An Optimal Algorithm for Time-slot Assignment in SS/TDMA Satellite Systems

Xili Wan[1] Feng Shan[1,2] Xiaojun Shen[1]
[1]University of Missouri, Kansas City, 5100 Rockhill Rd. Kansas City, MO, USA
[2]Southeast University, No. 2 Sipailou, Nanjing, China
Email: {xiliwan,shanf,shenx}@umkc.edu

***Abstract*—Packet scheduling in switches plays an important role in providing guaranteed service to meet Quality-of-Service (QoS) requirements. In this paper, we revisit a historical Maximum Traffic Time-slot Scheduling (MTTS) problem for both circuit- and packet-switched traffic in SS/TDMA satellite systems proposed two decades ago by Bonuccelli et al.. They developed a two-step approach to solve the MTTS problem, which scheduled the circuit-switched traffic first into TDMA frames and then inserted packets from packet-switched traffic into empty slots in these scheduled TDMA frames as many as possible. This problem was referred as Incremental Time-slot Assignment (ITSA) Problem in their paper, and was proved to be NP-hard. Therefore, MTTS problem seems to be NP-hard due the the NP-hardness of ITSA problem induced by their two-step approach. However, in this paper, we show that MTTS problem is NOT NP-hard, since there exists an optimal algorithm for this problem. By aggregating the traffic in scheduled TDMA frames, an optimal algorithm based on network flow method is developed for MTTS problem, which can solve the MTTS problem in polynomial time. Theoretic proof for the optimality of our algorithm is given and simulations results further validate the correctness of our algorithm.**

I. Introduction

Internet has been witnessed with the rapid growth of video on call, multimedia streaming, online games, voice calls etc. These applications requires Quality-of-Service(QoS) to guarantee their packet delivery rates. Packet scheduling in switches is one of the important aspects to provide guaranteed rate service for these time-sensitive applications. Input-queued (IQ) switches have gained lots of attention, due to its scalability. To meet QoS requirement in IQ switches, extensive studies have been made to design packet scheduling algorithms in decades [1]–[9].

Bonuccelli et al. [1] studied the time-slot scheduling problem for satellite communication. They showed the hardness of the induced scheduling problem (Incremental Scheduling Time-slot Assignment) in their two-step approach and proposed three heuristic algorithms by applying network flow algorithm and Lagrangian relaxation technique. Philp et al. [10] studied real-time periodic packets scheduling in satellite-switched system and presented algorithms based on Earliest Deadline First (EDF) and Minimum Laxity First (MLF) policies. Later, Giles et al. [11] proposed Nested Period Scheduling (NPS) and EDF Earliest-Arrival-First (EDF-EAF) algorithms to schedule all traffic when each period in the traffic was divided into longer periods and the utilization of each link was less than 1. Their algorithms can also be extended to work with arbitrary packet periods, under the condition that the link utilization is less than $1/4$. In paper [12], Dynamic Minimum Laxity First (DMLF) was proposed for scheduling real-time packets in optical networks. DMLF is similar to MLF, except for dynamically computed laxities.

Li et al. [5] presented a frame-based scheduling algorithm with 100% throughput under bounded end-to-end delay and delay jitter. Based on Birkhoff-Von Neumann method to decompose the traffic rate matrix, Chang et al. [8] provided an algorithm which achieved the guaranteed rate service by adopting the Packetized Generalized Processor Sharing (PGPS) or the Weighted Fair Queueing (WFQ) method to generate schedules. Moreover, they also proposed algorithms to achieve both guaranteed rate service and best-effort service by mapping the problem to a maximum flow problem, which made their schemes more practical [7].

