λ-ADJUST Scheduling Algorithm for Optical Switches with Reconfiguration Delay

Xin Li and Mounir Hamdi
Department of Computer Science
Hong Kong University of Science & Technology
Clear Water Bay, Kowloon, Hong Kong
hamdi@cs.ust.hk

***Abstract*— Using optical technology brings with it advantages such as scalability, high bandwidth and power consumption for the design of switches/routers. However, reconfiguring the optical fabric of these switches requires significant time under current technology. Conventional slot-by-slot scheduling may severely cripple the performance of these switches due to its frequent request for fabric reconfiguration. A more appropriate way is to use a Time Slot Assignment (TSA) scheduling approach. The switch gathers the incoming traffic periodically and schedules them in batches, while trying to minimize the transmission time. This Optical Switch Scheduling (OSS) problem is defined in this paper and it is NP-complete. Earlier TSA algorithms normally assume the reconfiguration delay to be either zero or infinity for simplicity. To this end, we propose a λ-ADJUST algorithm, which breaks this limitation and self-adjusts with different reconfiguration delay values. The algorithm runs at $O(\delta N^2 \log N)$ time complexity, guarantees a 100% throughput and a bounded worst case delay. In addition, it outperforms existing TSA algorithms across a large spectrum of reconfiguration values.**

I. Introduction

The explosion of Internet traffic has increased the demand for high-speed, large-port-count backbone switches. Optical switches based on 2D/3D MEMS mirrors and similar technologies are favored for their scalability, high bit rate and low power consumption. However, reconfiguring the fabric connections in optical switches involves mechanical settling, synchronization, and thus is more time-consuming than their electric counterparts. These reconfiguration overheads range from milliseconds to microseconds. It is around 10^{-1} to 10^5 slot times for a system with slotted time equals to 50ns (64 bytes at 10Gb/s).

A traditional slot-by-slot scheduling method may severely cripple the performance of optical switches because of frequent fabric reconfiguration. An appropriate solution is to use the Time Slot Assignment (TSA) approach which holds the connections for several time slots.The switch periodically accumulates incoming traffic and maps this batch to a set of switch configurations [1][2][3][4][5]. The length of each individual configuration is decided by the algorithm to achieve the shortest batch transmission time. This batch-scheduling problem (named Optical Switch Scheduling in this paper) is NP-complete for non-zero reconfiguration delays. Although extensive studies have been done in this area, to the best of our knowledge, almost all of them simply assume the reconfiguration delay is zero [1][2] or infinity [3][4] to simplify the problem and cannot fit well into medium configuration values. We propose a new scheduling algorithm λ-ADJUST which is suitable for most reconfiguration values. The algorithm introduces a regulating factor $\delta = \sqrt{T/\lambda N}$ which self-adjusts with different system parameters (T is the accumulation time, N is switch port number and λ is reconfiguration delay). According to simulations and mathematical deductions, λ-ADJUST outperforms previous approaches across a large range of reconfiguration values. It runs with $O(\delta N^2 \log N)$ time complexity, guarantees to send out traffic within 3T time slots and is stable for all admissible traffic patterns.

This work was supported in part by a grant from Hong Kong Research Grant Council (Grant Number: RGC HKUST6202-99E).

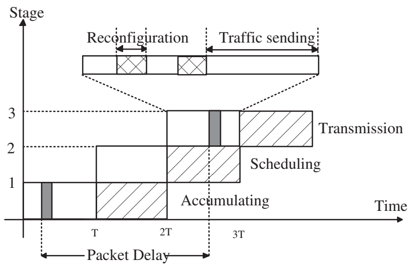

Fig. 1. Three-stage pipeline diagram

The rest of the paper is organized as follows. Section II defines the scheduling problem, reviews the previous work on the simplified form of OSS. The heuristic λ-ADJUST algorithm is described in detail in Section III. Section IV is the conclusion.

II. Optical Switch Scheduling (OSS) problem

This paper deals with the scheduling of non-blocking optical switches. To simplify the problem but still maintain its generality, an optical switch is abstracted as a crossbar-based, combined input-output queue model. Time is slotted and changing from one switch configuration to another requires λ time slots. An electronic switch can be viewed as a special case with $\lambda = 0$. A switch works in an accumulating-scheduling-transmission cycle. The length of the accumulating stage is set to be a predefined system constant T. Incoming traffic in these T time slots will be stored in traffic matrix D. Based on D, the scheduling stage determines a set of fabric configurations.

