Minimum Delay Scheduling for Performance Guaranteed Switches with Optical Fabrics

Bin Wu, Kwan L. Yeung, Pin-Han Ho and Xiaohong Jiang

Abstract—**We consider traffic scheduling in performance guaranteed switches with optical fabrics to ensure 100% throughput and bounded packet delay. Each switch reconfiguration consumes a constant period of time called *reconfiguration overhead*, during which no packet can be transmitted across the switch. To minimize the packet delay bound for an arbitrary traffic matrix, the number of switch configurations in the schedule should be no larger than the switch size *N*. This is called *minimum delay scheduling*, where the ideal minimum packet delay bound is determined solely by the total overhead of the *N* switch reconfigurations. A speedup in the switch determines the actual packet delay bound, which decreases towards the ideal bound as the speedup increases. Our objective is to minimize the required speedup S_{schedule} under a given actual packet delay bound. We propose a novel minimum delay scheduling algorithm QLEF (Quasi Largest-Entry-First) to solve this problem. Compared with the existing minimum delay scheduling algorithms MIN and α^i-SCALE, QLEF dramatically cuts down the required S_{schedule} bound. For example, QLEF only requires S_{schedule}=17.89 for *N*=450, whereas MIN and α^i-SCALE require S_{schedule}=37.13 and 27.82 respectively. This gives a significant performance gain of 52% over MIN and 36% over α^i-SCALE.**

Index Terms—**Optical switch, performance guaranteed switching, reconfiguration overhead, scheduling, speedup.**

I. Introduction

RECENT progress on optical switching technologies [1-4] has enabled the implementation of high-speed scalable switches with optical switch fabrics, as shown in Fig. 1. These switches can efficiently provide huge switching capacity as demanded by the backbone routers in the Internet. Since the input/output modules are connected to the central switch fabric by optical fibers, they can be distributed over several standard telecommunication racks. This reduces the power consumption for each rack, and makes the resulting switch architecture more scalable.

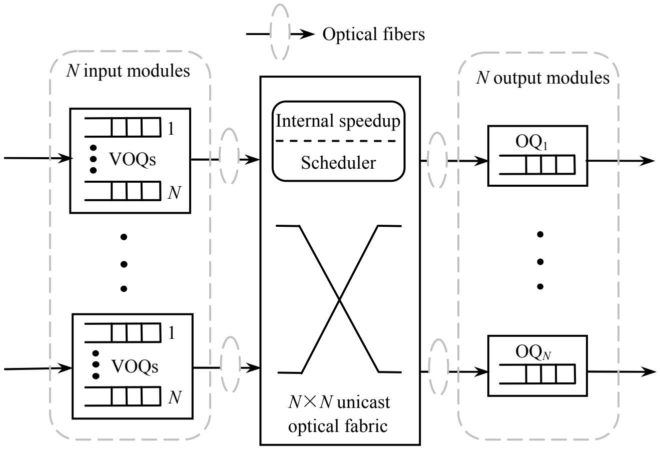


Fig. 1. A scalable switch with an optical switch fabric.

On the other hand, optical switch fabric usually needs a non-negligible amount of time to change its switch configuration. This *reconfiguration overhead* is due to three factors [5]. First, the optical switch fabric needs time to change its interconnection pattern, and this time varies from 10 ns to several milliseconds depending on the switching technology adopted [1-4]. Second, time (10 to 20 ns or more [5]) is required to resynchronize the optical transceivers and the switch fabric. Finally, because optical signals may arrive at their corresponding input ports at different times, time is also needed to align the clock in order to avoid data loss.

During the reconfiguration period, no packet can be transmitted across the switch fabric (i.e. tune-transmit separability constraint [6-7]). To achieve *performance guaranteed switching* [8-11] (i.e., 100% throughput with bounded packet delay), the switch fabric has to transmit packets at an internal speed higher than the external line-rate, resulting in a *speedup*. The amount of speedup *S* is defined as the ratio of the internal packet transmission rate to the external line-rate.

Assume each switch reconfiguration takes an overhead of δ slots and each slot can accommodate one packet. Conventional slot-by-slot scheduling methods may severely cripple the performance of optical switches due to frequent reconfigurations. Hence, the reconfiguration frequency needs to be reduced by holding each configuration for multiple time slots. Time slot assignment (TSA) [8-11] is a common approach to achieve this, where a switch works in a pipelined four-stage cycle: traffic accumulation, scheduling, switching and transmission, as shown in Fig. 2. Stage 1 is for traffic accumulation. A traffic matrix $\boldsymbol{C}(\boldsymbol{T})=\{c_{ij}\}$ is obtained at the

This paper was presented in part at the IEEE GlobeCom '05, St. Louis, MO, Dec. 2005, and the IEEE GlobeCom '06, San Francisco, CA, Dec. 2006.

Bin Wu and Kwan L. Yeung are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam, Hong Kong (Tel: 852-2857-8493; Fax: 852-2559-8738; e-mail: {binwu, kyeung}@ eee.hku.hk).

Pin-Han Ho is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, N2L 3G1 (Tel: 1-519-888 -4567 ex. 32452; Fax: 1-519-746-3077; e-mail: pinhan@bbcr.uwaterloo.ca).

Xiaohong Jiang is with the Department of Computer Science, Graduate School of Information Science, Tohoku University, Aramaki Sendai 980-8579, Japan (e-mail: jiang@ecei.tohoku.ac.jp).

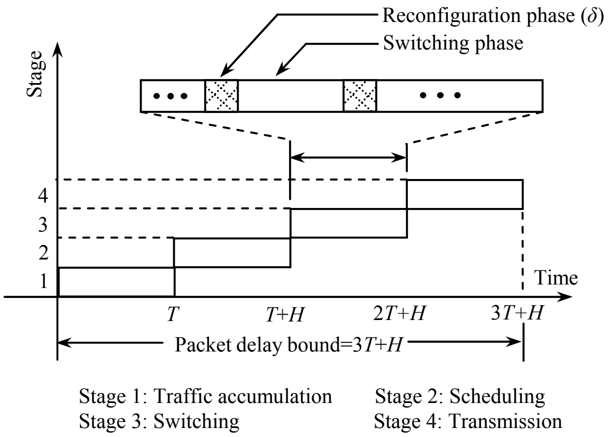


Fig. 2. Timing diagram for packet switching.

input buffers every T time slots. Each entry c_{ij} denotes the number of packets arrived at input i and destined to output j. Assume the traffic has been regulated to be *admissible* before entering the switch, i.e., the entries in each row or column of $\boldsymbol{C(T)}$ (defined as a *line* of $\boldsymbol{C(T)}$) sum to at most T. In Stage 2, a scheduling algorithm computes a schedule consisting of at most N_S configurations in H time slots. Each configuration is denoted by a permutation matrix $\boldsymbol{P_n}=\{p^{(n)}_{ij}\}$ ($N_S\geq n\geq 1$). If $p^{(n)}_{ij}=1$, input i is connected to output j, and we say that $\boldsymbol{P_n}$ covers entry c_{ij}. A weight ϕ_n is assigned to each $\boldsymbol{P_n}$, indicating the number of slots that $\boldsymbol{P_n}$ should be kept for packet switching in Stage 3. To ensure 100% throughput, the set of N_S configurations must *cover* $\boldsymbol{C(T)}$, i.e., $\sum^{N_S}_{n=1}\phi_n p^{(n)}_{ij}\geq c_{ij}$ for any $i, j\in\{0, \ldots, N-1\}$.

Packet switching takes place in Stage 3, where the switch fabric is reconfigured according to the N_S configurations obtained in Stage 2. Under a speedup S, the duration of a time slot is shortened by S times. A shortened slot (with $1/S$ duration of a regular slot) is called a *compressed slot*. The total holding time of the N_S switch configurations is $\sum^{N_S}_{n=1}\phi_n$ compressed slots, or $\sum^{N_S}_{n=1}\phi_n/S$ regular slots. Since speedup cannot reduce the reconfiguration overhead, the total overhead for the N_S reconfigurations is δN_S regular slots. Combining reconfiguration overhead and configuration holding time, Stage 3 requires $\delta N_S + \sum^{N_S}_{n=1}\phi_n/S$ regular slots. To ensure 100% throughput, the speedup S must satisfy

$$\delta N_S+\frac{1}{S}\sum_{n=1}^{N_S}\phi_n\leq T\,. \tag{1}$$

Rearranging (1), we have the minimum required speedup as

$$S=\frac{1}{T-\delta N_S}\sum_{n=1}^{N_S}\phi_n=S_{\text{reconfigure}}\times S_{\text{schedule}} \tag{2}$$

where

$$S_{\text{reconfigure}}=\frac{T}{T-\delta N_S}\,, \tag{3}$$

$$S_{\text{schedule}}=\frac{1}{T}\sum_{n=1}^{N_S}\phi_n\,. \tag{4}$$

We can see that the overall speedup S consists of two factors $S_{\text{reconfigure}}$ and S_{schedule}. In particular, $S_{\text{reconfigure}}$ compensates for hardware inefficiency caused by the N_S reconfigurations, and S_{schedule} compensates for algorithmic inefficiency caused by the bandwidth loss. During the holding time of a configuration (which is determined by the scheduling algorithm), some input-output connections become idle (earlier than others) if their scheduled backlog packets are all sent. As a result, the schedule will contain empty slots and this causes bandwidth loss, or algorithmic inefficiency.

Stage 4 takes another T slots to dispatch packets from the output buffers to the output lines in the first-in-first-out (FIFO) order. Without loss of generality, we define a flow as a series of packets coming from the same input port and destined to the same output port of the switch. Since packets in each flow follow FIFO order in passing through the switch, there is no packet out-of-order problem within the same flow. (But packets in different flows may interleave at the output buffers.)