In this paper, we revisit a historical scheduling problem: Maximum Traffic Time-slot Scheduling (MTTS) problem for both Circuit- and Packet-switched Traffic (TSCPT) in SS/TDMA satellite systems, which was proposed two decades ago [1]. This problem seeks for an assignment of maximum packets from packet-switched Traffic to circuit-switched traffic, such that these packets can be transmitted with circuit-switched traffic together. In paper [1], a two-step approach was developed for MTTS problem, which first decomposed the circuit-switched traffic into TDMA frames and then inserted packets from packet-switched traffic into empty slots in these TDMA frames as many as possible. Then, a scheduling problem, which finds an assignment of maximum packets from packet-switched to TDMA frames, is induced in their two-step approach. This scheduling problem was referred as Incremental Time-slot Assignment (ITSA) problem in their paper, and proved to be NP-hard. Therefore, it seems that MTTS problem is NP-hard due the NP-hardness of ITSA problem. However, in this paper, we show that their exists an optimal algorithm for MTTS problem which runs in polynomial time. Thus, MTTS problem is NOT NP-hard. Our proposed algorithm is based on network flow method, and it can obtain the optimal schedules for MTTS problem in polynomial time. Theoretical proof for our algorithm is given and simulation results further validate the correctness of our algorithm.

This paper is structured as follows. In section II, we first review the background of scheduling for circuit- and packet-switched traffic and give the formal definition of MTTS problem. Section III discusses the two-step approach proposed in [1] and presents our optimal algorithm for MTTS problem. Theoretical proof for optimality is also presented in this section. Section IV reports our simulation result and conclusion is made in section V.

II. Maximum Traffic Time-slot Scheduling Problem

In this section, we briefly introduce the Maximum traffic Time-slot Scheduling (MTTS) problem for both circuit- and packet-switched traffic, and the two-step approach proposed in [1] including its induced Incremental Time-slot Assignment (ITSA) problem. Related notations and definitions are also given in this section. For reader's convenience, we follow definitions and notations given in [1].

Bonuccelli et al. [1] introduced the maximum traffic time-slot scheduling (MTTS) problem for both circuit- and packet-switched traffic in SS/TDMA satellite systems. Although the background of MTTS problem is the Satellite-Switched time division multiple access (SS/TDMA) system, MTTS problem itself is also applicable for input-queued (IQ) switches, since their abstracted models are essentially the same from the scheduling point of view. Consider an N×N IQ switch, and define *traffic matrix* R to be a N×N matrix with non-negative integers. For a matrix R, each entry r_{ij}^R $(i,j \in N)$ defines the number of packets to be transmitted from input port i to output port j in a switch. $|R| = \sum_{i,j} r_{ij}$ is the total number of packets in a traffic matrix R. Define $L(R) = max(max_j(\sum_i R_{ij}), max_i(\sum_j R_{ij})), (i,j \in N)$ to be the *duration* of traffic matrix R. If a traffic matrix S has at most one entry in any row or column, then it is a *switching matrix*.

Conflicts will occur when any output port is scheduled to receive more than one packet. It has been shown that all packets in a switching matrix S can be transmitted without conflicts in $L(S)$ consecutive time slots [13]. If a traffic matrix R can be decomposed into a set of switching matrixes $Q = Q_1, Q_2, \cdots, Q_m$ and $R = Q_1 + Q_2 + \cdots + Q_m$, then Q is the *transmission schedule* (or time-slot assignment) for R and $L_R = \sum_{i=1}^m L(Q_i)$ is the total transmission length (or the total number of time slots) of this schedule. A transmission schedule is optimal if the length of this schedule is minimized.

We use traffic matrixes P_c and P_p to denote circuit- and packet-switched traffic respectively. Next, we'll formally define the Maximum traffic Time-slot Scheduling (MTTS) problem.

Definition *Maximum Traffic Time-slot Scheduling (MTTS) Problem*: Given the circuit-switched traffic matrix P_c and the packet-switched traffic matrix P_p, MTTS problem asks for a non-conflict schedule such that all packets in circuit-switched traffic P_c and maximum number of packets from packet-switched traffic P_p can be scheduled to transmit in $L(P_c)$ time slots.