The transmission stage then follows the decision, changing the fabric between the actual traffic-sending and reconfiguration states alternatively. Transmission time equals to the summation of traffic-sending time and reconfiguration overheads. In order to send out the traffic collected in T time slots within T time slots, speedup is needed to compensate for the reconfiguration overhead λ. The operations can be further pipelined as shown in Fig. 1. The diagram indicates that a packet will go through the switch within 3T slot times (here, we assume a scheduling period equals T for simplicity). The scheduling algorithm is stable under any admissible traffic patterns and bounded worst delay (3T) makes it easy to provide QoS guarantees.

A. Definitions

Below is a list of notations used throughout the paper:

N	number of switch ports
T	traffic accumulation interval in slot times; batch size
D	cumulative traffic matrix in T time slots
λ	reconfiguration overhead in slot times
s	number of fabric configurations per batch
P	fabric configuration (0-1 matrix with at most one non-zero element at each row or column)
l	holding interval of fabric configuration (l is integer for a time slotted system)
δ	regulating factor, influenced by system parameters as λ, T and N

$D = (d_{ij})$ is a nonnegative integer $N \times N$ matrix. d_{ij} represents the number of packets received in input i whose destination is j during the accumulating stage. We define a line of matrix to be a row or a column. A line sum is the sum of all the elements in the line.

Definition 1 (Admissible traffic): If traffic matrix D stands for the packets that are accumulated during T time slots, a traffic pattern is admissible only when the line sum of D is no larger than T. That is, $\sum_{i=1}^{N} d_{ij} \le T$, and $\sum_{j=1}^{N} d_{ij} \le T$. We assume traffic to our optical switch is admissible.

Definition 2 (Matrix covering): A traffic matrix D is covered by a set of fabric configurations $P^{(1)}, \ldots, P^{(s)}$ and corresponding weights $l_1, \ldots, l_s$, if $\sum_{k=1}^{s} l_k p_{ij}^{(k)} \ge d_{ij}$, $\forall i, j \in 1, \ldots, N$. Here, $P_{ij}^{(k)}$ is the (i, j) element of configuration matrix $P^{(k)}$. In the case of equality for all i and j, the switch configurations *exactly cover* D.

Definition 3 (Covering cost): Given a traffic matrix D, and reconfiguration delay λ. If a particular set of switch configurations $P^{(1)}, \ldots, P^{(s)}$ with weights $l_1, \ldots, l_s$ covers the traffic matrix, its covering cost is defined as $cost = \sum_{k=1}^{s} l_k + s\lambda$.

Definition 4 (Optical switch scheduling (OSS) problem): Given a traffic matrix D, reconfiguration delay λ and accumulation time T; find a set of switch configurations and their respective weights, which covers the traffic matrix and minimizes the covering cost.

Input: $N \times N$ non-negative integer matrix D, positive integer λ and T, D satisfies:

$$\sum_{i=1}^{N} d_{ij} \le T, \sum_{j=1}^{N} d_{ij} \le T, \forall i, j \in 1, \ldots, N \tag{1}$$

Output: a set of configuration matrices $P^{(1)}, \ldots, P^{(s)}$ and the corresponding non-negative integer weights $l_1, \ldots, l_s$, that satisfies:

$$\sum_{k=1}^{s} l_k p_{ij}^{(k)} \ge d_{ij}, \forall i, j \in 1, \ldots, N \tag{2}$$

$$\sum_{k=1}^{s} l_k + s\lambda \text{ is minimized} \tag{3}$$

B. Simplified OSS problem

The OSS problem can be proven to be NP-complete for non-zero λ by being reduced from an existing NP-complete timetable design problem (For details of timetable design problem, please refer to [6]). Previous research normally assumes the reconfiguration time to be extreme values, such as $\lambda = 0$ or $\lambda \to \infty$, to simplify the problem.

