Quick Birkhoff-von Neumann Decomposition Algorithm for Agile All-Photonic Network Cores

Cheng Peng, Gregor v. Bochmann and Trevor J. Hall
Centre for Research in Photonics, School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada
{cpeng, bochmann, thall}@site.uottawa.ca

***Abstract*—This paper presents a simple and efficient algorithm for timeslot allocation in agile all-photonic network (AAPN) cores working under a time division multiplexing (TDM) mode, called the Quick Birkhoff-von Neumann Decomposition Algorithm (QBvN). The time complexity of QBvN can reach $O(N\eta)$ for a $N \times N$ switch with a TDM frame size of η. Another version of QBvN, called QBvN-cover, is also proposed to provide guaranteed scheduling with configuration overhead. For QBvN-cover, the bound of the number of generated switch configurations is provided and hence the necessary speedup for AAPN cores. Under stream-type, continuous bit rate traffic, QBvN-cover shows superior delay performance compared with other heuristics in the literature. Although QBvN-cover is unlike other BvN algorithms that use a service matrix as input, we show that service matrix construction from traffic demand is necessary for QBvN-cover to perform well.**

Keywords- Switching Systems, Optical Networks, Scheduling Algorithms, Time Division Multiplexing

I. INTRODUCTION

The term "agility" in optical networks describes the ability to deploy bandwidth on demand at a fine granularity that allows carriers to provision and deploy services rapidly, which radically increases network efficiency and brings to the users much higher performance at reduced cost. One possible scheme to provide such agility in WDM networks is multiplexing in the time domain, which is based on the principle of Time Division Multiplexing (TDM) [1]. In this context, optical switches along lightpaths must be scheduled to reconfigure every one to a few timeslots for bandwidth sharing. Centrally controlled Agile All-Photonic Networks (AAPN) [1] can provide such agility. In contrast to current backbone networks, all-photonic networks have the property that both transmission and switching are performed in the optical domain. The absence of optical-electrical-optical (OEO) conversion leads to two important advantages: greatly increased capacity and transparency to data formats and bit rates.

A. AAPN Networks

The AAPN has adopted an overlaid star topology, shown in Figure 1, which consists of a number of hybrid photonic / electronic edge nodes connected together via a wavelength stack of buffer-less transparent photonic switches placed at the core nodes (a set of space switches, one for each wavelength). Each edge node contains a separate buffer for the traffic destined to each of the other edge nodes. These buffers are called Virtual Output Queues (VOQs) [2] and are used to eliminate the Head-Of-Line blocking associated with First-In-First-Out (FIFO) queuing [3]. Traffic aggregation is performed in these VOQs, where packets are collected together in fixed-size slots that are then transmitted as single units across the network via optical links. At the destination edge node, the slots are partitioned, with reassembly as necessary, into the original packets.

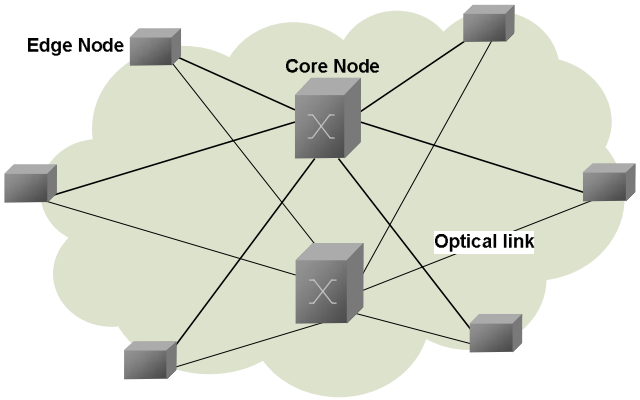

Figure 1 AAPN Overlaid Star Topology

The control of the AAPN is performed in the electronic domain. Each core switch has an associated electronic controller that performs timeslot allocation, switch configuration and other control functions. The control messages are exchanged between edge nodes and core nodes in the form of control slots out-of-band over a dedicated channel (on a particular wavelength) on each fiber. Each edge-node signals traffic demand information to the optical core along control channels every frame. The optical core uses this information to compute a schedule of timeslot allocations for all edge-nodes and signals back to inform each edge-node of the timeslots for each destination that it may use to transmit its traffic.

