

Group-Scheduling for Optical Burst Switched (OBS) Networks

Saravut Charcranon, Tarek S. El-Bawab, Hakki C. Cankaya, and Jong-Dug Shin¹
Network Strategy Group, Alcatel USA, Plano, Texas

Abstract— To date, all of the proposals for burst-scheduling techniques have considered scheduling individual bursts. We introduce a new scheme called OBS group scheduling. In this approach, a burst is represented by an interval of time. The process of scheduling a number of bursts, thus, turns to be a process of fitting a set of the corresponding time intervals on a channel time line that represents a channel-time resource. By doing so, we can formulate the scheduling process as a combinatorial optimization problem. Then, graph theory is applied to schedule as many non-overlapping intervals as possible onto the channel time line. The underlying concept of the group scheduling is that of briefly delaying the scheduling of a burst so that a much better decision can be made about a number of bursts all-together. This scheme is shown, through simulations, to improve performance in terms of burst loss probability and channel utilization over existing schemes.

Keywords- Burst scheduling; optical burst switching; interval scheduling; combinatorial optimization.

I. INTRODUCTION

Optical Burst Switching (OBS) has been receiving an increasing attention as a potentially bandwidth-efficient approach for future optical core networks. In OBS networks, an ingress node assembles incoming packets into data bursts (DBs). Along with each of these created DBs, the ingress node also generates a Burst Header Packet (BHP) that contains control information like channel identification, destination-node identification, and the DB length and DB arrival time. The DBs will later be disassembled into the original packets at an egress node. Unlike packet switched networks, OBS networks separate each DB from its control information (i.e. BHP) and transmit them on different channels. A channel carrying DBs is referred to as a data channel; while, a channel carrying BHPs is referred to as a control channel. Upon receiving the BHP, intermediate nodes allocate resources based on the information it carries. While a BHP is sent out to reserve network resources along a path, transmission of the DB is deferred for some time period, which is known as the “offset time”.

Several scheduling techniques have been proposed for OBS networks. Earlier proposals were mainly based on the concept of *Horizon scheduling* [1], where the scheduler attempts to reserve channels for a data burst immediately after the arrival of a header cell. If the attempt is successful, resources will be

reserved right away until a burst transmission is due to complete. The only information need to be kept is the latest time the channel was utilized. This scheme, however, can not utilize a transmission channel during the time gaps between scheduled transmissions. Later, the concept of *delayed reservation* was introduced where an *offset time* is maintained between DB and BHP transmissions. Protocols such as *Just-Enough-Time (JET)* and *Just-In-Time (JIT)* were based on this concept [2, 3] and aimed at allocating transmission resources only upon arrival of the DB and for the time period specified in its BHP. This made it possible to fit a future DB with the right size into a vacant transmission gap between the already scheduled DBs. Delayed reservation based with an offset time scheduling is more resource efficient than *horizon-based* scheduling and results in lower burst dropping probability. Yet, it reserves resource immediately after the BHP is received and processed. Since the scheduler processes one request at a time, delayed reservation resembles an on-line processing mechanism, in which no future information is available to support the process of decision making. In this paper, we introduce a new OBS scheduling technique called *OBS group-scheduling (OBS-GS)* that relies on BHP grouping to gain additional information and prior details about a group of incoming DBs before commencing to schedule them. Group scheduling is a scheme in which the task of serving the scheduling needs of arriving requests is not carried out immediately on an one-by-one basis. Instead, scheduling is delayed for certain period of time during which BHP arrivals are gathered. Then, the scheduler processes all these requests at once with the goal of optimizing resource utilization. The proposed group scheduling technique could be regarded as an approach, among others, that attempts to push scheduling efficiency to some limit with the goal of identifying the extents of OBS benefits in a network environment.

The paper is organized as follows. Section II discusses the group scheduling system model where fundamental building blocks are enumerated. Section III explains the operation of group scheduling based on a concept of interval graph. Details of algorithms are provided in section IV, and the performance evaluation of the purposed scheme is presented in section V. Finally, the conclusion is given in section VI.