Consider a packet arrived at the input buffer in the first slot of Stage 1. It suffers the worst-case delay of T slots in Stage 1 for traffic accumulation, and another delay of H slots in Stage 2 for algorithm execution. In the worst case, this packet will experience a maximum delay of $2T$ slots in Stages 3 & 4 (assume it is sent onto the output line in the last slot of Stage 4). Taking all the four stages into account, the delay experienced by any packet at the switch is bounded by $3T+H$ slots as shown in Fig. 2. Assume that the algorithm execution time H is a constant. The packet delay bound $3T+H$ is dominated by the traffic accumulation time T. According to (1), T depends on the number of configurations in the schedule (i.e. N_S), the speedup S, and the efficiency of the scheduling algorithm (which determines $\sum^{N_S}_{n=1}\phi_n$).

To minimize the packet delay bound $3T+H$, we first consider an ideal case where an infinite speedup can be deployed in the switch. Then, $\sum^{N_S}_{n=1}\phi_n/S$ in (1) can be ignored and the packet delay bound is $3T+H=3\delta N_S+H$. Therefore, a smaller N_S will lead to a lower packet delay bound. For performance guaranteed switching with an arbitrary traffic matrix $\boldsymbol{C(T)}$, N_S must be no less than the switch size N. This is because each configuration can cover at most N entries, and the N^2 entries in $\boldsymbol{C(T)}$ must be covered by at least N configurations [8, 10]. Consequently, the ideal minimum packet delay bound is $3\delta N+H$, which can only be obtained with the minimum number of $N_S=N$ configurations in the schedule. Accordingly, if a scheduling algorithm always ensures at most $N_S=N$ configurations in the schedule, we call it a *minimum delay scheduling* algorithm. In contrast, if an algorithm requires $N_S>N$ configurations (in the worst-case), the packet delay bound $3\delta N_S+H$ must be larger than the ideal bound $3\delta N+H$, even if the speedup is infinite.

Though an infinite speedup is infeasible in practice, the actual packet delay bound in minimum delay scheduling can be made as close to the ideal bound $3\delta N+H$ as possible if the speedup is large enough. On the other hand, a large speedup means a high implementation cost of the switch. Therefore, a key issue is to minimize the required speedup for a given actual packet delay bound (or equivalently a given traffic accumulation time T). Under the requirement of N_S=N, this translates to minimizing $\sum_{n=1}^{N_S}\phi_n$ in (1) or S_{schedule} in (4), because δ, T and N are given parameters and thus $S_{\text{reconfigure}}$ in (2)-(3) becomes a constant.

In essence, the above minimum delay scheduling problem is equivalent to a matrix decomposition problem with the constraint N_S=N, where an N×N traffic matrix $\boldsymbol{C(T)}$ is decomposed into N_S=N configurations and the sum of the N weights is minimized. In fact, traffic matrix decomposition is a classic problem [8-17]. Algorithms based on Hall's theorem [14] or Birkhoff-von Neumann decomposition [12-13, 15-17] generate a large number of configurations (e.g. N_S=N^2−2N+2 in Birkhoff-von Neumann decomposition), and thus are only favorable in scheduling problems without reconfiguration overhead. For scheduling problems with reconfiguration overhead, greedy algorithms such as LIST [7-8, 18] and decompositions based on graph theory [7-9] are invented with a smaller number of configurations in the schedule. The impact of reconfiguration overhead on the scheduling performance is also studied in [19]. A common objective of those works is to minimize the schedule length which includes time for both reconfigurations and packet transmission. Minimum delay scheduling in this paper differs from those works by ensuring N_S=N, with the objective of minimizing speedup under a given packet delay bound. We also notice that algorithm GOPAL [20] ensures N_S=N, but it is designed for average performance instead of worst-case performance guarantee.

To the best of our knowledge, there are only two existing minimum delay scheduling algorithms MIN [8] and α^i-SCALE [10] that can achieve performance guaranteed switching. Though α^i-SCALE generally outperforms MIN, its S_{schedule} bound is still too high. In this paper, a novel minimum delay scheduling algorithm QLEF (Quasi Largest-Entry-First) is proposed. Compared with MIN and α^i-SCALE, QLEF requires the lowest S_{schedule} bound. For example, QLEF cuts down the S_{schedule} bound by 52% over MIN and 36% over α^i-SCALE for a switch with size N=450. The rest of the paper is organized as follows. In Section II, QLEF is proposed. In Section III, we derive the speedup bound for QLEF. The paper is concluded in Section IV.

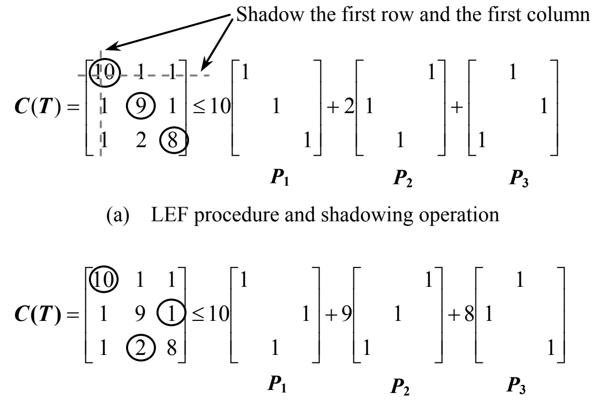
II. QLEF Algorithm

A. LEF (Largest-Entry-First) Procedure

In minimum delay scheduling, we need to find N configurations $\boldsymbol{P_n}$, $n\in\{1, …, N\}$ and the corresponding weights ϕ_n to cover an arbitrary admissible traffic matrix $\boldsymbol{C(T)}$. From (2)-(4), minimizing speedup S (or S_{schedule}) is equivalent to minimizing the sum of the N weights $\sum_{n=1}^{N}\phi_n$. Intuitively, each configuration can be constructed by always covering N largest not-yet-covered entries, each from a distinct row and column of $\boldsymbol{C(T)}$. In this way, those N entries can share the same large weight. This potentially saves the weights of subsequently constructed configurations. So our basic idea is to always cover the "largest" entry first.

Specifically, the N configurations $\boldsymbol{P_n}$={$p^{(n)}_{ij}$}, $n\in\{1, …, N\}$ are constructed one by one as follows. Large entries in $\boldsymbol{C(T)}$ are first considered. If an entry $c_{ij}\in\boldsymbol{C(T)}$ is covered by a particular configuration $\boldsymbol{P_n}$, we set $p^{(n)}_{ij}$ to 1 in $\boldsymbol{P_n}$ and c_{ij} to 0 in $\boldsymbol{C(T)}$. For simplicity, we still use $\boldsymbol{C(T)}$ to denote the updated traffic matrix, though some of its entries may have been set to 0. Each $\boldsymbol{P_n}$ can cover only one entry in each row and column of $\boldsymbol{C(T)}$. To avoid covering multiple entries in the same line of $\boldsymbol{C(T)}$ by the same configuration, we define a *shadowing* operation. Particularly, if a "largest" entry c_{ij} is to be covered by $\boldsymbol{P_n}$, we use two dashed lines to shadow row i and column j of $\boldsymbol{C(T)}$. The next "largest" entry can only be selected in the remaining not-yet-shadowed part. As an example, the largest entry c_{00}=10 in Fig. 3a is first selected and is covered by $\boldsymbol{P_1}$. The first row and the first column of $\boldsymbol{C(T)}$ are shadowed. $p^{(1)}_{00}$ is set to 1 and c_{00} is set to 0. After that, c_{11}=9 is selected as the largest entry in the remaining not-yet-shadowed part of $\boldsymbol{C(T)}$. We repeat the above steps until N large entries are selected. At this point, all the lines of $\boldsymbol{C(T)}$ are shadowed and the construction of $\boldsymbol{P_n}$ completes. Then, we *un-shadow* $\boldsymbol{C(T)}$ by removing all the dashed lines, and repeat the above process to construct the next configuration $\boldsymbol{P_{n+1}}$. We call this procedure LEF (*Largest-Entry-First*).

Fig. 3 shows two possible schedules for the same 3×3 $\boldsymbol{C(T)}$. The schedule in Fig. 3a is obtained using the above LEF procedure. Entries c_{00}=10, c_{11}=9 and c_{22}=8 are covered by $\boldsymbol{P_1}$ with a large weight of 10. The remaining entries are covered by $\boldsymbol{P_2}$ and $\boldsymbol{P_3}$ with small weights 2 and 1, respectively. The sum of the three weights is 13. The schedule in Fig. 3b is generated by some non-LEF procedure. Entries c_{00}=10, c_{21}= 2 and c_{12}= 1 are



(a) LEF procedure and shadowing operation

(b) non-LEF procedure

Fig. 3. "Largest-Entry-First" and "shadow".

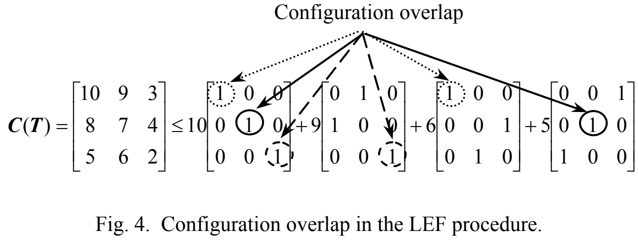


Fig. 4. Configuration overlap in the LEF procedure.

covered by $\boldsymbol{P}_1$, which requires a large weight of 10. As a result, the not-yet-covered large entries c_{11}=9 and c_{22}=8 may become the weights for $\boldsymbol{P}_2$ and $\boldsymbol{P}_3$. Compared with the LEF procedure, this non-LEF procedure gives a much larger weight sum of 27.

Unfortunately, the above LEF procedure is unlike MIN [8] and α^i-SCALE [10] which can ensure that N configurations are always sufficient to cover all the N^2 entries in $\boldsymbol{C(T)}$. This is because LEF cannot prevent *configuration overlaps*, i.e., several configurations may cover the same entry. An example is shown in Fig. 4, where the schedule returned by the LEF procedure consists of four configurations instead of the minimum three. The entries that are covered more than once (i.e., configuration overlaps) are circled in the figure.