The MTTS problem seeks to find maximum packets from packet-switched traffic which will be promoted to circuit-switched traffic, such that the both of the promoted packets and the circuit-switched traffic can be all scheduled to transmit within the length of transmission schedule for the circuit-switched traffic.

III. Optimal Algorithm

In this section, we first discuss the two-step approach proposed in [1], and then present our optimal algorithm for MTTS problem, followed by the theoretical proof for its optimality. An instance from [1] to illustrate our algorithm is also given in this section.

A. Analysis of Two-step Approach for MTTS Problem

To solve MTTS problem, Bonuccelli et al. [1] adopted a two-step approach to solve this problem: First, construct schedules for static circuit-switched traffic P_c by decomposition of P_c into a set of switching matrices $S_1, S_2, \cdots, S_m$, using known BCW algorithm in [13]. Here, a set of switching matrixes $S_1, S_2, \cdots, S_m$ denotes the predefined transmission schedule for circuit-switched traffic P_c. Note that $P_c = S_1 + S_2 + \cdots + S_m$ according to BCW algorithm in [13]. Next, for each switching matrix S_i, fill the gaps (unused slots) with packets from packet-switched traffic P_p as many as possible.

This scheduling problem induced in the second step (filling the gaps with maximal number of packets from P_p) was referred as *Incremental Scheduling Time-slot Assignment* (ISTA) problem in their paper, which was shown to be NP-hard. Three heuristic algorithms are proposed in their paper, all of which follow the two-step approach. In other words, after decomposition of circuit-switched traffic P_c into switch matrixes, their two-step approach asks for an algorithm which is able to determine a set of packets R_p selected from traffic matrix P_p and insert these packets into at least one of the switching matrixes $S_1, S_2, \cdots, S_m$, such that the new switching matrixes $S'_1, S'_2, \cdots, S'_m$ can still be scheduled in $L(P_c)$ time slots and $|R_p|$ is maximized. Unfortunately, finding such an algorithm is NP-hard as they have proved in their paper.

In fact, ISTA problem is different from MTTS problem. After the decomposition of P_c, a pre-defined schedule $S_1, S_2, \cdots, S_m$ for circuit-switched traffic is attained. This pre-defined schedule remains unchanged in the final schedule for both kinds of traffic, since packets from packet-switched traffic are inserted to empty slots in the pre-defined schedule for circuit-switched traffic. Therefore, ISTA actually asks for the maximization of $|R_p|$, under the implicit requirement that $S_1, S_2, \cdots, S_m$ are fixed (or given). However, from the definition of the MTTS problem, it does not have such requirement. Therefore, MTTS problem may not be NP-hard, although ISTA problem which is induced in the two-step approach is NP-hard. Actually, MTTS problem is solvable in polynomial time, if we don't fix the schedule for the circuit-switched traffic. We'll present our algorithm which achieves the optimal result for MTTS problem in next subsection. Our algorithm guarantees that the maximum number of packets from packet-switched traffic will be promoted to the existing circuit-switched traffic, such that both of promoted packets and circuit-switched packets can be scheduled within $L(P_c)$ time slots.