The AAPN technical criteria are defined as follows. The dimension of the photonic switches is no more than 64×64.

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada and industrial and government partners, through the Agile All-Photonic Networks (AAPN) Research Network.

The timeslot duration is specified as 10 microseconds with a configuration overhead of less than 1 microsecond. The link capacity is 10Gbps.

B. Different Timeslot Allocation Schemes in the Literature

The precise timeslot allocation scheme to be adopted in the agile all-photonic core is an important issue. The timeslot allocation scheme under the AAPN context should possess the property that the time complexity should be low enough to permit a practical implementation in the context of high-speed optical switching.

In the literature, many timeslot allocation schemes have been proposed such as Parallel Iterative Matching (PIM) [4], Iterative Round-Robin Matching (iRRM) [5], Iterative Round-Robin with SLIP (iSLIP) [5], Dual Round-Robin Matching (DRRM) [6], and Karp's randomized algorithm [7]. These algorithms were developed in the context of input-queued switches which consist of a switching fabric equipped with VOQs at its input ports. If the traffic is non-uniform and the occupation of some queues starts to increase, these algorithms cannot contribute more to the service of these queues and further cause the input queues to become unstable without internal switch speedup.

Non-uniform traffic is prevalent in current networks. In order to handle such traffic, the Birkhoff-von Neumann (BvN) decomposition approach to input-queued switch scheduling has been introduced by Chang et al [8]. Chang's algorithm needs an offline part to compute the switch configurations with time complexity of $O(N^{4.5})$ where N denotes the number of ports of the switch. Towles et al. [9] also introduced a BvN decomposition approach to input-queued switch scheduling. In [9], Towles introduced three different offline approaches, EXACT, MIN and DOUBLE. The latter two are heuristics. The time complexity of these three approaches is $O(N^{4.5})$, $O(N^{3.5})$ and $O(N^2 \log N)$ respectively. Paredes [10] uses an edge-coloring scheme to perform the BvN decomposition with time complexity $O(N\eta \log \eta)$ where η denotes the frame size. The particular implementation described by Paredes is restricted to η a power of two; however an algorithm with the same time complexity order due to Cole et al. [11] may be used for η not a power of two. Cole's algorithm is sophisticated and, although it achieves a time complexity that scales as $O(N\eta \log \eta)$, it has a large overhead. Keslassy et al [12] proposed a simple heuristic approach, called Greedy Low Jitter Decomposition (GLJD), to reduce overhead. However, GLJD can only find partial permutation matrices. In the worst case, the time complexity is $O(N^3)$ (which is implied by theorem 5 and figure 1 of [12]).

A simple and efficient resource allocation approach that can work well even under non-uniform traffic is needed to schedule high-speed agile all-photonic cores. In this paper, a simple and efficient BvN-based decomposition algorithm is proposed. To the best of our knowledge, the proposed algorithm has the lowest time complexity compared with other known BvN decomposition heuristics in the literature.

The paper is organized as follows. Section II lays the theoretical foundation of BvN decomposition. Section III introduces the quick BvN decomposition algorithm and analyzes its performance. Section IV presents and discusses the results of simulations and Section V draws conclusions.

II. Theoretical Foundation

A $N \times N$ traffic matrix $M = (m_{ij})$ represents the traffic demand between ingress and egress ports of a $N \times N$ switch. The traffic matrix is η-*admissible* if all its entries are non-negative and its row and column sums are no greater than η. An η-*server service matrix* $S = (s_{ij})$ is defined as a matrix with integer entries such that:

$$s_{ij} \geq 0 \text{ and } \sum_i s_{ij} = \eta, \quad \sum_j s_{ij} = \eta, \quad \eta \in Z^+.$$

The η-server service matrix represents the maximal traffic demands that an optical core can serve in a frame size of η. An η-service matrix can always be found that can serve an η-admissible traffic matrix.