B. QLEF Algorithm

QLEF algorithm is summarized in Fig. 5. It is designed to rectify the configuration overlap problem of LEF. This qualifies QLEF as a minimum delay scheduling algorithm. Specifically, we use an *N*×*N reference matrix* $\boldsymbol{R}=\{r_{ij}\}$ to record the status of each entry $c_{ij}\in\boldsymbol{C(T)}$. If c_{ij} is not-yet-covered, then r_{ij}=1; otherwise r_{ij}=0. $\boldsymbol{R}$ is initialized to an all-1 matrix. Assume that the N configurations are sequentially constructed from $\boldsymbol{P}_1$ to $\boldsymbol{P}_N$. When a configuration is obtained, the corresponding entries in both $\boldsymbol{C(T)}$ and $\boldsymbol{R}$ are set to 0. The updated $\boldsymbol{C(T)}$ and $\boldsymbol{R}$ are then used to construct the next configuration.

Without loss of generality, we first focus on the construction of the first half configurations $\boldsymbol{P}_{n+1}$, $0\leq n<\lceil N/2\rceil-1$. Assume that both $\boldsymbol{C(T)}$ and $\boldsymbol{R}$ are updated and un-shadowed. The construction process of $\boldsymbol{P}_{n+1}$ is similar to the LEF procedure, but we only select $N-(2n+1)$ largest entries from the not-yet-shadowed part of $\boldsymbol{C(T)}$ (instead of selecting N entries as in LEF). We call them *selected-entries*. For each selected-entry, the corresponding entry values in both $\boldsymbol{C(T)}$ and $\boldsymbol{R}$ are set to 0. At the same time, the row and column it resides in are shadowed in both $\boldsymbol{C(T)}$ and $\boldsymbol{R}$. The remaining not-yet-shadowed part of $\boldsymbol{R}$ is a $(2n+1)\times(2n+1)$ matrix defined as $\boldsymbol{U}$ (see Fig. 6). We then construct a bipartite graph [21-23] $\boldsymbol{U_G}$ from $\boldsymbol{U}$ (see the example in Fig. 7). Note that a "0" in $\boldsymbol{U}$ corresponds to a covered entry which will not become an edge in $\boldsymbol{U_G}$. Then, we find a *perfect matching* [8] in $\boldsymbol{U_G}$ by performing maximum-size matching (MSM) [22] (to be proved later). As defined in Fig. 7, a perfect matching is a subset of edges where each vertex is incident on exactly one edge in that subset. So, the perfect matching obtained in $\boldsymbol{U_G}$ contains $(2n+1)$ edges. It corresponds to $(2n+1)$ not-yet-covered entries, called *MSM-entries*. We also set these entries to 0 in both $\boldsymbol{C(T)}$ and $\boldsymbol{R}$. Combining the $(2n+1)$ MSM-entries with the $N-(2n+1)$ selected-entries, we get N entries, each in a distinct row and column. $\boldsymbol{P}_{n+1}$ is obtained by setting those N entries to 1 and all other entries to 0.

QLEF ALGORITHM

Input:

An $N\times N$ matrix $\boldsymbol{C(T)}$ with maximum line sum not more than T.

Output:

$N\times N$ configurations $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_N$ and weights $\phi_1, \ldots, \phi_N$.

Step 1: Initialization:

Set 0→n. Initialize $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_N$ to all-zero matrices and an $N\times N$ reference matrix $\boldsymbol{R}=\{r_{ij}\}$ to all 1*s*.

Step 2: Construct the first "half" configurations $\boldsymbol{P}_{n+1}$:

a) Un-shadow $\boldsymbol{C(T)}$ and $\boldsymbol{R}$. Set 1→w.

b) Select the largest entry c_{ij} (with the corresponding r_{ij}=1 in $\boldsymbol{R}$) from the not-yet-shadowed part of $\boldsymbol{C(T)}$. If w=1, set 0→w and $c_{ij}\rightarrow\phi_{n+1}$ where ϕ_{n+1} is the weight of $\boldsymbol{P}_{n+1}$. Shadow row i and column j in both $\boldsymbol{C(T)}$ and $\boldsymbol{R}$, and set c_{ij} and r_{ij} to 0. Set $1\rightarrow p^{(n+1)}_{ij}$ where $p^{(n+1)}_{ij}$ is the entry (i, j) of $\boldsymbol{P}_{n+1}$. Repeat this step until $N-(2n+1)$ entries are selected.

c) Construct a bipartite graph $\boldsymbol{U_G}$ from the remaining not-yet-shadowed part of $\boldsymbol{R}$ and perform maximum-size matching in $\boldsymbol{U_G}$ to get $(2n+1)$ edges. Record their corresponding entries (i, j) to $\boldsymbol{P}_{n+1}$ by setting $1\rightarrow p^{(n+1)}_{ij}$. Set these entries to 0 in both $\boldsymbol{C(T)}$ and $\boldsymbol{R}$. Then set n+1→n.

d) Repeat *Step 2a-2c* until $n=\lceil N/2\rceil-1$.

Step 3: Construct the second "half" configurations:

a) Un-shadow $\boldsymbol{C(T)}$ and $\boldsymbol{R}$. Find the largest entry c_{ij} in the updated $\boldsymbol{C(T)}$ and set c_{ij} as the weight for all the subsequent configurations.

b) Find a maximum-size matching in the bipartite graph of $\boldsymbol{R}$ and set the corresponding entries of $\boldsymbol{P}_{n+1}$ to 1. Set these entries to 0 in $\boldsymbol{C(T)}$ and $\boldsymbol{R}$, and then set n+1→n. Repeat this step until n=N.

Fig. 5. QLEF algorithm.

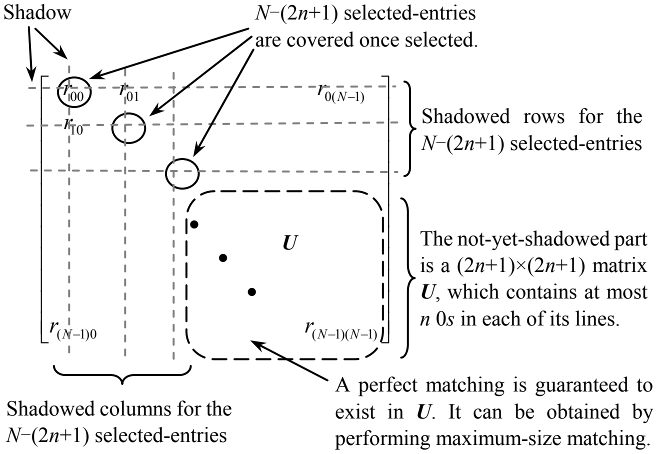

Shadow
N−(2n+1) selected-entries are covered once selected.
r00
r01
r0(N−1)
r10
Shadowed rows for the N−(2n+1) selected-entries
U
The not-yet-shadowed part is a (2n+1)×(2n+1) matrix U, which contains at most n 0s in each of its lines.
r(N−1)0
r(N−1)(N−1)
Shadowed columns for the N−(2n+1) selected-entries
A perfect matching is guaranteed to exist in U. It can be obtained by performing maximum-size matching.

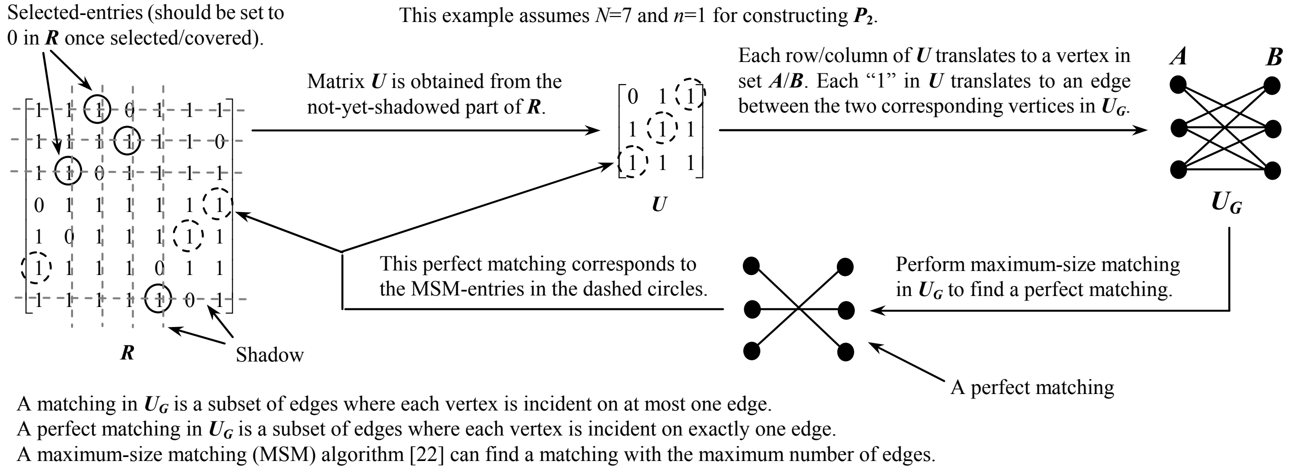
Fig. 6. Reference matrix $\boldsymbol{R}$ in QLEF algorithm.

In the above procedure, the $N-(2n+1)$ selected-entries can always be properly chosen from the not-yet-covered entries in $\boldsymbol{C(T)}$, as explained below. In constructing $\boldsymbol{P}_{n+1}$, as a result of constructing the previous n configurations $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n$, each line of $\boldsymbol{C(T)}$ has $N-n$ not-yet-covered entries, and n covered entries (denoted by 0 in both $\boldsymbol{C(T)}$ and $\boldsymbol{R}$). Here we only need to select $N-(2n+1)$ not-yet-covered entries (each from a distinct row and column). By referring to $\boldsymbol{R}$, the success of this selection is ensured because $N-(2n+1)<N-n$.

Next we prove that a perfect matching containing $(2n+1)$ edges always exists in $\boldsymbol{U_G}$. This is ensured by the following Theorem (taken from Theorem 7 of [8]).

Theorem: For a bipartite graph $\boldsymbol{G}=(\boldsymbol{X}\cup\boldsymbol{Y}, \boldsymbol{E})$ with $|\boldsymbol{X}|=|\boldsymbol{Y}|=k$, there always exists a perfect matching in $\boldsymbol{G}$ if its minimum degree is greater than $k/2$.

Since n configurations are constructed ahead of $\boldsymbol{P}_{n+1}$, there are at most n 0*s* (i.e., covered entries) in each line of $\boldsymbol{U}$. Because



Selected-entries (should be set to 0 in ***R*** once selected/covered).