Algorithm 1 maximalSchedule(P_c, P_p)

```
Require:
    Circuit-Switched traffic, represented by traffic matrix $P_c$;
    Packet-Switched traffic, represented by traffic matrix $P_p$ ;
Ensure:
    A set of packets from $P_p$, represented by $P_s$;
1: For traffic matrix $P_p$, construct a directed bipartite Graph $G(U, V, E)$, where nodes $u_i$ in $U$ represent the rows of $P_p$, and nodes $v_i$ in $V$ correspond the columns of $P_p$. Edge set $E$ consists of an edge between $u_i$ and $v_i$ if $d_{ij}^{P_p} \neq 0$. The capacity of edge $(u_i, v_i)$ is set to be $d_{ij}^{P_p}$.
2: Set $D = L(P_c)$. For each row $i$, compute row vacancy degree $e_i = max\{0, D - \sum_j d_{ij}^{P_c}\}$ and column vacancy degree $f_j = max\{0, D - \sum_i d_{ij}^{P_c}\}$;
3: Add a source node $T$ and a sink node $K$ to Graph $G(U, V, E)$. Construct a flow network $G'$ by connecting the source node $T$ to each $u_i$ and connecting each $v_i$ to the sink node $K$. Set the capacity of edge $(T, u_i)$ and $(v_i, K)$ as $e_i$ and $f_j$, respectively.
4: Apply network flow algorithm [14] on the constructed graph $G'$ to find a maximal network flow $f$.
5: This maximal flow $f$ indicates packets that can be promoted to matrix $P_c$. Define $f_{ij}$ as the resulting maximal flow on edge $(u_i, v_j)$. Particularly, a flow $f_{ij}$ on edge $(u_i, v_j)$ corresponds to promote $f_{ij}$ packets in the row $i$ and column $j$ of matrix $P_c$ to $P_p$.
6: Construct a $N \times N$ matrix $P_s$, the entry of which on row $i$ and column $j$ is $f_{ij}$, if $f_{ij}$ exists, otherwise 0.
7: return $P_s$;
```

B. Optimal Algorithm for MTTS Problem

In this subsection, an algorithm based on network flow algorithm is presented to solve MTTS problem. Before giving our detailed description of our algorithm, we first state a few definitions.

Let matrix P_c denote the total circuit-switched traffic, which takes $L(P_c)$ time slots to transmit all of its packets in $L(P_c)$ according to Birkhoff-Von Neumann theorem. Traffic matrix P_p represents the packet-switched traffic and $d_{ij}^{P_p}$ is an entry of P_p on row i and column j. Let P_s be a matrix for the selected packets selected from P_p. The MTTS problem is to determine a set of packets P_s, such that $P_c + P_s$ can be scheduled in $L(P_c)$ time slots without conflicts and $|P_s|$ is maximized.

Next, we present our algorithm *maximalSchedule(P_c, P_p)* to completely solve MTTS problem in polynomial time. The details of *maximalSchedule* algorithm is described in Algorithm 1.

By preprocessing the input traffic P_c, the algorithm *maximalSchedule* successfully transforms MTTS problem to be a maximum flow problem, which can be solved by existing network flow algorithms optimally. The algorithm *maximalSchedule* returns a traffic matrix P_s indicating the packets from P_p that can be promoted to matrix P_c. The new traffic matrix $P_c' = P_c + P_s$ can be decomposed into a linear combination of switch matrixes by applying BCW algorithm [13], so that all traffic in the P_c' can be scheduled without conflicts in $L(P_c)$ time slots. Next we give our proof for the optimality of our algorithm.