A *permutation matrix* is a matrix with (0,1)-entries whose row sums and column sums are one. Similarly, a *partial permutation matrix* is defined as a matrix with (0,1) entries whose row sums and column sums are at most one. Permutation matrices represent the configuration of optical switches, i.e., the connectivity pattern between ingress and egress ports without contention for one timeslot.

Birkhoff-von Neumann decomposition, the foundation of the algorithm proposed in this paper relies on the *Birkhoff-von Neumann Theorem* [13] that is stated in a form restricted to integer arithmetic in Theorem 1.

Theorem 1 [7]: Any η-server service matrix $S = (S_{ij})$ can be written as a convex combination of permutation matrices P'.

$$S = \sum_{k=1}^{K} \alpha_k P_k^{'}, \text{ with } \sum_{k=1}^{K} \alpha_k = \eta, \quad \alpha_k, K \in Z^+ \qquad (1)$$

Theorem 1 implies that all η-server service matrices can be provided by a photonic core configuration schedule with a frame size of η.

III. Quick Birkhoff-von Neumann decomposition algorithm

In this section, a simple and efficient BvN-decomposition based heuristic, called *Quick BvN decomposition algorithm (QBvN)*, is discussed. The motivation of QBvN is that, given a η-server service matrix, finding partial permutation matrices instead of permutation matrices, the combination of which is close to the η-server service matrix, may be accepted for high-speed agile all-optical cores provided that a low time complexity can be achieved.

A. The Basic Idea of QBvN Algorithm

QBvN algorithm treats a $N \times N$ η-server service matrix S as a *bipartite graph* $G=(V_i, V_j, E)$, a mathematical structure consisting of two disjoint vertex sets, V_i and V_j, identified here with ingress and egress switch ports respectively, and one edge set E, where each edge joins one vertex from each vertex set. A bipartite graph G can be derived from a service matrix by applying (2).

$$\forall i, j, \begin{cases} \text{k edges on G between vertex i and j,} & \text{if } s_{ij} = k > 0 \\ \text{no edges on G between vertex i and j,} & \textit{otherwise} \end{cases} \quad (2)$$

A *bipartite graph matching* G_M is a sub-graph of G such that no more than one edge is incident on any vertex. A bipartite graph matching G_M is said *perfect* if there is an edge incident to every vertex. Similarly, a bipartite graph matching G_M can be derived from a partial permutation matrix P by applying (3).

$$\forall i, j, \begin{cases} \text{one edge on G}_M \text{ between vertex i and j,} & \text{if } p_{ij} = 1 \\ \text{no edges on G}_M \text{ between vertex i and j,} & \textit{otherwise} \end{cases} \quad (3)$$

The general idea of QBvN is to decompose a given bipartite graph G into bipartite graph matchings $\{ G_M \}$ in the following way: For a $N \times N$ agile photonic core, the algorithm visits all N ingress ports on G one-by-one to generate a G_M. For each ingress port i, i=0, 1, .., N-1, the following two steps, *matching* and *selection*, are applied.

i. *Matching* obtains all augmenting edges incident on port i. An edge $<i, j>$ on G is said to be an *augmenting edge* if both port i and j on G_M are not yet part of the matching.
ii. *Selection* chooses one of the augmenting edges with the smallest egress port number and moves it to G_M from G.

After all N ingress ports are visited, no augmenting edges can be added into G_M and thus G_M is a maximal bipartite graph matching. Therefore its associated partial permutation is constructed. This procedure is repeated η times to find the first η partial permutations. Note that QBvN changes the visiting sequence of the ingress ports in a round-robin fashion before generating the next G_M.

Another version of QBvN, called *QBvN-cover*, which offers guaranteed scheduling is proposed. The difference between the QBvN-cover and QBvN lies in the termination condition. QBvN terminates when η partial permutation matrices are generated. The QBvN-cover stops when the service matrix S can be covered by F partial permutation matrices $P(1), P(2), ...P(F), F \geq \eta$:

$$\forall 0 \leq i, j < N, \sum_{k=1}^{F} P_{ij}(k) \geq S_{ij} .$$

Note that the QBvN-cover may generate more than η permutation matrices in order to provide guaranteed scheduling. Consequently, an internal speed-up for switches running under the QBvN-cover is required.