This example assumes N=7 and n=1 for constructing $\boldsymbol{P}_2$.

Matrix ***U*** is obtained from the not-yet-shadowed part of ***R***.

Each row/column of ***U*** translates to a vertex in set ***A***/***B***. Each "1" in ***U*** translates to an edge between the two corresponding vertices in $\boldsymbol{U}_G$.

This perfect matching corresponds to the MSM-entries in the dashed circles.

Perform maximum-size matching in $\boldsymbol{U}_G$ to find a perfect matching.

A matching in $\boldsymbol{U}_G$ is a subset of edges where each vertex is incident on at most one edge.
A perfect matching in $\boldsymbol{U}_G$ is a subset of edges where each vertex is incident on exactly one edge.
A maximum-size matching (MSM) algorithm [22] can find a matching with the maximum number of edges.

Fig. 7. Bipartite graph, perfect matching and maximum-size matching (MSM).

U is a (2n+1)×(2n+1) matrix, the minimum degree of the corresponding bipartite graph $\boldsymbol{U}_G$ is at least (2n+1)−n= n+1>(2n+1)/2. Therefore, a perfect matching containing (2n+1) edges exists in $\boldsymbol{U}_G$ according to the Theorem.

When $\boldsymbol{P}_{n+1}$ is obtained, we un-shadow ***C*(*T*)** and ***R***, and repeat the above procedure (i.e., *Steps 2a-2c* in Fig. 5) to construct the next configuration. In Fig. 5, w is an auxiliary variable that helps to pick up the first-met largest entry as the weight ϕ_{n+1} for $\boldsymbol{P}_{n+1}$. This ensures that the corresponding entries in ***C*(*T*)** can be properly covered by $\boldsymbol{P}_{n+1}$ with a sufficiently large weight.

So far, we have discussed all the key operations of QLEF algorithm. QLEF also has two additional features. First, since N−(2n+1)>0 requires n<(N−1)/2, we use the above approach (i.e., *Step 2* in Fig. 5) to construct only the first $\lceil N/2\rceil-1$ configurations. This ensures that the N−(2n+1) selected-entries in $\boldsymbol{P}_{n+1}$ can always be properly chosen. Second, after the first $\lceil N/2\rceil-1$ configurations are obtained, we find the largest c_{ij} in the updated ***C*(*T*)** and use it as the constant weight for each subsequent configuration (so as to cover the remaining small entries). Because the bipartite graph of ***R*** is always *regular* [8, 10, 21] after a configuration is constructed, each subsequent configuration in $\{\boldsymbol{P}_{\lceil N/2\rceil}, \ldots, \boldsymbol{P}_N\}$ can be obtained by performing maximum-size matching in the updated ***R***.

$$\boldsymbol{C(T)}=\begin{bmatrix}1&2&10&13&1&2&6\\6&2&4&13&3&4&4\\4&9&5&3&5&5&5\\8&1&6&4&1&8&8\\3&12&3&1&1&8&8\\7&7&3&1&13&0&5\\7&3&5&0&12&9&0\end{bmatrix}\le 13\underbrace{\begin{bmatrix}0&0&0&\textcircled{1}&0&0&0\\0&0&0&0&0&0&1\\0&0&\textcircled{1}&0&0&0&0\\\textcircled{1}&0&0&0&0&0&0\\0&\textcircled{1}&0&0&0&0&0\\0&0&0&0&\textcircled{1}&0&0\\0&0&0&0&0&\textcircled{1}&0\end{bmatrix}}_{\boldsymbol{P}_1}+$$

$$13\underbrace{\begin{bmatrix}0&0&\textcircled{1}&0&0&0&0\\0&0&0&\textcircled{1}&0&0&0\\0&\textcircled{1}&0&0&0&0&0\\0&0&0&0&0&0&1\\0&0&0&0&0&1&0\\1&0&0&0&0&0&0\\0&0&0&0&\textcircled{1}&0&0\end{bmatrix}}_{\boldsymbol{P}_2}+8\underbrace{\begin{bmatrix}0&0&0&0&1&0&0\\0&0&1&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&0&\textcircled{1}&0\\0&0&0&0&0&0&\textcircled{1}\\0&1&0&0&0&0&0\\1&0&0&0&0&0&0\end{bmatrix}}_{\boldsymbol{P}_3}+6\underbrace{\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&1&0&0\\0&0&1&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&1\end{bmatrix}}_{\boldsymbol{P}_4}+$$

$$6\underbrace{\begin{bmatrix}0&1&0&0&0&0&0\\1&0&0&0&0&0&0\\0&0&0&0&0&1&0\\0&0&0&0&1&0&0\\0&0&1&0&0&0&0\\0&0&0&0&0&0&1\\0&0&0&1&0&0&0\end{bmatrix}}_{\boldsymbol{P}_5}+6\underbrace{\begin{bmatrix}0&0&0&0&0&0&1\\0&0&0&0&0&1&0\\1&0&0&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&1&0&0\\0&0&1&0&0&0&0\\0&1&0&0&0&0&0\end{bmatrix}}_{\boldsymbol{P}_6}+6\underbrace{\begin{bmatrix}0&0&0&0&0&1&0\\0&0&0&0&1&0&0\\0&0&0&0&0&0&1\\0&1&0&0&0&0&0\\1&0&0&0&0&0&0\\0&0&0&1&0&0&0\\0&0&1&0&0&0&0\end{bmatrix}}_{\boldsymbol{P}_7}$$

Fig. 8. An example of QLEF. The circled entries are select-entries.

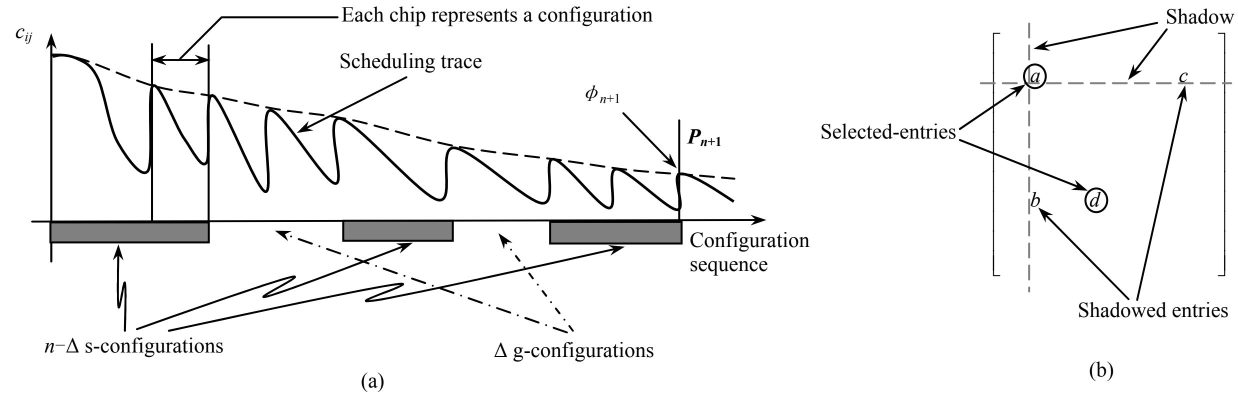
Summarily, QLEF guarantees that there is no configuration overlap in scheduling ***C*(*T*)**. So, ***C*(*T*)** can always be covered by N configurations. The time complexity of QLEF is dominated by running the maximum-size matching algorithm [22] (which has a complexity of $O(N^{2.5})$) for N times, resulting in the same time complexity of $O(N^{3.5})$ as MIN [8] and α^i-SCALE [10]. But edge-coloring [8, 23] and partitioning in MIN and α^i-SCALE do not appear in QLEF.

C. *An Example*

To cover the 7×7 traffic matrix ***C*(*T*)** in Fig. 8, QLEF generates 7 configurations with a weight sum of 58. As an example, the construction process of $\boldsymbol{P}_2$ in Fig. 8 is detailed in Fig. 7. For a given traffic matrix, we can also find an optimal schedule using the ILP (Integer Linear Program) formulated in Appendix A. (But ILP is generally impractical due to its extremely long execution time.) Appendix A gives an ILP-based schedule for the traffic matrix in Fig. 8. The weight sum 58 in QLEF is only slightly above the ILP-based result.

III. Speedup Bound

In QLEF, the N configurations are sequentially constructed from $\boldsymbol{P}_1$ to $\boldsymbol{P}_N$. We first focus on the first half configurations $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n, \boldsymbol{P}_{n+1}, \ldots, \boldsymbol{P}_{\lceil N/2\rceil-1}$. Fig. 9 shows the conceptual QLEF scheduling procedure. In Fig. 9a, we use a "scheduling trace" to represent the trend of c_{ij} values covered in the ordered



(a)

(b)

Fig. 9. Conceptual QLEF scheduling procedure.

configurations. Although QLEF gives priority in covering the largest entry in the not-yet-shadowed part of ***C*(*T*)**, the scheduling trace is generally a wavelike curve rather than a monotonically decreasing curve. This is due to the shadowing operation. In Fig. 9b, assume that entry *a* is first chosen as a selected-entry. Then, entries *b* and *c* are shadowed, where *b*≤*a* and *c*≤*a*. Next, entry *d* is chosen as a selected-entry even though *d*<*b* or *d*<*c*. So *b* and *c* can only be covered by other configurations constructed later. On the other hand, we can see that the selected-entries covered by the same configuration are always chosen in a descending order of their entry values, e.g., *a*>*d* in Fig. 9b.

We now focus on the construction of configuration $\boldsymbol{P}_{n+1}$. Note that ϕ_{n+1} (the weight of $\boldsymbol{P}_{n+1}$) is also an entry in ***C*(*T*)**, and it is the first selected-entry in $\boldsymbol{P}_{n+1}$. Hereafter, we treat ϕ_{n+1} as an entry in ***C*(*T*)** rather than a weight. Let $\{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\}$ denote the set of *n* ordered configurations constructed earlier than $\boldsymbol{P}_{n+1}$. If ϕ_{n+1} is shadowed in the construction process (*Step 2b* of Fig. 5) of $\boldsymbol{P}_k \in \{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\}$, then $\boldsymbol{P}_k$ is called an *s-configuration*. Otherwise, $\boldsymbol{P}_k$ is called a *g-configuration*. Among the *n* configurations $\{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\}$, assume there are Δ g-configurations and $n-\Delta$ s-configurations, as shown in Fig. 9a.

