

Non-Preemptive Scheduling of Optical Switches

Alex Kesselman

Max Planck Institut für Informatik, Saarbrücken, Germany.

Email: akessel@mpi-sb.mpg.de

Kirill Kogan

Cisco Systems, Natanya, Israel.

Email: kkogan@cisco.com

Abstract— Many high-speed routers today use Input-Queued (IQ) architectures with a crossbar switching fabric based on optical technology. Packets in the input queues are divided into cells of unit length and the goal is to find a schedule of minimum makespan that forwards all packets to the output ports. The problem is complicated since in optical switches so called *configuration delay*, that is the time required to reconfigure the switching fabric, is non-negligible with respect to the cell transmission time. We aim to design a scheduler whose complexity does not depend on the number of packets in the input queues. Thus, we focus on the Non-Preemptive Bipartite Scheduling (NPBS) problem, where each input queue is connected to each output port in at most one configuration. We demonstrate that the NPBS problem is NP-hard for any value of the configuration delay and approximation within a ratio smaller than $7/6$ is NP-hard as well. For the offline version of the NPBS problem, we show that a simple greedy algorithm achieves an approximation factor of 2 for arbitrary configuration delay. Then we consider the online version of the NPBS problem, where the switch gathers the incoming traffic periodically and then schedules the accumulated batches (batch scheduling). We propose a scheduling algorithm which guarantees strict delay for any admissible traffic provided that the switch has a moderate speedup of two.

I. INTRODUCTION

The Internet is built around a large variety of transmission and switching systems and the information transfer is aggregated into packets, which are individually forwarded and switched toward their destinations. The main tasks of a router (switch) are to receive a packet from the input port, find its destination port with regard to the routing table, transfer the packet to the output port via the switch fabric and finally transmit it on the output line. If a burst of packets destined to the same output port arrives, all packets cannot be transmitted on the fly and thus some of them need to be buffered. A critical aspect of the switch architecture is the placement of buffers. In the Input Queuing (IQ) switch architecture packets arriving from the input lines are queued at the input ports. The packets are then extracted from the input queues to cross the switch fabric and to be forwarded to the output ports. The IQ architecture is used extensively in the design of high-speed routers.

An optical switch operates on fixed-size cells and the time that elapses between two consecutive switching decisions is called a time slot. Each arriving IP packet is divided into cells at the input, and these cells are scheduled across the switch fabric separately. When the last cell arrives, the packet is reassembled and sent on the output link. A major issue in the design of IQ switches is the *scheduler* that controls the access to the switching fabric in order to avoid contention at the input and

the output ports. Every time slot, the scheduler determines the *configuration* of the switch, that is a matching between the inputs and the outputs, and each input queue is allowed to forward a cell to the output port it is matched to, if any. Underlying the design of a wide range of switch architectures is the assumption that the switch configuration time is negligible compared to the cell transmission time. However, in optical switches the configuration time dominates over the cell transmission time. In such a switch, reconfiguring the switching fabric makes it unavailable for a number of time slots equal to the *configuration delay*. Minimizing the effects of reconfiguration overhead on the switch performance requires specially designed schedulers that hold each configuration for several time slots, thus reducing the number of reconfigurations.

The Time Slot Assignment (TSA) problem is to find a conflict-free schedule of cells so as to minimize its makespan. Next we define the problem more formally. The traffic demand is specified by a matrix D . An element $d_{i,j}$ of D represents the number of cells to be transferred from input i to output j . A schedule \mathcal{P} is formed by a sequence of k configurations (permutation matrices) P_1, P_2, \dots, P_k and the corresponding holding times $\phi_1, \phi_2, \dots, \phi_k$. In any configuration, each input is connected with at most one output, and each output is connected with at most one input. For an element $p_{i,j}^q$ of P_q , if $p_{i,j}^q = 1$ then in the configuration P_q input i is connected to output j . A matrix D is said to be *covered* by \mathcal{P} if all the demands are satisfied, i.e. $\forall i, j \sum_{q=1}^k \phi_q \cdot p_{i,j}^q \geq d_{i,j}$. The schedule is called *non-preemptive* if each input is connected at most once to each output, i.e. $\forall i, j \sum_{q=1}^k p_{i,j}^q \leq 1$. Otherwise, the schedule is said to be *preemptive*. Let us denote by C the *configuration delay*, expressed in units of the cell transmission time. The *duration* of \mathcal{P} is the total holding time of all the configurations, that is $\sum_{q=1}^k \phi_q$. The *configuration overhead* of \mathcal{P} is kC . The *length* of \mathcal{P} is the sum of the duration and the configuration overhead, that is $\sum_{q=1}^k \phi_q + kC$. Given a traffic demand matrix and the configuration delay, the goal is to find a schedule of minimum length.