B. Performance

Theorem 2 provides the bounds on the number of partial permutations found by QBvN-cover.

Theorem 2: Let F be the number of partial permutation matrices decomposed from a η-server service matrix by the QBvN-cover algorithm. Then

$$\eta \leq F \leq 1.5\eta$$

Proof: The lower bound derives directly from Theorem 1. The upper bound is given according to the following argument.

Let $\sigma(1), \sigma(2)...\sigma(N)$ be a permutation of the ingress ports that describes the order they are visited in the procedure of generating a partial permutation matrix. Note that QBvN-cover visits all N ingress ports to obtain a partial permutation matrix and hence N rounds of selection-matching procedures are called. At the kth matching-selection round, the ingress port $\sigma(k)$ is visited and the selected egress port is denoted by $\delta(k)$.

According to QBvN-cover's matching-selection procedure, it is possible that an ingress port $\sigma(k)$ is blocked by a set of ingress ports, $\Gamma_k = \{\sigma(i)\}, i < k$. By *blocking* it is meant that the ingress ports in Γ_k select all egress ports that $\sigma(k)$ can select for matching. As a result, if a blocking happens, $\sigma(k)$ cannot find any egress ports for matching at the kth matching-selection round.

For an ideal BvN decomposition algorithm, $|\Gamma_k| = 0$ for any $k \leq N$. It implies that there does not exist any blocking, which is true in terms of Theorem 1.

Now we use reduction to absurdity to prove that $|\Gamma_k|$ is impossible to be 1 for any $k \leq N$.

Assume that for a given k and w, $\sigma(w), w < k$, is the only element in Γ_k. At the wth matching-selection round, $\sigma(w)$ selects egress port $\delta(w)$. Investigating the $\sigma(k)$th row sum $R_{\sigma(k)}$ and the $\delta(w)$th column sum $C_{\delta(w)}$ in the η-server service matrix, we have $R_{\sigma(k)} = \eta$ and $C_{\delta(w)} \geq \eta + 1$, which contradicts with the definition of η-server service matrix.

Hence, the worst case occurs when $|\Gamma_k| = 2$, where an ingress port is blocked every two matching-selection rounds. It implies that after N rounds of selection-matching procedures, $N/3$ ingress ports are blocked and thus cannot find any egress ports for matching. Consequently, the cardinality of a matching, c, is

$$c \geq \frac{2N}{3}$$

With the condition

$$c \cdot F = N \cdot \eta$$

We have

$$F \le \frac{N\eta}{2N/3} = 1.5\eta \qquad \square$$

Corollary 1: Let Δ denote switching overhead in units of timeslots. For any admissible traffic, a speedup of $\frac{3}{2-3\Delta}$ is sufficient for QBvN-cover to schedule switch configurations that provide at least a η-server service.

Proof: Let $F_{\max} = 1.5\eta$. In order to schedule $F_{\max}$ partial permutation matrices in η timeslots, the required speedup γ of QBvN-cover can be calculated.

$$\gamma = \frac{F_{\max}}{\eta - F_{\max} \cdot \Delta} = \frac{3}{2-3\Delta} \qquad \square$$

C. Complexity of QBvN

The complexity of QBvN depends on the implementation of the matching and selection. In this paper, the complexity can be reached in the condition that N is no more than 64 under Intel's 64-bit processor. Note that the condition is compliant with the AAPN technical criteria.

The matching can be implemented as a label-matching procedure. Each ingress port i, i=0, 1, …, N-1, keeps a label with N bits, called L_i, and each bit indicates an egress port. If there exist edges from the ingress port i to egress port j on the bipartite graph G, the jth bit of L_i is set to 1; 0 otherwise. For each bipartite graph matching G_M, there is also a label with N bits, called L_{free}, and each bit indicates an egress port. If the egress port j is free on the G_M, the jth bit of L_{free} is set to 1; 0 otherwise. An egress port on G_M is said to be *free* if the egress port is not yet part of the matching. The L_{free} marks all free egress ports ready for current matching on G_M. The label-matching procedure may thus obtain all augmenting edges incident on port i by applying the following operation:

$$L_M = L_i \text{ AND } L_{free}$$

Here L_M is also a label with N-bits and each bit indicates an egress port. The edge $<i, j>$ is an augmenting edge incident on port i if the jth bit of L_M is 1. The notation "AND" denotes a bitwise "and" operation. Here we assume the operation AND takes O(1) time, which is true if N is smaller than the effective word length of a computer (in number of bits). Thus the time complexity of matching is O(1).