A. General Idea for Speedup Bound Formulation

In the *original* ***C*(*T*)**, we define an entry larger than or equal to ϕ_{n+1} as a *large-entry* (or LE). Let *M* be the minimum number of LEs covered by the first *n* configurations $\{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\}$. These *M* LEs reside in *N* lines (rows or columns) of ***C*(*T*)**, and the line with the maximum number of LEs must contain at least the average number of $\lceil M/N \rceil$ LEs. As a result, the smallest LE in this line must be smaller than or equal to the $\lceil M/N \rceil$-th largest entry of this line. Yet, this smallest LE is not smaller than ϕ_{n+1}. Because the maximum line sum of ***C*(*T*)** is *T*, according to Lemma 1 in Appendix B, we have

$$\phi_{n+1} \le \frac{T}{\left\lceil \frac{M}{N} \right\rceil}. \tag{5}$$

On the other hand, because ϕ_{n+1} is shadowed by $n-\Delta$ s-configurations, from Lemma 2 in Appendix B, we have

$$\phi_{n+1} \le \frac{T}{\left\lceil \frac{n-\Delta}{2} \right\rceil + 1} = \frac{T}{\left\lceil \frac{n-\Delta}{2} + 1 \right\rceil}. \tag{6}$$

Combining (5) and (6), we can bound ϕ_{n+1} as follows:

$$\phi_{n+1} \le \max_{0\le\Delta\le n}\left\{ \min\left[\frac{T}{\left\lceil \frac{M}{N} \right\rceil} \ , \ \frac{T}{\left\lceil \frac{n-\Delta}{2} + 1 \right\rceil} \right]\right\}$$
$$\text{for} \quad 0 \le n < \left\lceil \frac{N}{2} \right\rceil - 1. \tag{7}$$

The inequality in (7) indicates that no matter what is the value of Δ ($0\le\Delta\le n$), the bound always holds because we have taken the worst case into consideration (i.e., the "max" function).

For the remaining $N-\lceil N/2 \rceil+1$ configurations constructed in *Step 3* of Fig. 5, QLEF uses a small constant as their common weight. Since the weights in QLEF follow a monotonically decreasing order as shown by the dashed line in Fig. 9a, this constant is not larger than any weight of the first $\lceil N/2 \rceil - 1$ configurations. In fact, it can be bounded by the weight of $\boldsymbol{P}_{\lceil N/2 \rceil - 1}$. That is

$$\phi_{n+1}\Big|_{N \ge n+1 \ge \left\lceil \frac{N}{2} \right\rceil} \le \phi_{\left\lceil \frac{N}{2} \right\rceil - 1}. \tag{8}$$

Consequently, we can replace the weights in (4) by the bounds in (7) & (8) to get an upper-bound for S_{schedule}. (Note that N_S=*N* in (4) for minimum delay scheduling.)

B. A Simple Speedup Bound

To get an S_{schedule} bound, we still need to determine *M* in (7) which denotes the minimum number of LEs covered by $\{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\}$. In this part, we first count *M* using a simple approach, and a more in-depth analysis will be carried out in Part C to render a refined speedup bound. The simple speedup bound provides a reference to which we can check how much additional gain is achieved by the refined speedup bound. It also helps to better understand the refined speedup bound in Part C.

According to QLEF, all the selected-entries in any g-configuration must be LEs, and each s-configuration must cover at least one LE in the same line as ϕ_{n+1}. However, due to the shadowing operation, it is generally difficult to count how many additional LEs in other lines are covered by an s-configuration. Here we simply ignore the LEs contributed by the $n-\Delta$ s-configurations, and only count those LEs (or selected-entries) contributed by the Δ g-configurations. Then, M can be minimized only if the Δ g-configurations are the last Δ (out of the first n) configurations, or $\{\boldsymbol{P}_{n-\Delta+1}, \ldots, \boldsymbol{P}_n\}$. This is because $N-(2n+1)$ decreases as n increases, and thus the number of selected-entries in each subsequent configuration becomes less and less. As a result, the minimum value of M is $M=(N-2n+1)+(N-2n+3)+(N-2n+5)+\ldots+[(N-2n)+(2\Delta-1)]=(N-2n)\Delta+\Delta^2$. Note that all the counted LEs are contributed by the Δ g-configurations, and thus none of them resides in the same line as ϕ_{n+1}. In other words, these M LEs reside in $N-1$ lines instead of N lines of $\boldsymbol{C(T)}$. Replacing N in (7) by $N-1$ and substituting $M=(N-2n)\Delta+\Delta^2$ into (7), we get

$$\phi_{n+1} \le \max_{0\le\Delta\le n}\left\{\min\left[\left\lceil\frac{T}{\left\lceil\frac{(N-2n)\Delta+\Delta^2}{N-1}\right\rceil}\right\rceil , \left\lceil\frac{T}{\left\lceil\frac{n-\Delta}{2}+1\right\rceil}\right\rceil\right]\right\}$$

$$\text{for}\quad 0\le n<\left\lceil\frac{N}{2}\right\rceil-1. \tag{9}$$

This is equivalent to

$$\phi_{n+1} \le \frac{T}{\left\lceil\frac{n-\Delta}{2}+1\right\rceil}\Bigg|_{0\le n<\left\lceil\frac{N}{2}\right\rceil-1 \text{ and } \Delta=\left\lfloor\frac{\sqrt{(3N-4n-1)^2+8(N-1)(n+2)}-(3N-4n-1)}{4}\right\rfloor}. \tag{10}$$

Consequently, S_{schedule} can be bounded by

$$S_{\text{schedule}}=\frac{1}{T}\sum_{n=1}^{N}\phi_n \le \sum_{n+1=1}^{\lceil N/2\rceil-1}\frac{1}{\left\lceil\frac{n-\Delta}{2}+1\right\rceil}\Bigg|_{\Delta=\left\lfloor\frac{\sqrt{(3N-4n-1)^2+8(N-1)(n+2)}-(3N-4n-1)}{4}\right\rfloor}+$$

$$\frac{\left\lfloor\frac{N}{2}\right\rfloor+1}{\left\lceil\frac{n-\Delta}{2}+1\right\rceil}\Bigg|_{n+1=\left\lceil\frac{N}{2}\right\rceil-1 \text{ and } \Delta=\left\lfloor\frac{\sqrt{(3N-4n-1)^2+8(N-1)(n+2)}-(3N-4n-1)}{4}\right\rfloor}. \tag{11}$$

The bound in (11) is plotted in Fig. 10 by a dashed line.

C. A Refined Speedup Bound

A lower S_{schedule} bound can be obtained if the LEs covered by the $n-\Delta$ s-configurations are judiciously counted. Assume $\{\boldsymbol{P_s}\}$ is a set of consecutive s-configurations and $\boldsymbol{P_g}$ is the first g-configuration after $\{\boldsymbol{P_s}\}$. According to Lemma 3 in Appendix B, the number of LEs covered by each s-configuration $\boldsymbol{P_s}\in\{\boldsymbol{P_s}\}$ is at least half of the number of the selected-entries in $\boldsymbol{P_g}$.

In Fig. 9a, the Δ g-configurations and the $n-\Delta$ s-configurations may line up in any order. From Lemma 4 in Appendix B, in order to minimize M, the $n-\Delta$ s-configurations should be consecutively located at either the very beginning or the very end of the configuration sequence $\{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\}$.

Case 1: The $n-\Delta$ s-configurations are consecutively located at the very end of the configuration sequence $\{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\}$. In this case, all the selected-entries covered by the Δ g-configurations are LEs. Since no g-configuration follows the $n-\Delta$ s-configurations, the number of LEs contributed by the s-configurations is trivial and is ignored when counting M. So, $M=(N-1)+(N-3)+\ldots+(N-2\Delta+1)=(N-\Delta)\Delta$. These LEs reside in $N-1$ lines of $\boldsymbol{C(T)}$. Replacing N in (7) by $N-1$ and substituting $M=(N-\Delta)\Delta$ into (7), we have

$$\phi_{n+1} \le \max_{0\le\Delta\le n}\left\{\min\left[\left\lceil\frac{T}{\left\lceil\frac{(N-\Delta)\Delta}{N-1}\right\rceil}\right\rceil , \left\lceil\frac{T}{\left\lceil\frac{n-\Delta}{2}+1\right\rceil}\right\rceil\right]\right\}$$

$$\text{for}\quad 0\le n<\left\lceil\frac{N}{2}\right\rceil-1. \tag{12}$$

or

$$\phi_{n+1} \le \frac{T}{\left\lceil\frac{n-\Delta}{2}+1\right\rceil}\Bigg|_{0\le n<\left\lceil\frac{N}{2}\right\rceil-1 \text{ and } \Delta=\frac{(3N-1)-\sqrt{(3N-1)^2-8(N-1)(n+2)}}{4}}. \tag{13}$$