- $\lambda = 0$. Equation (3) now changes to: minimize $\sum_{k=1}^{s} l_k$. The lower bound of $\sum_{k=1}^{s} l_k$ equals to $max(\sum_{i=1}^{N} d_{ij}, \sum_{j=1}^{N} d_{ij})$, $\forall i, j \in 1, \ldots, N$, which is the maximum line sum. Please note the OSS problem with zero configuration overhead is not NP-complete and its optimal solution is achieved by many algorithms [1][4]. As a representative, EXACT algorithm [1] meets this lower bound using the System of Distinct Representatives (SDR) method. The number configurations used is bounded by $N^2 - 2N + 2$. It has a $O(N^5)$ time complexity.
- $\lambda \to \infty$. Now $s\lambda$ occupies the major portion of covering cost. Equation (3) is satisfied only when s is first minimized. The lower bound of s is $max(max_i(r_i), max_j(c_j))$, $\forall i, j \in 1, \ldots, N$, where r_i (c_j) is the number of nonzero entries on row i (column j). The OSS problem with an infinite configuration delay is still NP-complete and only a suboptimal solution can be achieved. The representative algorithm is MIN proposed in [3]. MIN uses the least possible number of configurations and thus minimizes $s\lambda$ part. Furthermore, it tries to group the matrix entries which are roughly equal in the same switching matrix. This helps to avoid wasted time slots and hence reduces $\sum_{k=1}^{s} l_k$. MIN has a $O(N^4)$ time complexity.

III. λ-ADJUST ALGORITHM

Although assuming the reconfiguration delay to be extreme values may greatly simplify the problem, we need to deal with medium reconfiguration value in most cases. Scheduling algorithms should perform well under different reconfiguration values. An example in Fig. 2 shows two different methods for covering a traffic matrix. The strategy in Fig. 2a covers the matrix exactly with four configurations , using $12 + 4\lambda$ time slots. Its alternative in Fig. 2b uses only three configurations with cost $18+3\lambda$. The better strategy is determined by different λ values.

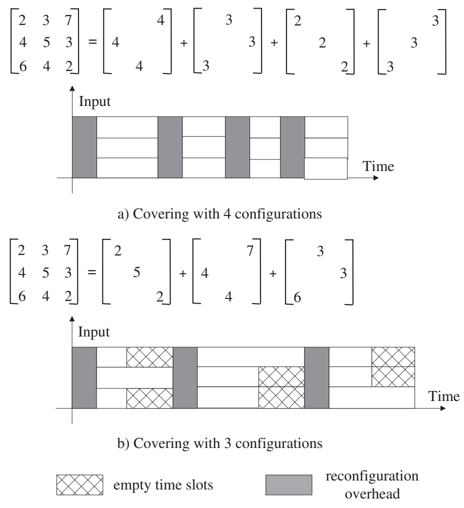
Input
Time
a) Covering with 4 configurations
Input
Time
b) Covering with 3 configurations
empty time slots
reconfiguration overhead

Fig. 2. Tradeoff between empty time slot and number of configuration matrices

A. Tradeoff between empty time slots and reconfiguration overheads

Extreme approaches, such as EXACT and MIN, perform well at their target particular cases, but they are not scalable to the other medium reconfiguration delay values. The covering cost of EXACT is bounded by $T+(N^2+2N-2)\lambda$ and grows at least with the square of the number of ports of the switch. This method is unacceptable for a system with a large port number or reconfiguration delay. MIN algorithm avoids the above problem at the cost of allowing more empty time slots. Although MIN groups similar elements together to reduce the empty time slots, it cannot fully avoid wastage. [5] shows the traffic-sending time may be as large as $\Omega(\log N)$ times of the actual needed time for 'minimizing the scheduling number' kind of algorithm.

The problem of EXACT and MIN is that they only concentrate on minimizing one of the two influencing factors: number of switchings or empty time slots, rather than finding a balance between them. To keep the summation of traffic-sending time and reconfiguration overhead minimal, a heuristic solution should carefully keep a good balance between them.

B. Algorithm description

The idea of λ-ADJUST is motivated from: 1. Uses a moderate number of configurations to achieve balance. 2. Self-adjusts to different system parameters. 3. Has low time complexity.