The selection can be formulated as finding the index of the least significant "1" bit (the lowest "1" bit) in L_M. The index denotes the egress port with the smallest number joining an augmenting edge.

The selection procedure is divided into two steps. The first step is to find the least significant "1" bit; the second step is to find the index of that "1" bit. We use the operation

$$L_{least} = L_M \text{ AND } (-L_M)$$

to obtain an N bit label L_{least} where only the position of the least significant "1" bit is marked by "1". Note that $-L_M$ is the 2's complement of L_M. Then we perform the following two operations

$$L_{least} = L_{least} \text{ -1}$$

popcount

to find the index of the least significant "1" bit. The assembly instruction *popcount* (IA-64) is used to count the number of "1" bits in L_{least}, which is exactly the same as the index of the least significant "1" bit after the operation $L_{least} = L_{least}$ -1. The time complexity of the selection procedure is O(1) under the above assumption about the word length of the computer.

The time complexity of QBvN scales as $O(N\eta)$ because it needs to repeat the matching-selection procedure $N\eta$ times to find η matchings. The maximum number of partial permutation matrices generated by QBvN-cover is 1.5η. The time complexity of QBvN-cover thus scales as $O(N\eta)$ as well.

Note that if the QBvN or QBvN-cover is running on a K-bit processor where $K < N$, the time complexity will scale as $O(N\eta\left\lceil \frac{N}{K} \right\rceil)$.

IV. Simulations and Discussion

In this section, the delay performance of QBvN and QBvN-cover is studied in a simulated AAPN environment. The traffic is generated with a composite Markov Poisson Process model [14] that simulates statistics of the following types of traffic: continuous bit rate (e.g. voice), variable bit rate (e.g. video) and variable bursty traffic (data). The destination of each slot is chosen by a uniform random process. The capacity of each VOQ is assumed to be infinite. The longest propagation delay between edge nodes and AAPN core nodes is assumed to be d. The AAPN core collects traffic information, i.e. the length of each VOQ, and organizes it in a traffic matrix. A service matrix is constructed in terms of the traffic matrix by running a simple scaling and rate filling routine [10]. Given the service matrix, the timeslot allocation scheme, e.g. QBvN or QBvN-cover, can be applied to generate partial permutation matrices. A schedule generator is used to complete the partial permutation matrices by connecting residual free ingress and egress ports. The generated timeslot schedules are then signaled to the edge nodes.

Two scenarios, Metropolitan Area Network (MAN) and Wide Area Network (WAN), are studied in this paper. MAN is defined as an AAPN network with 100km optical links (d=0.5ms). WAN is defined as a national wide AAPN network with 2000km optical links (d=10ms). The number of edge nodes considered is 16 (small AAPN switch) and 64 (large AAPN switch).

A. Delay Performance

For comparison, two algorithms, EXACT and GLJD, in the literature are selected. The EXACT provides an exact BvN decomposition from a service matrix with high computation complexity. The GLJD provides the least switch configurations with low computation complexity.

Figure 2 shows the comparison of delay performance among these algorithms. For MAN (d=0.5ms), the delay performance of QBvN-cover and that of EXACT are close. But for WAN (d=10ms), QBvN-cover shows better performance because the long propagation delay causes the traffic demand information collected in the AAPN core to be out-of-date. GLJD shows the worst delay performances although it shows a competitive delay performance in WAN with load more than 0.9.