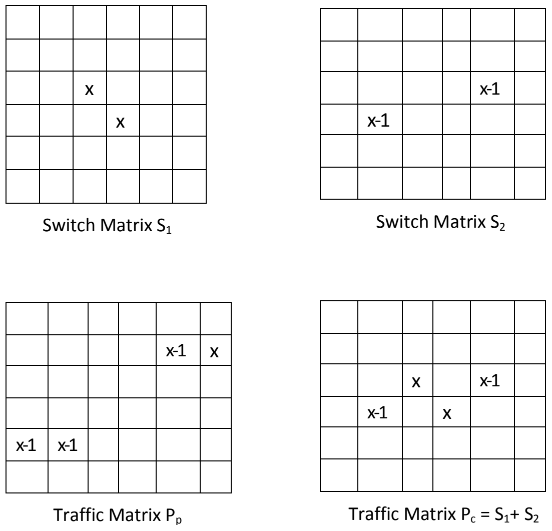

Fig. 1. Instance: Switching Matrixes and Traffic Matrix

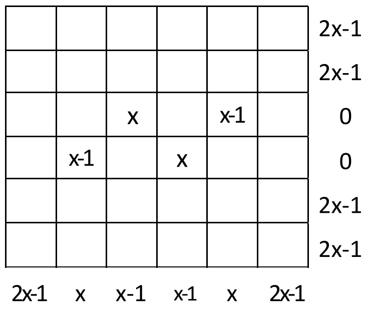

Fig. 2. Vacancy Degree for Each Row and Column

Theorem 1: The algorithm *maximalSchedule* solves MTTS problem with optimal result in polynomial time.

Proof: The capacity on edges (T, u_i) and (v_j, K), where $i, j \in [1, N]$, are the upper bounds of vacant slots for the row i and column j in matrix P_c, respectively. These two upper bounds make sure that the promoted packets from P_p will not overflow the duration of matrix S, such that the new matrix $P_c' = P_c + P_s$ can be schedule in $L(P_c)$ time slots by existing BCW algorithm [13]. Besides, the maximal flow achieved in step 5 corresponds the maximal amount of packets selected from matrix P_p. Therefore, the total number of packets scheduled to be transmitted in $L(P_c)$ time slots is maximized, while packets from P_c are all scheduled. The time complexity of algorithm *maximalSchedule* is dominated by the network flow algorithm applied in step 4, the complexity of which is $O(N^3)$ by the classical push-relabel method [14]. Hence, *maximalSchedule* algorithm solves MTTS problem with optimal result in polynomial time. ■

To better understand *maximalSchedule* algorithm, an instance from [1] is given to illustrate our algorithm. Figure 1 shows this instance including switching matrixes $S =$

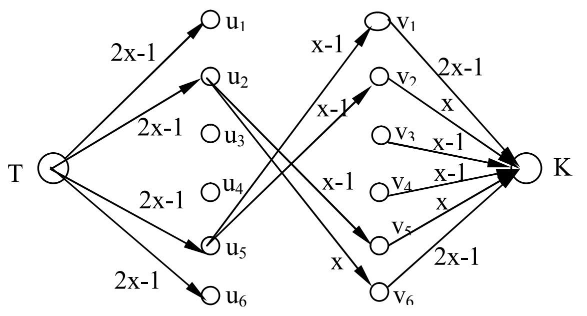

Fig. 3. The Constructed Network Flow Graph for the Instance

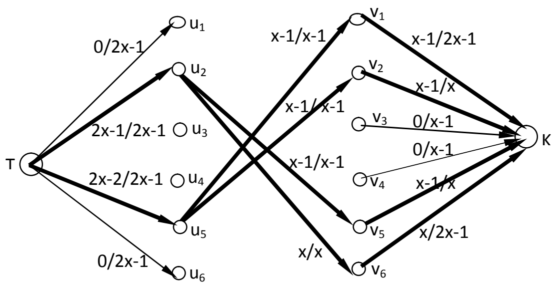

Fig. 4. The Maximal Network Flow for the Instance

$\{S_1, S_2\}$, which is the predefined schedule for circuit-switched traffic $P_c = S_1 + S_2$, and a traffic matrix P_p for packet-switched matrix. Figure 2 shows the vacancy degree of each row and column for the circuit-switched traffic matrix P_c.

A network flow graph constructed for this instance is shown in Figure 3 according to *maximalSchedule* algorithm. The resulting maximum network flow shown by bold edges in Figure 4 indicates that all packets in P_p can be inserted into P_c. In other words, the final output of *maximalSchedule* algorithm for this instance is $P_s = P_p$, which is obviously the optimal result for this instance. Note that, this instance serves as an example in [1] to show their heuristic algorithms can not find the optimal schedule.