Case 2: The $n-\Delta$ s-configurations are consecutively located at the beginning of the configuration sequence $\{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\}$. In this case, the first g-configuration (i.e., $\boldsymbol{P}_{n-\Delta+1}$) has $(N-1)-2(n-\Delta)$ selected-entries. Therefore, each s-configuration will cover at least $(N-1)/2-(n-\Delta)$ LEs according to Lemma 3 in Appendix B. Taking the LEs (or selected-entries) covered by the Δ g-configurations into account, we get $M=Nn-n^2-(N+1)(n-\Delta)/2$ by simple calculation. From (7) we have

$$\phi_{n+1} \le \max_{0\le\Delta\le n}\left\{\min\left[\left\lceil\frac{T}{\left\lceil\frac{2Nn-2n^2-(N+1)(n-\Delta)}{2N}\right\rceil}\right\rceil , \left\lceil\frac{T}{\left\lceil\frac{n-\Delta}{2}+1\right\rceil}\right\rceil\right]\right\}$$

$$\text{for}\quad 0\le n<\left\lceil\frac{N}{2}\right\rceil-1. \tag{14}$$

Again, this is equivalent to

$$\phi_{n+1} \le \frac{T}{\left\lceil\frac{n-\Delta}{2}+1\right\rceil}\Bigg|_{0\le n<\left\lceil\frac{N}{2}\right\rceil-1 \text{ and } \Delta=\frac{2n^2+n+2N}{2N+1}}. \tag{15}$$

$$\phi_{n+1}\Bigg|_{0\le n<\left\lceil\frac{N}{2}\right\rceil-1} \le \max\left\{\frac{T}{\left\lceil\frac{n-\Delta}{2}+1\right\rceil}\Bigg|_{\Delta=\frac{(3N-1)-\sqrt{(3N-1)^2-8(N-1)(n+2)}}{4}} , \frac{T}{\left\lceil\frac{n-\Delta}{2}+1\right\rceil}\Bigg|_{\Delta=\frac{2n^2+n+2N}{2N+1}}\right\} ,\quad \phi_{n+1}\Bigg|_{N\ge n+1\ge\left\lceil\frac{N}{2}\right\rceil} \le \phi_{\left\lceil\frac{N}{2}\right\rceil-1} . \tag{16}$$

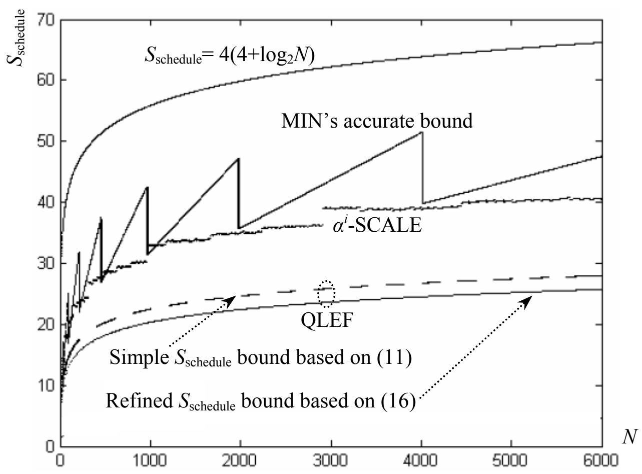


Fig. 10. S_{schedule} bound of the minimum delay scheduling algorithms.

Combining (13), (15) and (8), we get the bound for ϕ_{n+1} in (16). Then, we can replace the weights in (4) by the bound in (16) to find the refined S_{schedule} bound for QLEF.

D. Performance Comparison and Discussion

Fig. 10 shows the S_{schedule} bounds for the three minimum delay scheduling algorithms MIN [8], α^i-SCALE [10] and QLEF. The original bound for MIN algorithm is S_{schedule}=4(4+log$_2N$) in [8], which is refined in [10] to produce the saw-toothed curve in Fig. 10. A tighter bound is then provided by α^i-SCALE algorithm [10]. This is followed by the two QLEF bounds derived in this paper using (11) & (16) respectively. We can see that if the LEs contributed by the s-configurations are judiciously counted, a further cut of 10% in S_{schedule} bound can be achieved (i.e., solid vs dashed QLEF bound in Fig. 10).

As an example, we consider a switch with size N=450. MIN requires S_{schedule}=37.13 and α^i-SCALE requires S_{schedule}=27.82. However, QLEF only requires S_{schedule}=17.89. This gives a cut of 52% over MIN and 36% over α^i-SCALE.

QLEF is designed for delay sensitive applications, where the packet delay bound is close to the ideal minimum bound of $3\delta N+H$. Note that two performance guaranteed scheduling algorithms ADAPT and SRF are proposed in [9]. Both of them are self-adaptive to the given switch parameters (δ, T and N) by generating a proper number of N_S configurations to minimize the required speedup. However, ADAPT and SRF require N_S>N configurations in the schedule and thus are not minimum delay scheduling algorithms. Though they can generate a schedule with N_S slightly larger than N (e.g., N_S=N+1, N+2, etc.), the required speedup is significantly larger than that required by QLEF. On the other hand, if the application is not delay sensitive and N_S>>N is allowed, then minimum delay scheduling is not necessary and ADAPT/SRF may be favorable. Therefore, QLEF complements ADAPT and SRF for delay sensitive applications.

IV. Conclusion

Recent progress of optical switching technologies has enabled the implementation of high-speed scalable switches with optical fabrics. Due to the reconfiguration overhead, packet delay can be minimized by using the minimum number of N_S=N switch configurations (where N is the switch size) to schedule the traffic (i.e. minimum delay scheduling). In general, minimum delay scheduling can be formulated as a traffic matrix decomposition problem with the constraint of N_S=N. The traffic matrix is decomposed into the weighted sum of N non-overlapping permutation matrices. In this paper, a novel minimum delay scheduling algorithm QLEF (Quasi Largest-Entry-First) was proposed. It minimizes the required speedup for achieving performance guaranteed switching (i.e. 100% throughput with bounded packet delay), while ensuring the minimum number of N_S=N configurations in the schedule. Compared with the two existing minimum delay scheduling algorithms MIN and α^i-SCALE, QLEF dramatically cuts down the required speedup bound with the same time complexity of $O(N^{3.5})$.

Although QLEF is presented based on optical switch scheduling, it may find wide applications in similar scheduling problems, such as those in SS/TDMA [15, 20, 24], TWIN [25] and sensor surveillance networks [26].

Appendix A ILP Formulation

To cover a given $N\times N$ matrix $\boldsymbol{C(T)}$=$\{c_{ij}\}$, we can use the ILP (Integer Linear Program) below to construct an optimal schedule consisting of N configurations $\boldsymbol{P_n}$=$\{p^{(n)}_{ij}\}$ with corresponding weights ϕ_n ($N\geq n\geq 1$). In the ILP, ϕ_n and $a^{(n)}_{ij}$ are general integer variables [27], $p^{(n)}_{ij}$ is a binary variable, and T is the maximum line sum of the matrix.

$$\text{minimize}\left\{\sum_{n=1}^{N}\phi_n\right\}; \tag{17}$$

s. t.

$$\sum_{j=0}^{N-1} p_{ij}^{(n)} = 1, \quad \forall n\in\{1,...,N\},\ \forall i\in\{0,...,N-1\}; \tag{18}$$

$$\sum_{i=0}^{N-1} p_{ij}^{(n)} = 1, \quad \forall n\in\{1,...,N\},\ \forall j\in\{0,...,N-1\}; \tag{19}$$

$$T\left(1-p_{ij}^{(n)}\right)+\phi_n \geq a_{ij}^{(n)}, \quad \forall n\in\{1,...,N\},\ \forall (i,j)\in\{0,...,N-1\}; \tag{20}$$

$$a_{ij}^{(n)} \leq Tp_{ij}^{(n)}, \quad \forall n\in\{1,...,N\},\ \forall (i,j)\in\{0,...,N-1\}; \tag{21}$$

$$\sum_{n=1}^{N} a_{ij}^{(n)} \geq c_{ij}, \quad \forall (i,j)\in\{0,...,N-1\}. \tag{22}$$

The sum of the N weights is minimized in (17). Constraints (18) & (19) define each configuration $\boldsymbol{P_n}$ as a permutation matrix. Constraints (20) & (21) define the auxiliary variable $a^{(n)}_{ij}$. For an arbitrary entry $p^{(n)}_{ij}\in\boldsymbol{P_n}$, $a^{(n)}_{ij}$ takes 0 if $p^{(n)}_{ij}$ =0, and ϕ_n if $p^{(n)}_{ij}$ =1. Then, constraint (22) ensures that all the entries in $\boldsymbol{C(T)}$ are properly covered by the N configurations. Note that the ILP formulated in (17)-(22) allows configuration overlaps in the schedule.

$$\boldsymbol{C(T)}=\begin{bmatrix}1&2&10&13&1&2&6\\6&2&4&13&3&4&4\\4&9&5&3&5&5&5\\8&1&6&4&1&8&8\\3&12&3&1&1&8&8\\7&7&3&1&13&0&5\\7&3&5&0&12&9&0\end{bmatrix}\le 3\underbrace{\begin{bmatrix}1&0&0&0&0&0&0\\0&0&0&0&1&0&0\\0&0&0&1&0&0&0\\0&1&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&1\end{bmatrix}}_{\boldsymbol{P}_1}+$$

$$13\underbrace{\begin{bmatrix}0&0&0&1&0&0&0\\1&0&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&0&0&0&1\\0&1&0&0&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\end{bmatrix}}_{\boldsymbol{P}_2}+13\underbrace{\begin{bmatrix}0&0&1&0&0&0&0\\0&0&0&1&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&1\\1&0&0&0&0&0&0\\0&0&0&0&1&0&0\end{bmatrix}}_{\boldsymbol{P}_3}+5\underbrace{\begin{bmatrix}0&1&0&0&0&0&0\\0&0&0&0&0&0&1\\0&0&0&0&0&1&0\\0&0&0&0&1&0&0\\1&0&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&1&0&0&0\end{bmatrix}}_{\boldsymbol{P}_4}+$$

$$5\underbrace{\begin{bmatrix}0&0&0&0&0&1&0\\0&1&0&0&0&0&0\\1&0&0&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&0&1\\0&0&1&0&0&0&0\end{bmatrix}}_{\boldsymbol{P}_5}+7\underbrace{\begin{bmatrix}0&0&0&0&1&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&1\\0&0&1&0&0&0&0\\0&0&0&1&0&0&0\\0&1&0&0&0&0&0\\1&0&0&0&0&0&0\end{bmatrix}}_{\boldsymbol{P}_6}+8\underbrace{\begin{bmatrix}0&0&0&0&0&0&1\\0&0&1&0&0&0&0\\0&0&0&0&1&0&0\\1&0&0&0&0&0&0\\0&0&0&0&0&1&0\\0&0&0&1&0&0&0\\0&1&0&0&0&0&0\end{bmatrix}}_{\boldsymbol{P}_7}$$

Fig. 11. An ILP-based schedule for ***C*(*T*)** in Fig. 8.