λ-ADJUST works by separating traffic D into *quotient* and *residue* matrices and assigning configurations to each. The algorithm generates quotient matrix A by dividing the elements in D by $T/\delta N$ and taking the floor. That is, $a_{ij} = \lfloor \frac{d_{ij}}{T/\delta N} \rfloor$, $1 \le i,j \le N$. Here δ is the regulating factor. Choosing an appropriate value for δ is crucial to the performance of the algorithm.

Lemma 1: The line sum of the quotient matrix is bounded by δN.

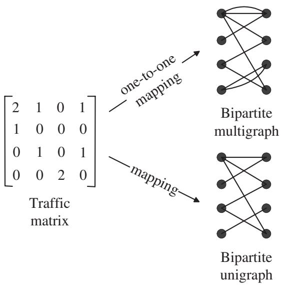

Fig. 3. One-to-one mapping between traffic matrix and bipartite multigraph

Proof: In this paper, we assume traffic is always admissible. That is, for $\forall i,j \in 1,2,\ldots,N$, $\sum_{i=1}^{N} d_{ij} \le T$, $\sum_{j=1}^{N} d_{ij} \le T$. Consider a row in quotient matrix A,

$$\sum_{j=1}^{N} a_{ij} = \sum_{j=1}^{N} \lfloor \frac{d_{ij}}{T/\delta N} \rfloor \le \lfloor \frac{\sum_{j=1}^{N} d_{ij}}{T/\delta N} \rfloor \le \lfloor \frac{T}{T/\delta N} \rfloor \le \delta N$$

Proof for columns in A follows the same way.

Lemma 2: A $N \times N$ traffic matrix D with maximum line sum S can be covered using S configuration matrices.

Proof: There exists a one-to-one mapping between this matrix-covering problem and edge-coloring problem for bipartite multigraph. The upper part of Fig. 3 shows an example of mapping between traffic matrix and bipartite multigraph. Input and output ports are mapped to be vertex sets. If $d_{ij} = \omega$ in traffic matrix D, there are ω edges between the corresponding vertices. Bipartite multigraph with maximum degree S is proven to be S-colorable. Vertices and edges belonging to a particular color can be mapped back to a switch configuration. For detailed proof of the edge-coloring, please refer to Chapter 5 of [7].

Theorem 1: Quotient matrix A can be covered using δN configuration matrices.

Proof: Comes directly from Lemma 1 and Lemma 2.

The residue matrix B is defined as $b_{ij} = max(0, d_{ij} - \lceil T/\delta N \rceil \times a_{ij})$. From its construction method, we can see $0 \le b_{ij} \le \lceil T/\delta N \rceil$. The simplest way to schedule it is to find N configuration matrices with weight $\lceil T/\delta N \rceil$ which collectively represent an all $1s$ matrix. MIN may provide better performance with higher time complexity ($O(N^4)$). Edge-coloring algorithm is another candidate and its input now is a bipartite 'unigraph' (lower part of Fig. 3) constructed from residue matrix B. All these three covering methods will use at most N configurations. Furthermore, the latter two use the minimum possible number of configurations.

By now, we can see,

$$\begin{aligned} D &= \lceil \frac{T}{\delta N} \rceil \times A + B \\ &\le \lceil \frac{T}{\delta N} \rceil \times \sum_{k=1}^{\delta N} P^{(k)} + \lceil \frac{T}{\delta N} \rceil \times \sum_{k=\delta N+1}^{\delta N+N} P^{(k)} \end{aligned} \quad (4)$$

The covering cost of quotient and residue matrix is:

$$
\begin{aligned}
cost &= \sum_{k=1}^{s} l_k + \lambda s \\
&= (\sum_{k=1}^{\delta N} \lceil \frac{T}{\delta N} \rceil + \lambda \times \delta N) + (\sum_{k=\delta N+1}^{\delta N+N} \lceil \frac{T}{\delta N} \rceil + \lambda N) \\
&= (T + \lambda N) + (\lambda \delta N + \frac{T}{\delta}) \qquad (5)
\end{aligned}
$$

The first part of (5) is a constant determined by system settings. In order to minimize (5), $\lambda\delta N + T/\delta$ needs to be minimized.

Lemma 3: Take two non-negative real numbers a and b. If $a \times b = constant$, $a + b$ is minimized when $a = b$.