The TSA problem can be reduced to an equivalent bipartite graph matching problem in the following way. We construct a bipartite graph G^D corresponding to the traffic demand matrix D . The nodes of the graph are the input and the output ports of the switch. For each $d_{i,j} > 0$, there is an edge $e = (i, j)$ of weight $w(e) = d_{i,j}$ representing the traffic demand from input i to output j . If the schedule is preemptive, an edge can be sub-divided into a number of parallel edges adding up to the original weight. The cost of a matching is the weight of the heaviest edge plus the configuration delay. The aim is to find

a collection of matchings of minimum cost in which each edge takes part in exactly one of the matchings. We call this problem the Bipartite Scheduling problem.

Gopal and Wong [6] proved that the Preemptive Bipartite Scheduling (PBS) problem is NP-hard for $C = \infty$. However, for $C = 0$ the PBS problem can be solved optimally in a polynomial time using bipartite edge coloring algorithms (see e.g. [12]). Crescenzi et al. [2] considered the PBS problem with *arbitrary* configuration delay. Crescenzi et al. showed that this problem cannot be approximated within a factor of $7/6$ unless $P = NP$ and presented two polynomial time approximation algorithms that have an approximation factor of 2. Afrati et al. [1] derived an algorithm with an improved approximation factor of $2 - \frac{1}{C+1}$.

The Non-Preemptive Bipartite Scheduling (NPBS) problem was studied in the context of matrix decomposition for $C = 0$. Ribeiro et al. [11] solved it optimally using a branch and bound procedure. Prais and Ribeiro [9] proposed a heuristic called Reactive Greedy Randomized Adaptive Search Procedure (GRASP) and compared its performance with the performance of the greedy algorithm. In this paper we consider *arbitrary* configuration delay and for the first time prove *worst-case* guarantees on the performance of the greedy algorithm.

Many existing scheduling algorithms have running time proportional to the number of packets in the input queues, which incurs a significant computation overhead. Since optical switches operate at very high speeds, implementation of these algorithms for batch scheduling may be too costly or even infeasible. In this paper we design algorithms whose complexity depends only on the switch size, which makes them amenable to implementation at wire-speed. Another advantage of non-preemptive scheduling is that all cells of a given packet are *contiguously* transferred from the input queue to the output port saving the reassembly overhead. That is in contrast to preemptive scheduling, where one must keep a packet-reassembly buffer at each output port.

Our results. In this paper we study the NPBS problem with *arbitrary* configuration delay. We extend the results of [6], [2] to show that the NPBS problem is NP-hard for any value of the configuration delay and approximation of the NPBS problem within a ratio smaller than $7/6$ is also NP-hard (see [10], [5] for related results). We demonstrate that the greedy algorithm achieves an approximation factor of exactly 2 for the offline version of the NPBS problem. The running time of the greedy algorithm is $O(N^2 \log N)$ and it requires at most $2N - 1$ different configurations, where N is the number of the input and the output ports. Then we consider the online version of the NPBS problem, where batches of traffic are first accumulated and then scheduled without preemption (batch scheduling). We propose the greedy batch algorithm and show that with a moderate speedup of two it provides strict delay guarantees for any admissible traffic.

Related work. The TSA scheduling problem arises in the context of Satellite Switched Time Division Multiple Access (SS/TDMA) systems. The main research efforts concentrated on the extreme values of the configuration delay, such as $C = 0$ or $C = \infty$. Inukai [7] described algorithms to find an exact decomposition of a demand matrix for the case of $C = 0$. Gopal

and Wong [6] proposed heuristic algorithms for decomposition of a demand matrix using the minimum number of configurations for the case of $C = \infty$.

In the online version of the TSA scheduling problem, the switch periodically accumulates incoming traffic and maps this batch to a set of switch configurations. Towles and Dally [12] studied the batch scheduling problem and presented algorithms for $C = 0$ and $C = \infty$. Li and Hamdi [8] proposed an algorithm self-adjusting with different configuration delay values. Dolev and Kesselman [3] considered scheduling of variable size batches.

The rest of the paper is organized as follows. The model is described in Section II. Our scheduling algorithms are presented in Section III. Section IV contains the performance analysis. In Section V we derive the hardness results. The conclusion appears in Section VI.