We implement the above ILP in ILOG CPLEX 10.0 [27] on a standard Pentium IV 2.2 GHz computer with 500M memory. Fig. 11 shows an ILP-based solution for ***C*(*T*)** in Fig. 8 with T=36. It is obtained in 34.79 hours (125236.29 seconds) with a gap-to-optimality of 45.78% (the optimization stops due to out of memory), and the weight sum is 54. We also slightly modify the above ILP to forbid configuration overlaps. Then, an optimal solution can be obtained in 43.69 hours (157284.51 seconds) with a weight sum of 53.

APPENDIX B LEMMAS

Lemma 1: If a set of N non-negative entries $\boldsymbol{E}=\{e_1, e_2, \ldots, e_N\}$ sum to at most T and e_k is the k-th largest entry in $\boldsymbol{E}$, then $e_k \le T/k$.

Proof: Assume that the N entries in $\boldsymbol{E}$ are arranged in a monotonically decreasing order as $e_1 \ge e_2 \ge \ldots \ge e_N$. Because $\sum_{i=1}^{N} e_i \le T$, e_k can reach its maximum possible value only when $e_{k+1}= e_{k+2}=\ldots=e_N=0$ and $e_1=e_2=\ldots=e_k$. We then have

$$e_k=\frac{1}{k}\sum_{i=1}^{k}e_i=\frac{1}{k}\sum_{i=1}^{N}e_i\le\frac{T}{k}.$$

#

Lemma 2: In QLEF, if ϕ_{n+1} is shadowed by k s-configurations, then

$$\phi_{n+1}\le\frac{T}{\left\lceil\frac{k}{2}\right\rceil+1}.$$

Proof: In QLEF, ϕ_{n+1} is shadowed in the construction of each s-configuration. Because ϕ_{n+1} can only be shadowed by an LE in the same line with it, the k s-configurations collectively cover at least k LEs in the same line with ϕ_{n+1}.

Assume that ϕ_{n+1} resides in row i and column j. Some of those k LEs may reside in row i whereas others reside in column j (because a line may refer to either a row or a column). Without loss of generality, we assume that k' out of the k LEs reside in row i and the other $k-k'$ LEs reside in column j. As a result, ϕ_{n+1} is (at most) the (k'+1)-th largest entry in row i and the ($k-k'$+1)-th largest entry in column j. Because the entries in either row i or column j sum to at most T, from Lemma 1 we have

$$\phi_{n+1}\le\frac{T}{k'+1}\text{ and }\phi_{n+1}\le\frac{T}{k-k'+1}.$$

That is,

$$\phi_{n+1}\le\min\left\{\frac{T}{k'+1}\quad,\quad\frac{T}{k-k'+1}\right\}\le\frac{T}{\left\lceil\frac{k}{2}\right\rceil+1}.$$

#

Lemma 3: Assume that $\{\boldsymbol{P_s}\}$ is a set of consecutive s-configurations, and $\boldsymbol{P_g}$ is the first g-configuration after $\{\boldsymbol{P_s}\}$. Then, any s-configuration $\boldsymbol{P_s}\in\{\boldsymbol{P_s}\}$ must cover at least h LEs, where h is half of the number of the selected-entries covered by $\boldsymbol{P_g}$.

Proof: Since $\boldsymbol{P_g}$ is a g-configuration, any selected-entry α covered by $\boldsymbol{P_g}$ is an LE. Because $\boldsymbol{P_s}$ is constructed earlier than $\boldsymbol{P_g}$ and α is not covered by $\boldsymbol{P_s}$, either 1) all the selected-entries covered by $\boldsymbol{P_s}$ are not smaller than α, or 2) α is shadowed in $\boldsymbol{P_s}$ construction if some smaller selected-entries are covered by $\boldsymbol{P_s}$. (Otherwise $\boldsymbol{P_s}$ should first cover α instead of those smaller selected-entries.)

In case 1), all the selected-entries covered by $\boldsymbol{P_s}$ are LEs, because each of them is not smaller than α (which is an LE). Since $\boldsymbol{P_s}$ is constructed earlier than $\boldsymbol{P_g}$, it contains more selected-entries than $\boldsymbol{P_g}$. Therefore, the number of LEs covered by $\boldsymbol{P_s}$ is larger than h. In case 2), any selected-entry α covered by $\boldsymbol{P_g}$ must have been shadowed in $\boldsymbol{P_s}$ construction. Since a selected-entry in $\boldsymbol{P_s}$ can shadow at most two smaller/equal selected-entries covered by $\boldsymbol{P_g}$ (in row and column directions, respectively), $\boldsymbol{P_s}$ must cover at least h LEs, where h is half of the number of the selected-entries covered by $\boldsymbol{P_g}$. Obviously, this is true for the first g-configuration $\boldsymbol{P_g}$ after $\{\boldsymbol{P_s}\}$.

#

Lemma 4: To minimize M (the total number of LEs), all the s-configurations should be consecutively located at either the very beginning or the very end of the configuration sequence $\{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\}$.

Proof: In Fig. 12, let y-axis denote the number of selected-entries covered in each configuration, and x-axis denote the configuration sequence. Without loss of generality, assume there are three sets of consecutive s-configurations, denoted by $\boldsymbol{A}_1$, $\boldsymbol{A}_2$ and $\boldsymbol{A}_3$. (Others are g-configurations.) Particularly, $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ contain x_1 and x_2 s-configurations respectively, and $\boldsymbol{A}_3$ locates at the very end of the configuration sequence $\{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\}$. The first s-configuration in $\boldsymbol{A}_1$ covers y_1 selected-entries, and the first g-configuration after $\boldsymbol{A}_1$ covers

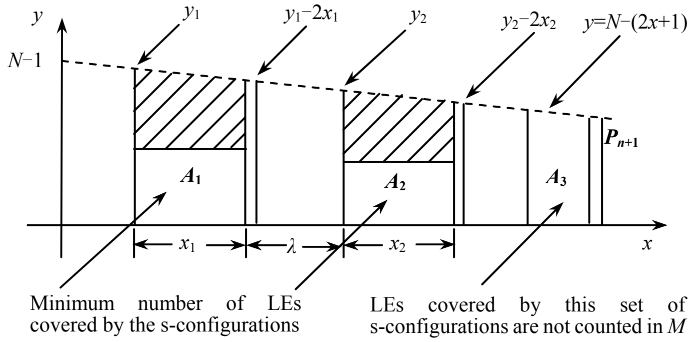

y
y1
y1−2x1
y2
y2−2x2
y=N−(2x+1)
N−1
P_{n+1}
A1
A2
A3
x1
λ
x2
x
Minimum number of LEs covered by the s-configurations
LEs covered by this set of s-configurations are not counted in M

Fig. 12. s-configurations may also cover a considerable number of LEs.

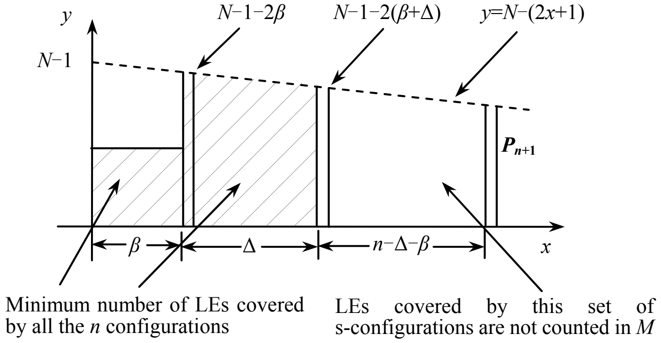

y
N−1−2β
N−1−2(β+Δ)
y=N−(2x+1)
N−1
P_{n+1}
β
Δ
n−Δ−β
x
Minimum number of LEs covered by all the n configurations
LEs covered by this set of s-configurations are not counted in M

Fig. 13. The $n-\Delta$ s-configurations should be located at either end of $\{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\}$.

(y_1-2x_1) selected-entries. Similarly, the first s-configuration in $\boldsymbol{A}_2$ covers y_2 selected-entries, and the first g-configuration after $\boldsymbol{A}_2$ covers (y_2-2x_2) selected-entries. From Lemma 3, each s-configuration in $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ covers at least (y_1-2x_1)/2 and (y_2-2x_2)/2 LEs respectively, as shown by the blank rectangles in $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ (see Fig. 12). However, we do not count any LEs covered by $\boldsymbol{A}_3$, because $\boldsymbol{A}_3$ is right before $\boldsymbol{P}_{n+1}$ and there is no g-configuration after it. Although each s-configuration in $\boldsymbol{A}_3$ covers (at least) one LE in the same line with Φ_{n+1}, it is trivial and is ignored when counting M.

We first consider $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. Since all selected-entries covered by g-configurations are LEs, minimizing M is equivalent to maximizing the number of selected-entries in the shadowed areas in $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. That is

$$\max\left\{\left[x_1\left(\frac{y_1-2x_1}{2}+2\right)+\frac{x_1(x_1-1)}{2}\times 2\right]+\left[x_2\left(\frac{y_2-2x_2}{2}+2\right)+\frac{x_2(x_2-1)}{2}\times 2\right]\right\}$$

$$\text{or}\quad \max\left\{\frac{(y_1+2)x_1+(y_2+2)x_2}{2}\right\}.$$

Let λ be the number of g-configurations between $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. We have $y_2=y_1-2(x_1+\lambda)$. So, the above formula is equivalent to

$$\max\left\{\frac{(y_1+2)(x_1+x_2)-2x_2(x_1+\lambda)}{2}\right\}.$$

For any given x_1 and x_2, the above value is maximized if y_1 takes the maximum possible value and $\lambda=0$. This entails that $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ should be consecutively located at the very beginning of the configuration sequence $\{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\}$. It is easy to generalize this conclusion to the case where more sets of consecutive s-configurations are involved.