Theorem 2: Covering cost of λ-ADJUST is minimized when $\delta = \sqrt{T/\lambda N}$.

Proof: Since $\lambda\delta N \times T/\delta = \lambda NT = constant$, following Lemma 3, the covering cost of λ-ADJUST is minimized when $\lambda\delta N = T/\delta$. That is to set the regulation factor δ equivalent to $\sqrt{T/\lambda N}$.

DOUBLE [5] is the special case of λ-ADJUST which always sets $\delta = 1$. Traffic matrix D is divided by T/N and generates two matrices: *coarse* matrix A $a_{ij} = \lfloor \frac{d_{ij}}{T/N} \rfloor$ and *fine* matrix B $b_{ij} = max(0, d_{ij} - \lceil T/N \rceil a_{ij})$. Both matrices can be covered using N configurations.

An example of the λ-ADJUST execution is shown in Fig. 4c. For $N = 3$, $T = 48$ and $\lambda = 1$, the regulating factor $\delta = \sqrt{T/\lambda N} = 4$. Quotient matrix A is generated by first dividing each element of traffic matrix D by $T/\delta N = 4$, and then taking the floor. For example, a_{11} is calculated by $a_{11} = \lfloor d_{11}/4 \rfloor$. The edge coloring algorithm is used to find out the configurations to cover A. Configurations for residue matrix B can be found using any algorithm listed in Step 3.

The two main operations of λ-ADJUST determine its time complexity. Dividing the traffic matrix D into quotient and residue part takes $O(N^2)$ time. The edge-coloring algorithm in Step 2 has a complexity of $O(E \log V)$ [8]. E is the number of edges and V is the number of vertices in the bipartite multigraph. For the bipartite multigraph corresponding to quotient matrix A, $V = O(N)$ and $E = O(\delta N^2)$. As a whole, λ-ADJUST has a time complexity of $O(\delta N^2 \log N)$.

Algorithm 1 λ-ADJUST algorithm

```
Input:
    $N \times N$ non-negative integer matrix D, positive integer $\lambda$ and $T$
Output:
    a set of configuration matrices $P^{(1)}, \ldots, P^{(s)}$ and the corresponding non-negative integer weights $l_1, \ldots, l_s$
Description:
1: Set the regulating factor $\delta = \sqrt{T/\lambda N}$. Split traffic matrix D into quotient matrix A and residue matrix B, such that
       $a_{ij} = \lfloor \frac{d_{ij}}{T/\delta N} \rfloor$
       $b_{ij} = max(0, d_{ij} - \lceil T/\delta N \rceil \times a_{ij})$
       $1 \le i, j \le N$
2: Schedule quotient matrix A
    1) Construct an $N \times N$ bipartite multigraph $G_A$ from A. Vertices in $G_A$ stand for switch ports. The number of edges between vertices is equal to the value of the corresponding entry in A.
    2) Find a minimal edge coloring of $G_A$. For detailed procedures, please refer to [8].
    3) Map the edge-coloring results back to switch configurations. Edges with a specific color in $G_A$ correspond to a switch configuration $P^{(i)}$. Repeat this step until no color is left.
3: Schedule residue matrix B
    1) Method a. Find any N non-overlapping switch schedules whose summation is an all 1's matrix, and set $l_i$ to be $\lceil T/\delta N \rceil$, or
    2) Method b. Use MIN algorithm
    3) Method c. Use edge-coloring algorithm. Construct an $N \times N$ bipartite unigraph $G_B$ from B. Vertices in $G_B$ stand for switch ports. There is one edge between vertex i and j if $b_{ij} \neq 0$. The following operation is similar to what is done to $G_A$.
```

C. Discussion

An example is given in Fig. 4 to further clarify the execution of MIN, DOUBLE and λ-ADJUST. In this particular example, λ is set to be relative small ($\lambda = 1$) compared to accumulating length ($T = 48$). It is not surprising that MIN has the largest covering cost due to huge number of wasted time slots. DOUBLE is not so cost effective too because the algorithm selects a partition factor $T/N = 16$, which incurs a large fine matrix at high cost (Please refer to [5] for execution details). In other words, the choice of 16 as a partition factor makes the coarse and fine matrices unbalanced. To the contrary, λ-ADJUST finds an appropriate number of reconfiguration matrices and maintains a small deviation for elements in the same reconfiguration.