As shown above, *maximalSchedule* algorithm guarantees the optimal result for MTTS problem. Therefore, MTTS problem is *NOT* NP-hard, and it is polynomial solvable. Be aware that the NP-hardness proof of ISTA problem induced in the two-step approach in [1] is correct. However, the NP-hardness of ISTA problem doesn't mean the NP-hardness of MTTS problem. Hence, we can say that their two-step approach is a NP-hard way to solve MTTS problem, which definitely can't achieve the optimal result.

C. Comparison of our optimal algorithm and three heuristic algorithms in [1]

For the MTTS problem, all three heuristic algorithms proposed in [1], which are *Local-Optimal*, *Weighted Entries*, and *Lagrangian-Relaxation* algorithms, follow the *decompose-promote* two-step approach: First, decompose the static circuit-switched traffic P_c into a set of switching matrices; Second, promote packets from packet-switched traffic P_p into the gaps (unused slots) in these switching matrices. Unfortunately, it is NP-hard to promote maximum number of packets from P_p to the unused slots in the resulting switching matrices. Therefore, optimal result can not be achieved by this two-step approach. Different from these three heuristic algorithms, our *maximalSchedule* algorithm adopts promote-decompose approach:First, promotes maximum packets from P_p to P_c, such that the duration of the resulting matrix $P_c' = P_c + P_s$ is equal or less than $L(P_c)$; Second, by Birkhoff-Von Neumann theorem, the resulting matrix P_c' can be fully decomposed into a set of switching matrices in $L(P_c)$ time slots, which forms the final schedule of the MTTS problem.

D. MTTS Problem Extension

In MTTS problem, it only requires the traffic to be scheduled in $L(P_c)$ time slots. In other words, we can say that a deadline $D^{P_c} = L(P_c)$ is imposed on the traffic P_c, but not on traffic P_p. In addition to solving MTTS problem, we summarize extension of MTTS problem by considering that traffic P_p also has a deadline D^{P_p} ($D^{P_p} \geq D^{P_c}$), which means the packets in traffic P_p need to be scheduled before the deadline D^{P_p}.

First we define the non-overloaded conditions for traffic matrixes with deadlines. For the matrix P_c with a given deadline D^{P_c}, the non-overloaded conditions are defined as the following:

$$\sum_{j=1}^{N} d_{ij}^{P_c} \leq D^{P_c} \tag{1}$$

$$\sum_{j=1}^{N} d_{ij}^{P_c} \leq D^{P_c} \tag{2}$$

For the matrix P_p with a given deadline D^{P_p} ($D^{P_p} \geq D^{P_c}$), the non-overloaded conditions are defined as the following:

$$\sum_{j=1}^{N} d_{ij}^{P_p} \leq D^{P_p} - D^{P_c} \tag{3}$$

$$\sum_{j=1}^{N} d_{ij}^{P_p} \leq D^{P_p} - D^{P_c} \tag{4}$$

For both traffic matrixes P_c and P_p, the non-overloaded conditions are defined as the following:

$$\sum_{i=1}^{N} d_{ij}^{P_c} + \sum_{i=1}^{N} d_{ij}^{P_p} \leq D^{P_p} \tag{5}$$

$$\sum_{j=1}^{N} d_{ij}^{P_c} + \sum_{j=1}^{N} d_{ij}^{P_p} \leq D^{P_p} \tag{6}$$

The non-overloaded conditions (1) and (2) say that the number of packets for each input or output port should be less than or equal to the given deadline of the traffic matrix. Obviously, the traffic matrix P_c with deadline $D^{P_c} = L(P_c)$ satisfies non-overloaded conditions (1) and (2), according to the definition of $L(P_c)$. If traffic matrix P_p satisfies non-overloaded conditions (3) and (4), MTTS problem with deadlines is easy to solve, since each traffic matrix can be scheduled individually.