We still need to consider $\boldsymbol{A}_3$ in Fig. 12. In fact, the n configurations $\{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\}$ may also line up as shown in Fig. 13, where the Δ g-configurations are in the middle and the $n-\Delta$ s-configurations are at the both ends. (Assume that β s-configurations are consecutively located at the beginning of $\{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\}$ to minimize M as discussed earlier.) In Fig. 13, minimizing M is equivalent to maximizing the number of selected-entries in the two blank echelons. That is,

$$\max_{0\le\beta\le n-\Delta}\left\{\left[\left(\frac{N-1-2\beta}{2}+2\right)\beta+\frac{\beta(\beta-1)}{2}\times 2\right]+\left[(n-\Delta-\beta)(N-2n+1)+\frac{(n-\Delta-\beta)(n-\Delta-\beta-1)}{2}\times 2\right]\right\}$$

$$\text{or}\quad \max_{0\le\beta\le n-\Delta}\left\{\beta^2+\frac{4\Delta-N+1}{2}\beta+(n-\Delta)(N-n-\Delta)\right\}.$$

Because the above formula is a quadratic function of β, it can be maximized only if $\beta=0$ or $\beta=n-\Delta$. From Fig. 13, obviously all the $n-\Delta$ s-configurations should be consecutively located at either the very beginning ($\beta=n-\Delta$) or the very end ($\beta=0$) of the configuration sequence $\{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\}$. The specific location is determined by the values of N, n and Δ.

#

REFERENCES

[1] J.E Fouquet et. al, “A compact, scalable cross-connect switch using total internal reflection due to thermally-generated bubbles,” *IEEE LEOS Annual Meeting*, pp. 169-170, Dec. 1998.

[2] L. Y. Lin, “Micromachined free-space matrix switches with submilli-second switching time for large-scale optical crossconnect,” *OFC’98 Tech. Digest*, pp. 147-148, Feb. 1998.

[3] O. B. Spahn, C. Sullivan, J. Burkhart, C. Tigges, and E. Garcia, “GaAs-based microelectromechanical waveguide switch,” *Proc. 2000 IEEE/LEOS Intl. Conf. on Optical MEMS*, pp. 41-42, Aug. 2000.

[4] A. J. Agranat, “Electroholographic wavelength selective crossconnect,” *1999 Digest of the LEOS Summer Topical Meetings*, pp. 61-62, Jul. 1999.

[5] K. Kar, D. Stiliadis, T. V. Lakshman, and L. Tassiulas, “Scheduling algorithms for optical packet fabrics,” *IEEE Journal on Selected Areas in Communications*, vol. 21, issue 7, pp. 1143-1155, Sept. 2003.

[6] G. R. Pieris and G. H. Sasaki, “Scheduling transmissions in WDM broadcast-and-select networks,” *IEEE/ACM Trans. Networking*, vol. 2, no. 2, pp. 105-110, Apr. 1994.

[7] H. Choi, H.-A. Choi and M. Azizoglu, “Efficient scheduling of transmissions in optical broadcast networks,” *IEEE/ACM Trans. Networking*, vol. 4, no. 6, pp. 913-920, Dec. 1996.

[8] B. Towles and W. J. Dally, “Guaranteed scheduling for switches with configuration overhead,” *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 835-847, Oct. 2003.

[9] B. Wu, K. L. Yeung, M. Hamdi and X. Li, “Minimizing internal speedup for performance guaranteed switches with optical fabrics,” *IEEE/ACM Trans. Networking*, to appear. http://www.eee.hku.hk/~kyeung/CameraReady_TNET_00410_2006_R2.

[10] B. Wu and K. L. Yeung, “Scheduling optical packet switches with minimum number of configurations,” *IEEE ICC '05*, vol. 3, pp. 1830-1835, May 2005.

[11] X. Li and M. Hamdi, “On scheduling optical packet switches with reconfiguration delay,” *IEEE Journal on Selected Areas in Communications*, vol. 21, issue 7, pp. 1156-1164, Sept. 2003.

[12] G. Birkhoff, "Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucumán Rev. Ser. A*, vol. 5, pp. 147-151, 1946.

[13] J. von Neumann, "A certain zero-sum two-person game equivalent to the optimal assignment problem," *Contributions to the Theory of Games*, vol. 2, pp. 5-12, Princeton Univ. Press, Princeton, New Jersey, 1953.

[14] M. Hall, *Combinatorial Theory*, Waltham, MA: Blaisdell Publishing Co., 1967.

[15] T. Inukai, "An efficient SS/TDMA time slot assignment algorithm," *IEEE Trans. Commun*, vol. COM-27, no. 10, pp. 1449-1455, 1979.

[16] C. S. Chang, W. J. Chen and H. Y. Huang, "Birkhoff-von Neumann input buffered crossbar switches," *IEEE INFOCOM '00*, vol. 3, pp. 1614-1623, Mar. 2000.

[17] J. Li and N. Ansari, "Enhanced Birkhoff-von Neumann decomposition algorithm for input queued switches," *IEE Proc-Commun.*, vol. 148, issue 6, pp. 339-342, Dec. 2001.

[18] E. G. Coffman and P. J. Denning, *Operation Systems Theory*, Englewood Cliffs, NJ: Prentice-Hall, 1973.

[19] M. Azizoglu, R. A. Barry and A. Mokhtar, "Impact of tuning delay on the performance of bandwidth-limited optical broadcast networks with uniform traffic," *IEEE Journal on Selected Areas in Communications*, vol. 14, issue 5, pp. 935-944, Jun. 1996.

[20] S. Gopal and C. K. Wong, "Minimizing the number of switchings in a SS/TDMA system," *IEEE Trans. Commun.*, vol. COM-33, pp. 497-501, Jun. 1985.

[21] R. Diestel, *Graph Theory*, 2nd ed. New York: Spring-Verlag, 2000.

[22] J. E. Hopcroft and R. M. Karp, "An $n^{5/2}$ algorithm for maximum matching in bipartite graphs," *Soc. Ind. Appl. Math. J. Comput.*, vol. 2, pp. 225-231, 1973.

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

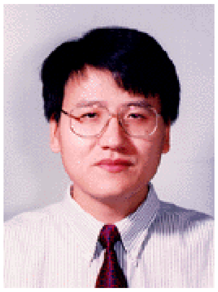
[24] Y. Ito, Y. Urano, T. Muratani, and M. Yamaguchi, "Analysis of a switch matrix for an SS/TDMA system," *Proceedings of the IEEE*, vol. 65, issue 3, pp. 411-419, Mar. 1977.

[25] K. Ross, N. Bambos, K. Kumaran, I. Saniee and I. Widjaja, "Scheduling bursts in time-domain wavelength interleaved networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, issue 9, pp. 1441-1451, Nov. 2003.

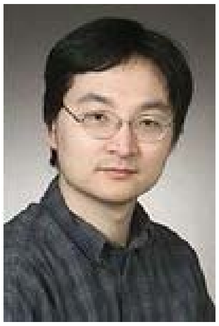
[26] Hai Liu, P. Wan, C.-W. Yi, Xiaohua Jia, S. Makki and N. Pissinou, "Maximal lifetime scheduling in sensor surveillance networks," *IEEE INFOCOM '05*, vol. 4, pp. 2482-2491, Mar. 2005.

[27] CPLEX solver, www.ilog.com.

Bin Wu received the B.Eng. degree in Electrical Engineering from Zhe Jiang University, Hangzhou, China, in 1993, and M.Eng. degree in Communication and Information Engineering from University of Electronic Science and Technology of China, Chengdu, China, in 1996. He joined Huawei Tech. Co. Ltd. in 1996 and served as the department manager of TI-Huawei DSP co-lab during 1997-2001. He is currently working toward the Ph.D. degree in Electrical and Electronic Engineering at The University of Hong Kong, Pokfulam, Hong Kong, with research emphasis on optical networking.



Kwan L. Yeung was born in 1969. He received his B.Eng. and Ph.D. degrees in Information Engineering from The Chinese University of Hong Kong in 1992 and 1995, respectively. He joined the Department of Electrical and Electronic Engineering, The University of Hong Kong in July 2000, where he is currently an Associate Professor, and the Information Engineering Program Co-Director. Before that, he has spent five years in the Department of Electronic Engineering, City University of Hong Kong as an Assistant Professor. During the summer of 1993, Dr. Yeung served with the Performance Analysis Department, AT&T Bell Laboratories (now Bell Labs, Lucent Technologies), Holmdel, USA, as a Member of Technical Staff-I. Dr. Yeung's research interests include next-generation Internet, packet switch/router design, all-optical networks and wireless data networks. He has obtained two patents and published over 120 papers in international journals and conferences since 1993.



Pin-Han Ho received his B.Sc. and M.Sc. degrees from the Electrical and Computer Engineering, Department of National Taiwan University in 1993 and 1995, respectively. He started his Ph.D. studies in 2000 at Queen's University, Kingston, Ontario, Canada, focusing on optical communications systems, survivable networking, and QoS routing problems. He finished his Ph.D. in 2002, and joined the Electrical and Computer Engineering Department at the University of Waterloo as an assistant professor in the same year. He is the author/co-author of more than 100 refereed technical papers and book chapters, and the co-author of a book on optical networking and survivability. He is the recipient of the Distinguished Research Excellence Award in the ECE Department at the University of Waterloo, the Early Researcher Award in 2005, the Best Paper Award at SPECTS '02 and the ICC '05 Optical Networking Symposium, and the Outstanding Paper Award in HPSR '02.

Xiaohong Jiang (M'03) received the B.S., M.S., and Ph.D. degrees from Xidian University, Xi'an, China, in 1989, 1992, and 1999 respectively. He is currently an Associate Professor in the Department of Computer Science, Graduate School of Information Science, Tohoku University, Aramaki Sendai, Japan. Before joining Tohoku University, he was an Assistant Professor in the Graduate School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), Ishikawa, from October 2001 to January 2005. He was a Japan Society for the Promotion of Science (JSPS) Postdoctoral Research Fellow at JAIST from October 1999 to October 2001. He was a Research Associate in the Department of Electronics and Electrical Engineering, University of Edinburgh, Edinburgh, U.K., from March 1999 to October 1999. He has published over 50 refereed technical papers in these areas. His research interests include optical switching networks,WDMnetworks, interconnection networks, IC yield modeling, timing analysis of digital circuits, clock distribution and fault-tolerant technologies for VLSI/WSI.