Table I shows a detailed comparison between the four algorithms mentioned in this paper: EXACT, MIN, DOUBLE and λ-ADJUST. They are compared with each other on time complexity, covering cost bounds and the number of configurations used to cover the traffic matrix.

Fig. 5 shows the number of configurations DOUBLE and λ-ADJUST algorithms will use for switch with port number 32. The three almost horizontal dotted lines stand for the number of configurations DOUBLE needed to cover *coarse* matrix, *fine* matrix and the whole traffic matrix (from bottom to up respectively). Even if λ changes from a relatively small value ($0.05T$) to a large value (T), DOUBLE cannot alter its scheduling decision. In contrast, the total number of configurations that λ-ADJUST uses decreases as λ gets larger. This automatically defeats the inverse effects of increased reconfiguration overheads. Fig. 6 is the covering cost comparison between DOUBLE and λ-ADJUST algorithms. Because of its property of self-adjustment to different system parameters, λ-ADJUST saves up to 20% covering cost on average over DOUBLE for small switch port number (N=16). The saving

$$D = \begin{bmatrix} 28 & 8 & 2 \\ 4 & 20 & 16 \\ 2 & 20 & 20 \end{bmatrix} \quad \text{N=3, T=48, Delay =1}$$

a) MIN algorithm

$$\begin{bmatrix} 28 & 8 & 2 \\ 4 & 20 & 16 \\ 2 & 20 & 20 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 2 \\ 0 & 20 & 0 \\ 2 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 8 & 0 \\ 4 & 0 & 0 \\ 0 & 0 & 20 \end{bmatrix} + \begin{bmatrix} 28 & 0 & 0 \\ 0 & 0 & 16 \\ 0 & 20 & 0 \end{bmatrix}$$

s=3, Traffic-sending=20+20+28=68, Total transmission cost=71

b) DOUBLE algorithm

$$\begin{bmatrix} 28 & 8 & 2 \\ 4 & 20 & 16 \\ 2 & 20 & 20 \end{bmatrix} = 16 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 12 & 8 & 2 \\ 4 & 4 & 0 \\ 2 & 4 & 4 \end{bmatrix}$$

$$= 16 \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \right) + \left(\begin{bmatrix} 12 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix} + \begin{bmatrix} 0 & 8 & 0 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 2 \\ 4 & 0 & 0 \\ 0 & 4 & 0 \end{bmatrix} \right)$$

s=2+3, Traffic-sending=16x2+12+8+4=56, Total transmission cost=61

c) ADJUST algorithm

$$\begin{bmatrix} 28 & 8 & 2 \\ 4 & 20 & 16 \\ 2 & 20 & 20 \end{bmatrix} = 4 \begin{bmatrix} 7 & 2 & 0 \\ 1 & 5 & 4 \\ 0 & 5 & 5 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

$$= 4 \left(\begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix} + \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 4 \\ 0 & 5 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 2 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) + \begin{bmatrix} 0 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

s=3+1, Traffic-sending=4x(5+5+2)+2=50, Total transmission cost=54

Fig. 4. Example of execution of MIN, DOUBLE and λ-ADJUST

TABLE I

Comparison between EXACT, MIN, DOUBLE and λ-ADJUST

	Number configurations	Covering cost bound	Time complexity
EXACT	$O(N^2 - 2N + 2)$	$T + (N^2 - 2N + 2)\lambda$	$O(N^5)$
MIN	$O(N)$	$\sum_{k=1}^{s} l_k + N\lambda$	$O(N^4)$
DOUBLE	$2N$	$2T + 2N\lambda$	$O(N^2 \log N)$
λ - ADJUST	$(\sqrt{T/\lambda N} + 1)N$	$T + \lambda N + 2\sqrt{\lambda T N}$	$O(\delta N^2 \log N)$

effect increases for larger switches, i.e. 50% for 32×32 switches, and longer reconfiguration times.