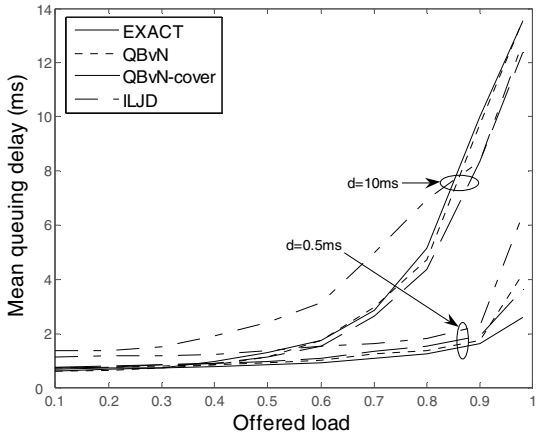

Figure 2 Comparison of Delay Performance among Different Algorithms (N=64, η =100)

Figure 3 shows the delay performance of QBvN-cover with different propagation delays (0.5ms and 10ms) as a function of offered load for switches with 16 and 64 ports. As we might expect, the performance degrades with the number of ports. However, the performance degrades differently under low and heavy load. For low offered load, i.e. load<0.5, the queuing delay degrades only slightly due to the existence of plenty of free timeslots for each source edge node. At high offered load, the delay performance degrades more sharply especially in the case of a WAN. The reason lies once again in the long propagation delay. We noticed that the mean queuing delay of slots is much less than the signaling time, i.e., the round-trip propagation delay, when the load is not very heavy. This is because the residual free bandwidth is allocated by QBvN-cover schedulers, which improves delay performance of slots, especially when the propagation delay is large, e.g., WAN [15].

B. Analysis of design tradeoffs

Naturally, it is not mandatory to input a service matrix to QBvN-cover. It might be valuable, in terms of reduced computation, to generate partial permutation matrices directly from traffic matrix instead of a service matrix. However, Figure 4 shows that a service matrix construction procedure efficiently reduces mean queuing delay. The reason for the reduction lies in the bursty characteristics of traffic arrivals, that is, a huge amount of traffic demand may flood certain edge nodes sometimes while nearly no traffic goes to others. In the case where the service matrix construction process is not adopted, nearly all bandwidth is allocated by QBvN-cover to fulfill such huge traffic demands so that the small traffic demands cannot be served promptly. Consequently, those small traffic demands may encounter heavy delay and cause poor delay performance.

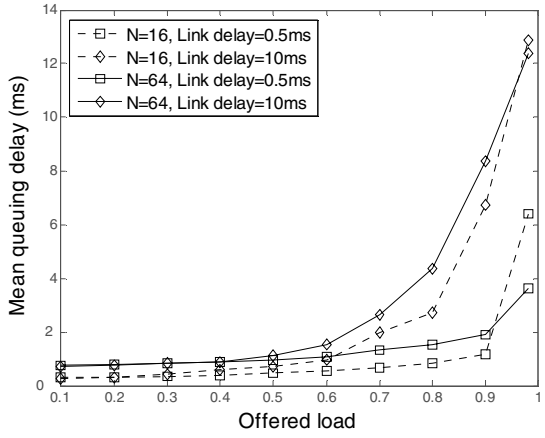

Figure 3 Mean Queuing Delay of QBvN-cover as a Function of Offered Load

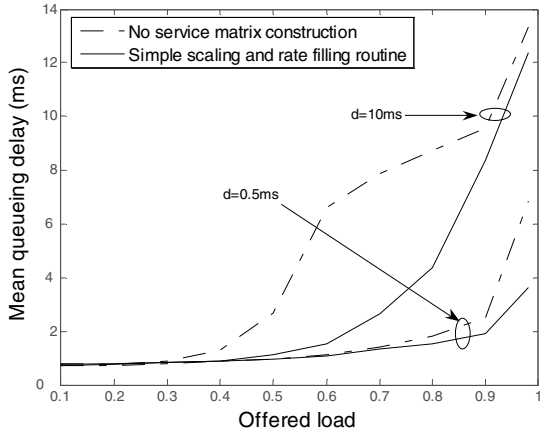

Figure 4 Comparison of QBvN-cover's Delay Performance with/without Service Matrix Construction (N=64)

V. CONCLUSION

In this paper, the problem of finding a simple and efficient approach to approximate BvN decomposition for AAPN cores is considered and the QBvN algorithm, whose time complexity can reach $O(N\eta)$ for a frame size of η, is proposed as a good solution. QBvN-cover, as another version of QBvN, is proposed. For QBvN-cover, the bound on the number of generated switch configurations is provided and hence the

necessary speedup for AAPN cores so that QBvN-cover can provide guaranteed scheduling with configuration overhead.