II. MODEL DESCRIPTION

We consider a $N \times N$ IQ switch with Virtual Output Queuing (VOQ), where each input maintains a separate queue per each output. Packets, of different size, arrive at input ports, and each packet is labeled with the output port on which it has to leave the switch. Each packet is divided into cells of unit length. Time is slotted and during a time slot exactly one cell can arrive at an input line and exactly one cell can be sent on an output line. If the switch has a *speedup* of S , up to S cells can be removed from any input and up to S cells can be added to any output during a time slot. Changing from one switch configuration to another requires C/S time slots (we assume that C/S is an integer). In the offline setting, we have $S = 1$ and the scheduled cells are immediately sent on the output lines (see Figure 1). In the online setting, we have $S > 1$ and cells are buffered at the inputs before they are scheduled and at the outputs before they are transmitted on the output lines (see Figure 2).

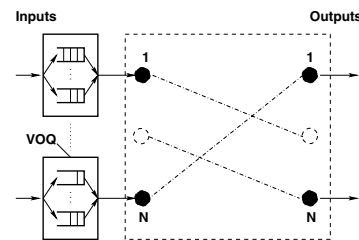


Fig. 1. An example of a switch with a speedup $S = 1$.

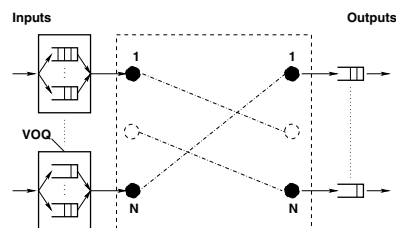


Fig. 2. An example of a switch with a speedup $S > 1$.

The offline version of the problem is as follows.

Definition II.1—NPBS Problem: Given a traffic demand matrix D and the configuration delay C , find a non-preemptive schedule of minimum length that covers D .

In the online setting, the switch gathers the incoming traffic periodically and then schedules the accumulated batches. We follow the model of [8]. The scheduler performs pipelined batch scheduling in three phases (see Figure 3). The length of each phase is a predefined system constant T . Note that the smaller the value of T , the smaller the delay guaranteed by our algorithms. However, the value of T must be large enough to accommodate bursts.

The incoming traffic is gathered during the accumulating phase and stored in the traffic demand matrix D . We call to the accumulated traffic a *batch*. In the scheduling phase, the accumulated batch is scheduled and the cells are stored in the output buffers. Finally, during the transmission phase the batch is transmitted from the output buffers onto the output lines.

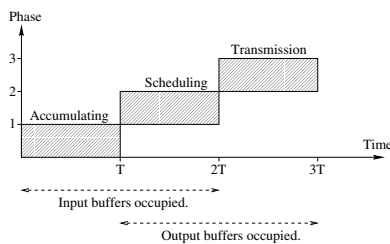


Fig. 3. Batch timeline diagram.

A traffic is said to be *admissible* if any batch can be scheduled in a non-preemptive fashion during T time slots without a speedup (i.e. with $S = 1$). A scheduling algorithm is said to be *stable*, if for any admissible traffic it provides a bound of $3T$ on the worst-case delay of a packet.

III. SCHEDULING ALGORITHMS

The scheduling task is the NPBS problem. We note that the optimal solution depends on the actual value of C . If C is small, the duration of a schedule dominates over the configuration overhead. Conversely, if C is large, minimizing the number of reconfigurations becomes critical.

The greedy algorithm for the offline version of the NPBS problem is presented in Figure 4. In the description of the algorithm we use the formulation of the problem in terms of bipartite graph matching. The main loop assigns each positive entry of the original traffic demand matrix to exactly one of the matchings. When a new matching is initialized, we first find the heaviest uncovered edge e_m . The weight of e_m determines the holding time of the configuration. Then we add edges remained uncovered so far in the order of non-increasing weight if they do not conflict with the current matching. Clearly, the running time of the greedy algorithm is $O(N^2 \log N)$. Observe that since the holding time of a configuration may be greater than the weight of some edges in the corresponding matching, the schedule may contain empty slots. The basic intuition behind the greedy algorithm is that in order to minimize the number of empty slots we try to group entries whose values are close to one another into the same matching. Besides, the number of configurations

is minimized because in each iteration we compute a maximal matching. We will show that the greedy algorithm finds a good trade-off between minimizing the duration against minimizing the configuration overhead of a schedule.