Let us further consider the performance of λ-ADJUST under the marginal overhead values as $\lambda \to 0$ and $\lambda \to \infty$. When $\lambda \to 0$, $\delta = \sqrt{T/\lambda N} \to \infty$, $T/\delta N \to 0$, which implies traffic matrix D is divided as $D = D + 0$. Now λ-ADJUST equals to using only Step 2 (edge-coloring) to find a covering for the traffic matrix. It can achieve the lower bound of $\sum_{k=1}^{s} l_k$ (which is the same result as EXACT), but may use a slightly more configurations. The result is satisfactory compared to the reduction of time complexity from $O(N^4)$ to $O(\delta N^2 \log N)$. If $\lambda \to \infty$, λ-ADJUST puts traffic matrix solely into residual matrix. Step 3 determines its performance. Choices can be made based on complexity and performance requirements.

IV. Conclusion

Along with the fast development of the Internet, optical switching technologies are becoming attractive for their huge capacity and scalability. However, the existence of reconfiguration delay makes the Optical Switch Scheduling (OSS) problem to be NP-complete. A good heuristic algorithm should find a balance between the number of configurations and empty time slots over all possible delay values. λ-ADJUST algorithm dynamically suits with system parameters and achieves good results. Mathematical deductions and simulations show that it outperforms the previous algorithm over a large range of configuration delay values. In addition, the algorithm is stable and provides bounded delay guarantee with small time complexity.

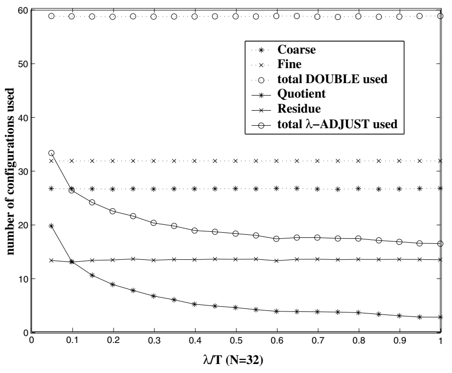

Fig. 5. Number of configurations DOUBLE and λ-ADJUST used to cover matrices for 32×32 switch

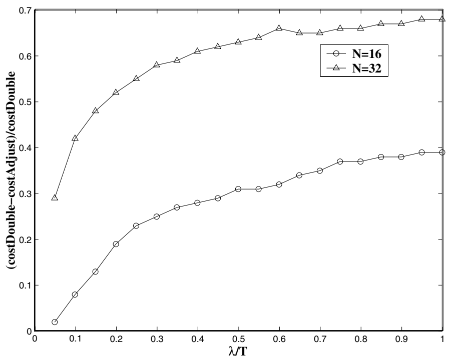

Fig. 6. Cost difference between DOUBLE and λ-ADJUST. The value of λ is set to be from $0.05T$ to T. Y axis equals to $(Cost_{DOUBLE} - CostADJUST)/(Cost_{DOUBLE})$.

References

[1] T. Inukai, "An efficient SS/TDMA time slot assignment algorithm," *IEEE Trans. Commun.*, vol. 27, pp. 1449–1455, Oct. 1979.

[2] G. Bongiovanni, D. Coppersmith, and C. K. Wong, "An optimum time slot assignment algorithm for an SS/TDMA system with variable number of transponders," *IEEE Trans. Commun.*, vol. 29, pp. 721–726, May 1981.

[3] I. S. Gopal and C. K. Wong, "Minimizing the number of switchings in an SS/TDMA system," *IEEE Trans. Commun.*, vol. 33, pp. 497–501, June 1985.

[4] M. Chen and T. S. Yum, "A conflict-free protocol for optical wdma networks," in *Proc. IEEE Globe Communications Conference (GLOBECOM'91)*, Phoenix, Arizona, Dec. 1991, pp. 1276–1281.

[5] B. Towles and W. J. Dally, "Guaranteed scheduling for switches with configuration overhead," in *Proc. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02)*, New York, June 2002, pp. 342–351.

[6] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM Journal on Comput.*, vol. 5, pp. 691–703, Dec. 1976.

[7] R. Diestel, *Graph Theory, 2nd Edition*. New York: Springer, 2000.

[8] R. Cole and J. Hopcroft, "On edge coloring bipartite graphs," *SIAM Journal on Comput.*, vol. 11, pp. 540–546, Aug. 1982.