¹ Currently with the School of Electronic Engineering, Soongsil University, Seoul, Korea (jdshin@ssu.ac.kr). This work was carried out at Alcatel where he was on a sabbatical leave.

II. GENERIC GROUP SCHEDULING SYSTEM MODEL

In our model, a channel is represented by a time line. Group scheduling scheme partitions this channel time line into small time windows where a channel scheduling decision can take place on a per-window basis. Only those DBs that intend to utilize channel during a certain channel window, not the others, are scheduled together. A BHP, in this scheme, thus serves as a DB agent in requesting channel resources for a particular channel window. To accomplish such a goal, a group-scheduling system requires three basic building blocks: a BHP grouper module, a classifier and channel assignment module, and a channel scheduler, as illustrated in Fig. 1.

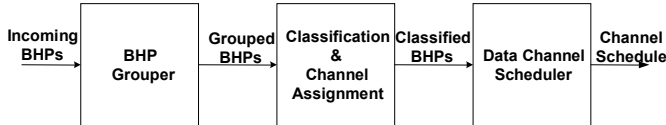


Figure 1 Generic Group Scheduling System Model

A. BHP Grouper Module

In order to be eligible for scheduling in a certain time window, a BHP must arrive prior to the closing time of this window. The BHP grouper scrutinizes the BHPs, by looking at their DBs arrival times and durations, to determine the channel window the DBs are supposed to be transmitted within. The BHP grouper then puts the BHP in an appropriate basket as shown in Fig. 2, each basket represents each channel time window. (i.e., it is a one-to-one mapping between a channel time window and a basket.) For each basket, BHPs may be sorted, e.g., by their data burst arrival times. This undertaking would be of help later in BHP classification and channel assignment for multiple-channel scheduling systems. Once the closing time of each channel window is reached, the corresponding basket will be removed from the BHP grouper, and no other BHPs will be eligible for scheduling consideration in that window. Fig. 2 illustrates the BHP grouper operation. Basket 1 collects all BHPs that want to use a channel during $[W_{Start}, W_{Stop}]$ and will be removed at time $T_{Close}(1)$, Basket 2 collects the BHPs that require to use channel during $[W_{Start} + One\ window\ period, W_{Stop} + One\ window\ period]$ and will be removed at $T_{Close}(2)$, and so on.

B. Classification and Channel Assignment Module

A schematic diagram of the classification and channel assignment module is depicted in Fig. 3. It performs channel (or wavelength) allocation and/or service differentiation. Service differentiation can be provided, for examples, by employing a priority scheme, a weighted round robin discipline, a weighted fair queue, or a pre-assigned proportion of BHPs from each class. After being grouped, BHPs are classified into classes and placed in the appropriate queues. In case of a single class, there is only a single queue. The channel (wavelength) assignment module then hands over BHPs to the appropriate channel schedulers. This is the place where channel management leads to efficient resource utilization and service differentiation. In the single channel case there is only one

channel scheduler and the channel assignment task is reduced to allocation of bursts, of various classes, on this channel