Under stream-type, continuous bit rate, traffic, QBvN-cover shows better delay performance than other algorithms in the literature, especially in a WAN environment. Although it is not mandatory for QBvN-cover to use a service matrix as its input, the service matrix construction procedure is necessary for QBvN-cover to deliver good delay performance.

ACKNOWLEDGMENT

Dr. Trevor J. Hall holds a Canada Research Chair in Photonic Network Technology at the University of Ottawa and is grateful to the Canada Research Chairs Program for their support. The authors are also indebted to Dr. Sofia Paredes for her insightful comments and careful reading of the manuscript.

REFERENCES

[1] G.v. Bochmann, M.J. Coates, T. Hall, L. Mason, R. Vickers and O. Yang, “The Agile All-Photonic Network: An architectural outline”, Proc. Queen's University Biennial Symposium on Communications, 2004, pp.217-218

[2] Y. Tamir and G. Frazier, “High performance multiqueue buffers for VLSI communication switches”, In Proceedings of 15th International Symposium on Computer Architecture (ISCA), May/June 1988, pp. 343--354

[3] N. McKeown, P. Varaiya and J. Warland, “Scheduling cells in an input-queued switch”, IEEE Electron. Letter, December 1993, pp.2174-2175

[4] T. Anderson, S. Owicki, J. Saxe and C. Thacker, “High Speed Switch Scheduling for Local Area Networks”, ACM Transactions on Computer Systems 11(4), November 1993, pp. 319 – 352

[5] N. McKeown, "iSLIP: A scheduling algorithm for input-queued switches", IEEE/ACM Transactions on Networking, 7(2), April 1999, pp. 188-201

[6] H. J. Chao, “Satrun: a terabit packet switch using dual round-robin”, IEEE Communication Magazine, vol. 38, no. 12, December 2000, pp. 78-79

[7] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, “An optimal algorithm for on-line bipartite matching”, Proceedings of the twenty-second annual ACM symposium on Theory of computing, April 1990, pp. 352-358

[8] C.S. Chang, W.J. Chen and H.Y. Huang, “Birkhoff-von neumann input buffered crossbar switches”, Proceedings of IEEE INFOCOM, 2000, pp. 1614-1623

[9] B. Towles and W. J. Dally, "Guaranteed scheduling for switches with configuration overhead," IEEE/ACM Transactions on Networking, vol. 11, no. 5, October, 2003, pp. 835-847

[10] S.A. Paredes and T.J. Hall, “Flexible bandwidth provision and scheduling in a packet switch with an optical core”. OSA Journal of Optical Networking, 4(5), May 2005, pp. 260-270.

[11] R. Cole, K. Ost and S. Schirra, “Edge Coloring Bipartite Multigraphs in O(E logD) Time”, Combinatorica, 21(1), 2001, pp. 5-12

[12] I. Keslassy, M. Kodialam, T.V. Lakshman and D. Stiliadis, "On guaranteed smooth scheduling for input-queued switches", Proceedings of IEEE INFOCOM, 2003, pp. 1384-1394

[13] G. Birkhoff, “Tres observaciones sobre el algebra lineal”, Univ. Nac. Tucuman, Rev. Ser. A 5, 1946, pp. 147-151

[14] M. Schwartz, “Broadband Integrated Networks”, New Jersey: Prentice Hall,1996, pp. 21-32

[15] C. Peng, P. He, G. v. Bochmann and T. J. Hall, “Delay Performance Analyses for an Agile All-Photonic Star Network”, accepted by the 5th IFIP-TC6 Networking, Coimbra, Portugal, May 2006