However, if conditions (3) and (4) are not satisfied by P_p, it's still possible to schedule both traffic matrixes by promoting a subset P_s of packets from P_p to P_c. Then MTTS problem extension becomes the problem to determine the optimal subset P_s with deadline constraints. We consider this problem with the following two cases:

Case 1: Traffic matrixes P_p and P_c meet the non-overloaded conditions (5) and (6). For this case, an algorithm *Promote* is given in paper [15] to find the optimal subset P_s, if it exits. If not, the algorithm *Two_Class_Scheduling* is proposed to drop the minimum packets from P_p so that P_p is still able to be scheduled before its deadline.

Case 2: The non-overloaded conditions (5) and (6) are not satisfied by traffic P_p and P_c. In this case, the total traffic of P_p and P_c is overloaded. MTTS problem is transformed to drop minimum number of packets from both P_p and P_c so that the remaining traffic can be scheduled within their deadlines. Paper [16] shows that this problem is NP-hard for this case and proposes an algorithm to drop packets so that the remaining traffic is non-overloaded.

IV. Evaluations

In this section, we compare the performance of three heuristic algorithms developed in [1], which are *Local-Optimal* algorithm, *Weighted Entries* algorithm, and *Lagrangian-Relaxation* algorithm, with our *maximalSchedule* algorithm which is optimal. We implemented these four algorithms in MATLAB.

The traffic matrix P_p and switch matrixes S_i, $i \in \{1, 2, \cdots, m\}$ (predefined schedule as the input for ISTA problem) with $P_c = \sum_i^m S_i$ are randomly generated under uniform distribution. In our first simulation, we randomly generated three switch matrixes, and then we performed the simulation for the performance of three heuristic algorithms and the optimal algorithm as the dimension of the switch matrixes grows. Figure 5 shows the result of this simulation. Not surprisingly, our optimal algorithm achieved the best performance, compared to the three heuristic algorithms. Next, we fixed the dimension of the switch matrixes to four, and tested the performance of all four algorithms as the number of switch matrix grows. Figure 6 reports the result for this simulation. Again, our optimal algorithm performs the best, since our algorithm is guaranteed to have the optimal result.

V. Conclusions

In this paper, we revisit the historical scheduling problemMaximum traffic Time-slot Scheduling (MTTS) problem proposed decades ago. This problem seems to be NP-hard, due to the NP-hardness of ISTA problem induced in the two-step approach proposed to solve MTTS probem in [1]. However, we show that MTTS is polynomial solvable. By using a preprocessing technique, we propose a network flow based algorithm to completely solve this problem. Simulation results validates that our optimal algorithm outperforms all previous heuristic algorithms for this problem.

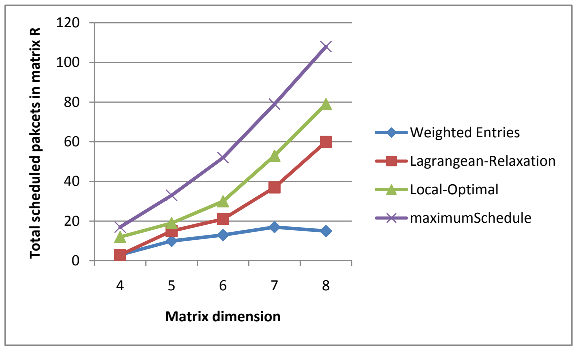

Fig. 5. Total scheduled packets in traffic matrix D as the dimension of matrix increases, with three switch matrixes

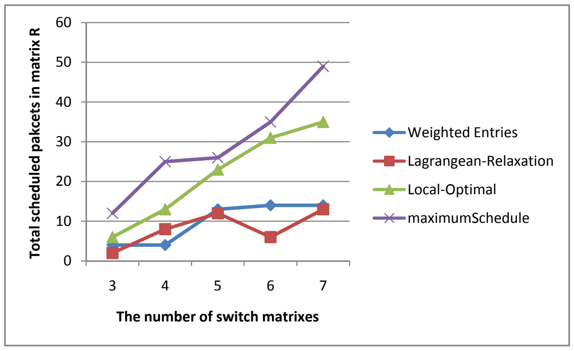

Fig. 6. Total scheduled packets in traffic matrix D as the number of switch matrixes grows, with matrixes dimension = 4

References

[1] M. Bonuccelli, I. Gopal, and C. Wong, "Incremental time-slot assignment in ss/tdma satellite systems," *Communications, IEEE Transactions on*, vol. 39, no. 7, pp. 1147 –1156, jul 1991.