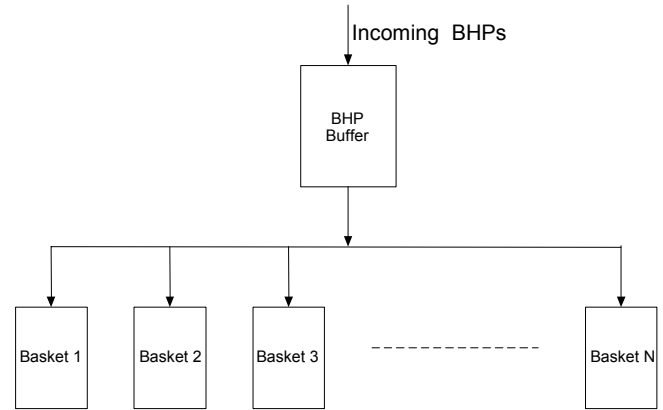


Figure 2 BHP Grouper Model

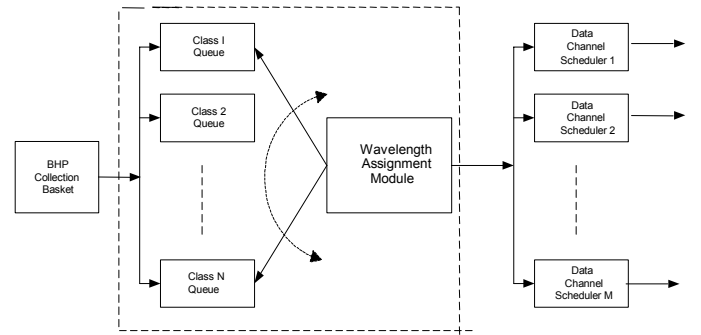


Figure 3 Classification and Channel Assignment Model

C. Channel Scheduler Module

The channel scheduler module schedules BHPs (based on a specification of the start and end times of their corresponding DBs). The primary objective is to maximize the number of BHPs (and, accordingly, DBs) to be scheduled in the channel time window. For a set of BHPs/DBs, the scheduler first establishes *interval representation profile*, as shown by the example in Fig. 4. The figure shows seven DBs with various start times and durations. Once the interval representation profile is created, it is transformed to an *interval graph*. Fig. 5 depicts the interval graph for the example of Fig. 4. Each vertex in the graph represents a burst. There exists an edge to connect two vertices if, and only if, their corresponding bursts are overlapped, otherwise there is no edge between these vertices. For example, DBs A and B are overlapped, there is thus an edge connecting vertices A and B in Fig. 5. On the other hand, there is no edge between vertices D and E since these DBs do not overlap. Based on the interval graph, the channel scheduler applies an appropriate algorithm to obtain the set of maximum number of non-overlapping intervals (under whatever desired constraints). For example, there are two possible sets of maximum non-overlapping DBs. The first set is $\{C, E, D\}$, and the other set is $\{C, E, A\}$. The channel scheduler can choose either group depending on the criteria deployed in the algorithm.

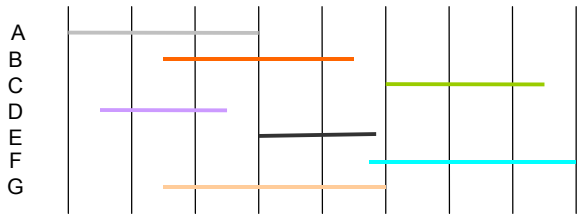


Figure 4 Interval Representation

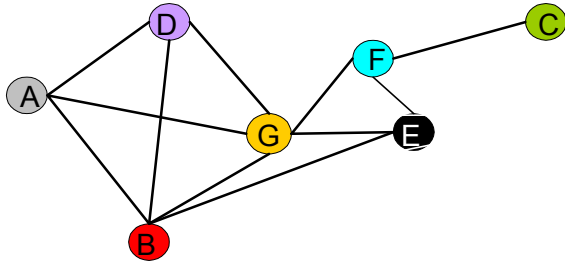


Figure 5 Interval Graph

III. GROUP SCHEDULING SYSTEM OPERATION

As mentioned earlier, the channel time (as the channel resource) is partitioned into a sequence of equal time windows. During each window, a core node (the BHP grouper therein) keeps collecting BHPs arriving over the control channel. To be eligible for being scheduled in this time window, the BHP must arrive before its closing time, T_{Close} (as indicated by a top horizontal broken line in Fig. 6). Typically, the window closing time is well before of the actual channel time during which DBs would be transmitted (taking into consideration other processing times such as scheduling time).