We give below an example of the application of the greedy algorithm. Consider the following traffic demand matrix:

$$D = \begin{bmatrix} 100 & 30 & 0 \\ 15 & 0 & 20 \\ 0 & 10 & 0 \end{bmatrix}.$$

Initially, we add the heaviest edge $(1, 1)$ of weight 100 to the first matching M_1 . The second heaviest edge $(1, 2)$ of weight 30 is not compatible with M_1 . Thus, we add edge $(2, 3)$ of weight 20. Since the fourth heaviest edge $(2, 1)$ of weight 15 conflicts with M_1 , we finish the construction of M_1 by including edge $(3, 2)$ of weight 10. Then we create the second matching M_2 containing of the remaining edges $(1, 2)$ of weight 30 and $(2, 1)$ of weight 15. The obtained schedule consists of two configurations

$$P_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \text{ and } P_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

with holding times $\phi_1 = 100$ and $\phi_2 = 30$ and its length is $120 + 2C$.

The greedy batch algorithm for the online version of the NPBS problem appears on Figure 5. The incoming traffic is periodically gathered and then scheduled as a batch using the greedy algorithm. We perform pipelined scheduling in three phases: accumulating phase, scheduling phase and transmission phase. The length of each phase is a predefined system constant T . At any time slot, there are at most three active batches in the system, i.e. one batch per phase. If all traffic accumulated in the accumulating phase traverses the switch during the scheduling phase and is transmitted during the transmission phase then no packet stays in the switch for more than $3T$ time slots. We will demonstrate that the greedy batch algorithm guarantees this property for any admissible traffic.

IV. PERFORMANCE ANALYSIS

In this section we will analyze the performance of our algorithms.

A. Offline Scheduling

Fix a traffic demand matrix D . Let \mathcal{P}^G be the schedule constructed by the greedy algorithm and \mathcal{P}^O be an optimal schedule covering D . We will demonstrate that the length of \mathcal{P}^G is at most twice that of \mathcal{P}^O for arbitrary configuration delay. Since all configurations in \mathcal{P} are independent, we can consider any permutation of \mathcal{P} . Thus, we assume that all configurations in \mathcal{P}^G and \mathcal{P}^O appear in the order of non-increasing holding time. Let P_q^A and ϕ_q^A denote the permutation matrix and the holding time of the q -th configuration in the schedule \mathcal{P}^A , respectively. The following lemma is the crux of the proof.

Lemma IV.1: The holding time of the q -th configuration in \mathcal{P}^G is at most the holding time of the $\lceil q/2 \rceil$ -th configuration in \mathcal{P}^O , that is $\phi_q^G \leq \phi_{\lceil q/2 \rceil}^O$.

```

Construct the graph  $G^D$  from  $D$  and set  $q = 0$ ;
While there exist uncovered edges in  $G^D$  Do
     $q = q + 1$ ;
    Find the heaviest uncovered edge  $e_m$ ;
     $M_q = \{e_m\}$ ;
     $\phi_q = w(e_m)$ ;
    For each uncovered edge  $e$  in the order of non-increasing weight Do
        If  $e$  is not conflicting with  $M_q$  Then
             $M_q = M_q \cup \{e\}$ ;
        Mark all edges in  $M_q$  as covered;
    Construct a schedule  $\mathcal{P}$  from the matchings found and set  $k = q$ ;

```

Fig. 4. The greedy algorithm.

```

Execute the following phases simultaneously during the interval  $[Ti, T(i+1))$ :
Accumulating phase: Accumulate the arriving traffic into the batch  $B_i$ . (We denote by  $B_i$  the  $i$ -th batch.)
Scheduling phase: Schedule the accumulated traffic demand matrix of the batch  $B_{i-1}$  according to the greedy algorithm.
Transmission phase: Transmit from the output buffers the packets of the batch  $B_{i-2}$ .

```

Fig. 5. The greedy batch algorithm.

Proof: Consider the q -th configuration in \mathcal{P}^G and let $d_{i,j}$ be the heaviest entry covered by P_q^G . Observe that in every iteration of the greedy algorithm we construct a maximal matching by iterating through the uncovered edges in the order of non-increasing weight. Since the edge (i, j) has not been selected by the greedy algorithm till the q -th iteration, it must be the case that at least one edge incident to either i or j of weight greater than or equal to $d_{i,j}$ has been covered in all configurations P_1^G, \dots, P_{q-1}^G . Otherwise, the edge (i, j) would have been already covered. Let x and y be the number of edges of weight at least $d_{i,j}$ (including (i, j) itself) that are incident to i and j , respectively. As we argued, $x + y \geq q$. Therefore, at least $z = \lceil q/2 \rceil$ such edges are incident to one of these nodes. Without loss of generality, assume that it is i . Now consider the schedule \mathcal{P}^O . Since \mathcal{P}^O is non-preemptive, in order to cover all edges incident to i it must contain at least z matchings with weight greater than or equal to $d_{i,j}$. The lemma follows from the fact that all the configurations in \mathcal{P}^O appear in the order on non-increasing holding time. ■

Next we demonstrate that the greedy algorithm achieves an approximation factor of 2.