[2] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100input-queued switch," in *Proceedings of the Fifteenth annual joint conference of the IEEE computer and communications societies conference on The conference on computer communications - Volume 1*, ser. INFOCOM'96. IEEE Computer Society, 1996, pp. 296–302.

[3] P. Giaccone, B. Prabhakar, and D. Shah, "Randomized scheduling algorithms for high-aggregate bandwidth switches," *Selected Areas in Communications, IEEE Journal on*, vol. 21, no. 4, pp. 546 – 559, may 2003.

[4] A. Kam and K.-Y. Siu, "Linear-complexity algorithms for qos support in input-queued switches with no speedup," *Selected Areas in Communications, IEEE Journal on*, vol. 17, no. 6, pp. 1040 –1056, jun 1999.

[5] S. Li and N. Ansari, "Input-queued switching with qos guarantees," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, mar 1999, pp. 1152 –1159 vol.3.

[6] K. F. Chen, E. H. M. Sha, and S. Q. Zheng, "Fast and noniterative scheduling in input-queued switches: Supporting qos," *Comput. Commun.*, vol. 32, no. 5, pp. 834–846, Mar. 2009.

[7] C.-S. Chang, W.-J. Chen, and H.-Y. Huang, "Birkhoff-von neumann input buffered crossbar switches," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, Mar 2000, pp. 1614–1623 vol.3.

[8] ——, "On service guarantees for input-buffered crossbar switches: a capacity decomposition approach by birkhoff and von neumann," in *Quality of Service, 1999. IWQoS '99. 1999 Seventh International Workshop on*, 1999, pp. 79 –86.

[9] A. Charny, P. Krishna, N. Patel, and R. Simcoe, "Algorithms for providing bandwidth and delay guarantees in input-buffered crossbars with speedup," in *Quality of Service, 1998. (IWQoS 98) 1998 Sixth International Workshop on*, may 1998, pp. 235 –244.

[10] I. R. Philp and J. W. S. Liu, “Ss/tdma scheduling of real-time periodic messages,” in *Intl. Conf. on Telecomm. Systems*, 1996, pp. 244–251.

[11] J. Giles, B. Hajek, and P. B. Hajek, “Scheduling multirate periodic traffic in a packet switch,” 1997.

[12] M. Bonuccelli and M. Clo, “Scheduling of real-time messages in optical broadcast-and-select networks,” *Networking, IEEE/ACM Transactions on*, vol. 9, no. 5, pp. 541 –552, oct 2001.

[13] G. Bongiovanni, D. Coppersmith, and C. Wong, “An optimum time slot assignment algorithm for an ss/tdma system with variable number of transponders,” *Communications, IEEE Transactions on*, vol. 29, no. 5, pp. 721 – 726, may 1981.

[14] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: theory, algorithms, and applications*. Prentice-Hall, Inc., 1993.

[15] Y. Lee, J. Lou, J. Luo, and X. Shen, “An efficient packet scheduling algorithm with deadline guarantees for input-queued switches,” *IEEE/ACM Trans. Netw.*, vol. 15, no. 1, pp. 212–225, Feb. 2007.

[16] X. Shen, J. Lou, W. Liang, and J. Luo, “Deadline guaranteed packet scheduling for overloaded traffic in input-queued switches,” *Theor. Comput. Sci.*, vol. 409, no. 3, pp. 477–485, Dec. 2008.