Fig. 6 illustrates the timing relationship between the BHPs and their associated DBs. The bold short dashes/lines on the left vertical axis represent incoming BHPs while the long horizontal lines represent their corresponding DBs. (Note that these DBs are created from information carried in the BHPs, but the actual DBs have not yet arrived.) The latter are shown in relation to the channel time window. The BHPs appearing above the collecting close time are not eligible for scheduling in this window. Once the BHP collection process is over, the system may make the scheduling decision(s). Fig. 7 depicts the scheduling process of two consecutive channel time windows. Fig. 8 depicts the sequence of processes as seen by a core node for a single-channel scheduling window. The diagram is adapted from the Gantt timing diagram. First, BHPs are grouped. Then, classification and channel assignment takes place. Once the data channel and a class of service are identified, BHPs are scheduled and a set of new BHPs are forwarded to the downstream. Finally, when DBs arrive, they are forwarded according to the schedule plan. Note that scheduling must finish by some amount of time before the DBs forwarding process actually starts.

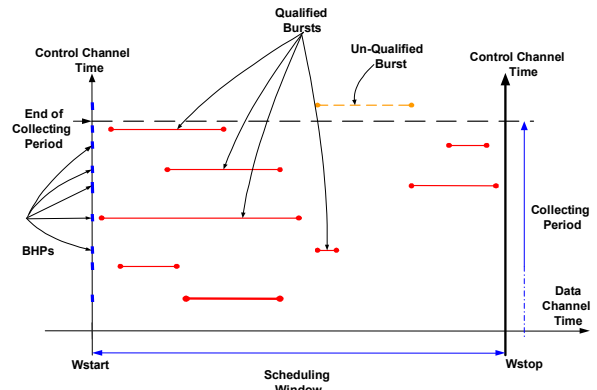


Figure 6 BHP Collecting Window in Relation to Channel Scheduling Window

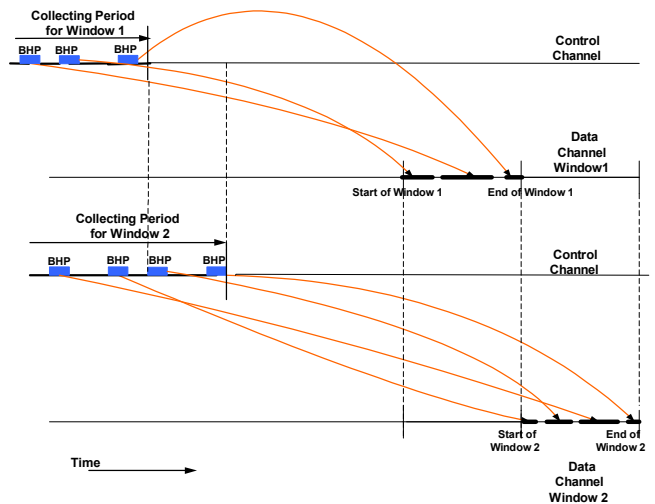


Figure 7 BHPs and Channel Window Mapping Diagram

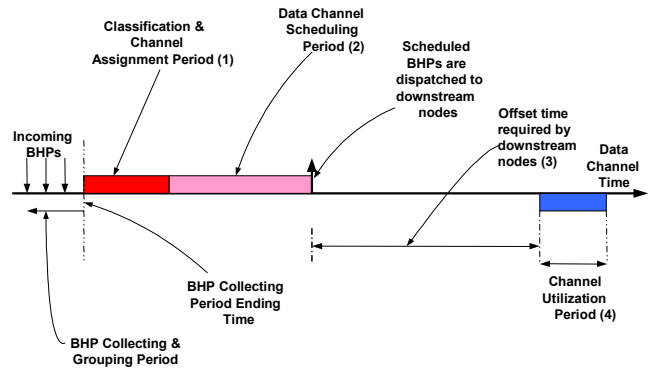


Figure 8 Sequence of events during a single epoch of data channel scheduling window