Theorem IV.2: The approximation factor achieved by the greedy algorithm for the NPBS problem is at most 2 for arbitrary configuration delay.

Proof: By definition, the length of \mathcal{P}^G is $\sum_{q=1}^k \phi_q^G + kC$. According to Lemma IV.1, $\phi_q^G \leq \phi_{\lceil q/2 \rceil}^O$. After summing over all configurations we obtain that

$$\sum_{q=1}^k \phi_q^G \leq \sum_{q=1}^k \phi_{\lceil q/2 \rceil}^O \leq 2 \sum_{q=1}^{\lceil k/2 \rceil} \phi_q^O.$$

Since $\phi_{\lceil k/2 \rceil}^O \geq \phi_k^G > 0$, the number of configurations in \mathcal{P}^O is at least $k/2$. Thus, the length of \mathcal{P}^O is at least

$$\sum_{q=1}^k \phi_q^G / 2 + kC / 2.$$

The theorem follows. ■

Next we demonstrate that approximation factor of the greedy algorithm is at least $2 - 1/n$, where n is the maximum node degree in G^D .

Theorem IV.3: The approximation factor achieved by the greedy algorithm for the NPBS problem is at least $2 - 1/N$ for $C = \infty$.

Proof: Consider the following scenario. Each of the input ports $1, \dots, N-1$ has two packets destined to each of the output ports $1, \dots, N-1$ and one packet destined to output port N . We call these input ports *active*. The schedule of the greedy algorithm consists of $2N-1$ configurations since during the first $N-1$ iterations it schedules only packets destined to output ports $1, \dots, N-1$ and then it takes $N-1$ additional iterations to schedule all packets destined to output port N . On the other hand, the optimal schedule contains only N configurations, in each of which packets from $N-2$ active input ports are scheduled to $N-2$ output ports among output ports $1, \dots, N-1$ and a packet from the remained active input port is scheduled to output port N . The theorem follows since $C = \infty$. ■

We also show that the schedule of the greedy algorithm contains at most $2N-1$ configurations.

Theorem IV.4: The schedule constructed by the greedy algorithm contains at most $2N-1$ configurations.

Proof: Suppose toward a contradiction that $k > 2N-1$ and consider the edge (i, j) covered at the $2N$ -th configuration. Note that the total number of edges other than (i, j) incident to i and j is at most $2N-2$. It follows that edge (i, j) must

have been selected by the greedy algorithm at the $(2N - 1)$ -th iteration, which contradicts to our assumption. ■

B. Online Scheduling

In batch scheduling, an internal speedup $S > 1$ is required by the greedy batch algorithm to ensure that a batch of admissible traffic can be completely scheduled during the frame time T . The next theorem shows that the greedy batch algorithm is stable for any admissible traffic with a speedup of at least two.

Theorem IV.5: The greedy batch algorithm is stable for any admissible traffic with a speedup $S \geq 2$.

Proof: The proof is by induction on the batch index. We claim that all packets of the batch B_i leave the input and the output buffers by time $T(i + 2)$ and $T(i + 3)$, respectively.

The basis is the first batch B_0 . Note that B_0 is accumulated during T time slots. By the definition of admissible traffic, B_0 can be scheduled during T time slots without a speedup. Theorem IV.2 implies that the greedy algorithm with a speedup $S \geq 2$ will schedule B_0 during the frame time T . We also argue that B_0 can be transmitted on the output lines during T time slots. If it is not the case and some output port receives traffic that requires for transmission more than T time slots then B_0 cannot be scheduled during T time slots without a speedup.

Now suppose that the induction hypothesis holds for all batches up to the batch B_j and let us show that it is also satisfied for B_j . By the induction hypothesis, all packets of the preceding batches leave the input and the output buffers by time $T(j + 1)$ and $T(j + 2)$, respectively. The claim regarding B_j follows by an argument similar to the previous case. ■

Thus, for any admissible traffic the maximal delay of a packet under the greedy batch algorithm is bounded by $3T$ provided that the switch has a speedup $S \geq 2$.

