow{\text{Eliminate}}\begin{bmatrix}2&0&2&0&0&0\\3&1&3&1&3&0\\1&2&1&0&0&2\end{bmatrix}\xrightarrow{\text{Add}}\begin{bmatrix}2&3&2&0&0&0\\3&1&3&1&3&0\\1&2&1&0&0&2\end{bmatrix}\xrightarrow{\text{Combine}}\begin{bmatrix}2&3&2&0&0\\3&1&3&1&3\\1&2&1&2&0\end{bmatrix}$$

There are three scheduling for $T_{1,2}$ in B'. We randomly choose $B_{1,4}$ to cut. The same reason goes for $B_{2,6}$ and $B_{3,5}$. While scheduling for $T_{1,3}$ isn't enough, $B_{1,2}$ is selected to be inserted by setting $B_{1,2}$ to be 3. At last, we find column 4 and column 6 can be combined. After combination, the final chromosome is obtained.

For the above crossover process, there are three main characters. Firstly, because redundant schedules for a node pair are selected randomly among the allocated slots in step 4, individuals obtained differ even if parents and crossover positions are the same. Secondly, the individual chromosome may be prolonged after step 3 and may be shorten after step 4. Thirdly, the crossover operation changes both inner and outer permutations.

E. Mutation

In the smooth scheduling problem, jitter is affected by both outer and inner permutation. However, crossover operation, the main operator of GA, isn't good enough to adjust outer

permutation. So mutation operation is desired to adjust outer permutation efficiently. In mutation, we try to change the order of these alleles to obtain new individuals. Mutation operators proposed in TSP such as inversion mutation, insertion mutation, movement mutation, exchange mutation and heuristic mutation can be employed here too. Here we adopt two-exchange operation to explore the neighbor solution. It works as follows: randomly select two alleles of the individual and exchange their positions.

IV. Numerical results and discussion

In this section, we present a study of the performance of our proposed genetic algorithm. Taking the network with 4 nodes for example, we randomly generate the following traffic matrix T.

$$T=\begin{bmatrix}0 & 3 & 9 & 7\\ 8 & 0 & 3 & 4\\ 3 & 8 & 0 & 4\\ 1 & 9 & 9 & 0\end{bmatrix}$$

ΔB referred in constraint (12) is set to be $\lceil 10\% \times B_{\min} \rceil$. The minimum bandwidth requirement $B_{\min}$ of the traffic is 21, thus constraint (12) is modified as $21 \le L \le 24$. Two-exchange operation is adopted for mutation. Crossover and mutation probability are fixed as 0.7 and 0.3. Maximal generation number is set 1000 which is the program termination condition. The following figure shows the improvement of jitter performance in one simulation.

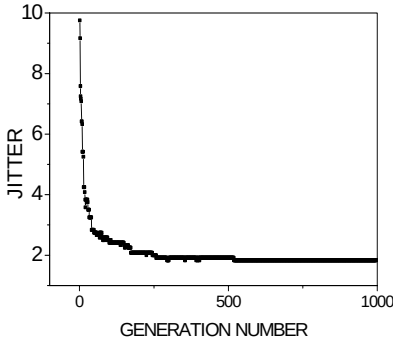

Fig. 3. Improvement of jitter over generations

It can be seen that jitter improves generation by generation on the whole though it deteriorate in several generations. Jitter gradually decreases in the first 200 generations. Between the 200th and the 528th generations, there is only a small improvement. After the 528th generation, Jitter keeps constant with a low value until 1000generations.

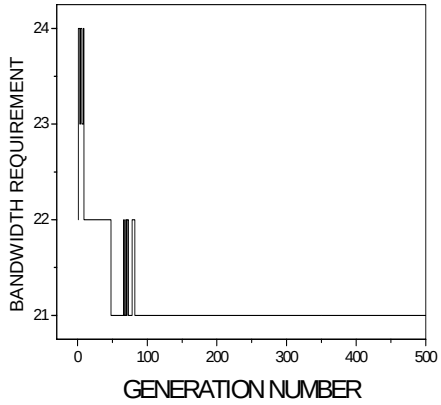

Fig. 4. Bandwidth variation over generations

Fig.4 shows the fluctuation of bandwidth requirement in 500 generations during one simulation. Bandwidth requirement decreases from the whole point of view. In the first 91 generations, the

bandwidths fluctuate between 21 and 24. It is stabilized with minimum bandwidth requirement 21 after the 91th generation. It's shown that individuals with low jitter always imply minimum bandwidth. The reason is: the shorter the length of the individual chromosome, the shorter the average scheduling intervals of entries, thus the difference between maximal scheduling interval and minimum one can be reduced. In some sense, low jitter object immanently requires small bandwidth requirement. In the following simulations, fittest individuals are all with lowest bandwidth and we will not discuss bandwidth requirement in particular.

$$S=\begin{bmatrix} 4&2&4&3&3&4&3&4&3&2&4&3&0&3&4&2&0&3&4&3&3 \\ 3&4&1&0&0&1&4&3&1&0&1&4&0&1&3&0&1&4&1&0&1 \\ 2&1&2&0&4&2&0&0&2&4&2&1&0&2&0&4&2&1&2&0&4 \\ 1&3&3&2&2&3&2&2&0&3&3&2&3&0&2&3&3&2&3&2&2 \end{bmatrix}$$

Fig. 5. Individual with the best performance in 1000 generations

The individual chromosome with the best performance among individuals in 1000 generations is shown in fig. 5. Its jitter is 1.8333. Adopting low jitter scheme proposed in [7], we get the following schedule table.

$$S=\begin{bmatrix} 4&3&0&0&0&3&4&3&2&0&0&3&4&3&2&4&3&0&4&3&2&4&3&4&3 \\ 1&0&4&3&1&0&1&0&4&3&1&0&0&0&4&1&0&3&1&0&4&1&0&1&0 \\ 2&0&1&0&2&4&2&4&0&0&2&0&0&4&1&2&0&0&2&4&1&2&0&2&0 \\ 3&2&0&0&3&2&3&2&0&0&3&2&3&2&0&3&2&1&3&2&0&3&2&3&2 \end{bmatrix}$$

Fig. 6. Individual obtained by greedy low jitter algorithm.