V. HARDNESS RESULTS

In this section we extend the impossibility results of [6], [2] to the NPBS problem. We show that the NPBS problem is NP-hard for any value of the configuration delay and approximating it within a ratio smaller than $7/6$ is also NP-hard. In [9] it is mentioned, without a proof, that the NPBS problem is NP-hard for $C = 0$. Remember that for $C = 0$, the PBS problem can be solved optimally in a polynomial time.

Both [6] and [2] use reduction from the Restricted Timetable Design problem [4] to the PBS problem. In the Restricted Timetable Design problem, teachers must be assigned to classes within three hours. The following observation is due to the fact that in the optimal solution to the corresponding instance of the PBS problem there are three disjoint matchings, each of which corresponds to the assignment of teachers for one of three hours.

Observation 1: There exists an optimal schedule that is *non-preemptive* for the PBS instance constructed in Theorem 3 of [6] and Theorem 4 of [2].

In the construction of [6], it is assumed that $C = \infty$. However, in this construction the optimal *non-preemptive* schedule is the same for all values of C . The next theorem follows from Observation 1.

Theorem V.1: The NPBS problem is NP-hard for any value of the configuration delay.

In [2] fractional weights are considered, but we can always scale them to be integers. Observation 1 also implies the following theorem.

Theorem V.2: Approximating the NPBS problem with fractional weights within a factor of $7/6$ is NP-hard.

VI. CONCLUSION

With the fast development of the Internet, optical switching technologies are becoming an alternative to electronic switches due to their capacity and scalability. However, optical switches have a large configuration delay, which complicates the scheduling problem. In order to design efficient scheduling algorithms, one has to consider this overhead. On the other hand, only low complexity scheduling algorithms are amenable to implementation in high speeds. In this paper we study the NPBS problem, where each input queue is connected to each output port in at most one configuration. In non-preemptive scheduling, the complexity of the scheduler does not depend on the number of packets in the input queues, which makes it attractive for batch scheduling where a large number of packets is accumulated within a batch. We show that a simple greedy algorithm achieves an approximation factor of 2 for the offline version of the NPBS problem and apply it to online batch scheduling. The running time of the greedy algorithm is $O(N^2 \log N)$ and it requires at most $2N - 1$ different configurations. We also demonstrate that the NPBS problem is NP-hard and hard to approximate within a ratio smaller than $7/6$. We believe that the proposed algorithms can be successfully deployed in real architectures since they are simple to implement and provide worst-case performance guarantees.

REFERENCES

- [1] F. Afrati, T. Aslanidis, E. Bampis, and I. Milis, "Scheduling in Switching Networks with Set-up Delays," *ALGOTEL'2002*, Muze, France, May 2002.
- [2] P. Crescenzi, X. Deng, C. Papadimitriou, "On Approximating a Scheduling Problem," *Journal of Combinatorial Optimization*, Vol. 5, pp. 287-297, 2001.
- [3] S. Dolev and A. Kesselman, "Bounded Latency Scheduling Scheme for ATM Cells," *Journal of Computer Networks*, Vol. 32(3), pp 325-331, March 2000.
- [4] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM J. Computing*, Vol. 5, pp. 691-703, 1976.
- [5] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP Completeness," *W. H. Freeman*, San Francisco, 1979.
- [6] 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, June 1985.
- [7] T. Inukai, "An Efficient SS/TDMA Time Slot Assignment Algorithm," *IEEE Transactions on Communication*, Vol. 27, pp. 1449-1455, 1979.
- [8] X. Li and M. Hamdi, " λ -ADJUST Algorithm for Optical Switches with Reconfiguration Delay," in *Proc. of International Conference on Communications (ICC'03)*, 2003.
- [9] M. Prais and C. C. Ribeiro, "Reactive GRASP: An Application to a Matrix Decomposition Problem in TDMA Traffic Assignment," *INFORMS Journal on Computing*, Vol. 12, pp. 164-176, 2000.
- [10] F. Rendl, "On the complexity of decomposing matrices arising in satellite communication," *Operations Research Letters*, Vol. 4, pp. 5-8, 1985.
- [11] C. C. Ribeiro, M. Minoux and M. C. Penna, "An Optimal Column-Generation-with-Ranking Algorithm for Very Large Scale Set Partitioning Problems in Traffic Assignment," *European Journal of Operational Research*, Vol. 41, pp. 232-239, 1989.
- [12] 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*, pp. 342-351, June 2002.