The length of the schedule table is 25 and jitter of it is 3.6667. Its bandwidth requirement is 19.4% more than that of GA; its jitter performance is two times of that of GA. Thus our GA can achieve much better performance in terms of throughput and jitter than greedy low jitter scheme proposed in [7].

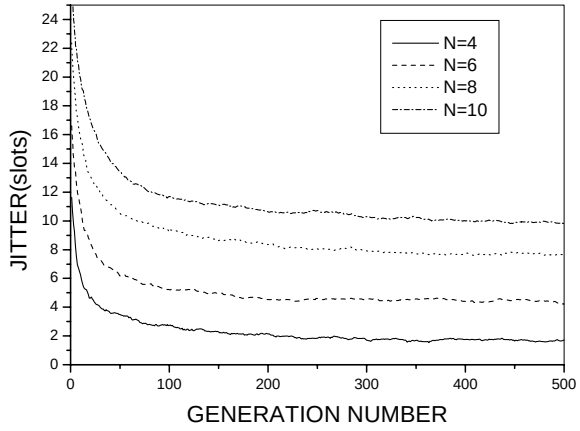

Fig. 7. Performance of different node numbers

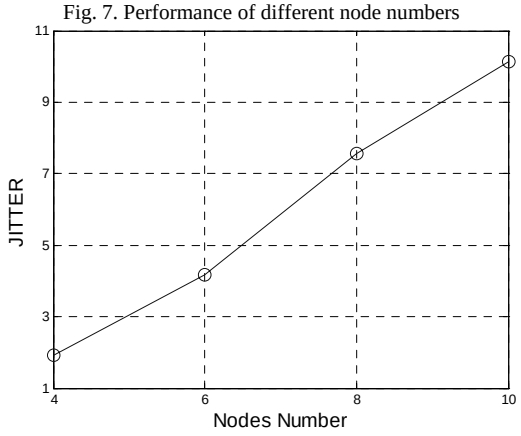

Fig. 8. Average minimum jitter

The above fig. 7 shows jitter performance of schedule table for different node number. Traffic demands are randomly generated. Individual number is chosen as 20 and maximal generation number is 500. 20 times simulations were performed. It can be seen that the larger the nodes number the more generation is needed for getting optimal result. For N=4, it's hard for jitter to improve after 200th generation, while when N=10, jitter still improves when it achieves the 500th generation. In addition, the larger the nodes number the larger the jitter. Fig. 8 shows the average minimum jitter in our 20 simulations. We can see that jitter nearly increase linearly with the nodes number.

V. Conclusion

Smooth scheduling in slotted WDM networks is a combination problem with permutation property. In this paper, we proposed a problem specific genetic algorithm to solve it. Problem specific crossover operation was designed to change inner and outer permutations of individuals. Simulation results show that GA can get good jitter performance without introducing additional bandwidth.

References

[1] Jelger, C.S., Elmirghani, J.M.H.: Photonic Packet WDM Ring Networks Architecture and Performance, IEEE communication magazine (2002) Vol. 40, No. 11, 110-115

[2] Hemenway, B.R., Castagnozzi, D.M., et al.: Autonomous Timing Determination in a Time-Slotted WDM All-Optical Network, Proceedings of Flat Panel Display Technology for Global Information Infrastructure/ICs for New Age Lightwave Communications/RF Optoelectronics (1995) 54-57

[3] Jin, Y., Su, Y., et al.: Slot Synchronization in Wavelength-routed Star Networks Based on Broadcasting Frames from a Multifrequency Laser, Proceedings of Optical Fiber Communications Conference (2003) Vol.1, 130 – 131

[4] Liew, S.Y., Chao, H.J.: On slotted WDM switching in Bufferless all-optical Network, Proceedings of High Performance Interconnects (2003) 96-101

[5] Su,Y., et al.: Experimental Demonstration of a WDM-TDM Ring Network Prototype, Lucent Technical Memo

[6] Chang, C.S., Chen, J.W., Huang, H.Y.: Birkhoff-Von Neumann Input Buffered Crossbar Switches, Proceedings of IEEE INFOCOM (2000) 1614-1623

[7] Keslassy, I., Kodialam, M., Lakshman, T.V., Stiliadis, D.: On Guaranteed Smooth Scheduling For Input-Queued Switches, Proceedings of IEEE INFOCOM (2003) Vol.2, 1384-1394

[8] Rendl, F., On the Complexity of Decomposing Matrices Arising in Satellite Communication, Operations Research Letters (1985) Vol. 4, 5-8

[9] Michalewicz, Zbigniew: Genetic Algorithms + Data Structures=Evolutionary Programs, Springer-Verlag Berlin Heidelberg press (1996)

[10] Gen, M., Cheng, R.: Genetic Algorithms and Engineering Optimization, Wiley (2000)