IV. ALGORITHMS

A. Maximum Stable-Set Interval Scheduling Algorithm

The interval-scheduling algorithm is the central algorithm in this work. It is an algorithm to be carried out by the channel scheduler to schedule DBs onto a channel. The basic task of this algorithm is to create an interval graph out of the set of DBs delivered by the classification and channel assignment module. The channel scheduler then uses this graph to obtain the maximum number of bursts that can be scheduled. To achieve the goal, the scheduler leverages the unique properties of an interval graph in order to find its *maximum stable set*. The *maximum stable set of the graph* is the set that contains a maximum number of non-adjacent vertices in the graph. Generally, the problem of finding such a set in an arbitrary graph is known to be an NP-Complete problem [4, 5, 6, 7]. However, in the case of a well-structured graph like an interval graph, there exists an algorithm to obtain such a set [8]. Interval graph is also known to be a perfect graph [8, 9] that possesses two unique properties. First, it is a triangulated graph. Second, it is a comparability graph. Being a triangulated graph means that the perfect graph gives a systematic scheme to obtain a set of stable vertices of maximum members. This scheme is the *perfect vertex elimination scheme* [8], as will be discussed below. The *maximum stable-set interval scheduling algorithm* is shown in Fig. 9. For a graph $G = (V, E)$, where V is a set of vertices and E is a set of edges, we define an adjacency set of a vertex $X \in V$ as a set of vertices Y where the edge $(X, Y) \in E$.

B. Lexicographic Breadth First Search (LexBFS)

From Fig. 9, we use the LexBFS (step 3) to obtain a perfect-vertex elimination scheme from a given adjacency sets of an interval graph. To understand LexBFS, we first discuss general Breadth First Search (BFS). In BFS [8, 10], we start from an arbitrary vertex in the graph and put it in the queue of vertices to be visited. Repeatedly, we remove the vertex X at the head of the queue, and add to the queue all the adjacent vertices of X that are not yet in the queue. BFS stops when no vertex is left in the queue. In lexicographic BFS, vertices are numbered from n to 1. During the search, each vertex is labeled by a sequence of numbers with a decreasing order. The vertex with the greatest lexicographic label will be removed from the queue and declared visited. A label $L_1 = [p_1, p_2, \dots, p_k]$ is said to be lexicographically greater than a label $L_2 = [q_1, q_2, \dots, q_l]$ if either for $i < j$, $p_i = q_i$ and $p_j > q_j$, or $p_i = q_i$ for $i \leq l$ and $k > l$. Two labels are said to be lexicographically equal if $k = l$ and $p_i = q_i$ for $i \leq l$. For example lexicographically, $9761 < 985$ and $643 < 6432$.

C. Perfect Vertex Elimination Order

In order to identify the maximum stable set, a perfect vertex elimination order is required. The following is a procedure to establish a perfect vertex elimination order [8]. We apply the LexBFS to an interval graph, represented in a form of adjacency sets. Let $G = (V, E)$ be an undirected graph and let $\sigma = [v_1, v_2, \dots, v_n]$ be an ordering of the vertices. σ is said to be a *perfect vertex elimination order* if each v_i is a simplicial vertex

(defined below) of the induced sub-graph $G_{\{v_i, \dots, v_n\}}$. That is, each set $Y_i = \{v_j \in \text{Adj}(v_i) \mid j > i\}$ is complete. Note that a vertex X of a graph G is called *simplicial* if its adjacent vertices, $\text{Adj}(X)$, induce a complete sub-graph of G . A graph is complete if all vertices are mutually adjacent. Note also that from [8], an undirected graph $G = (V, E)$ is a triangulated graph if, and only if, the ordering σ produced by the LexBFS is a perfect vertex elimination order, and our interval graph is the triangulated graph.

D. Maximum Stable-Set (Max-SS) Algorithm

Given the adjacency sets of an interval graph G and the induced perfect vertex elimination order σ , the Max-SS algorithm identifies a collection of stable sets. From the collection of stable sets, we can choose the one with the maximum members for scheduling. The Max-SS algorithm is not shown here due to space limitation.

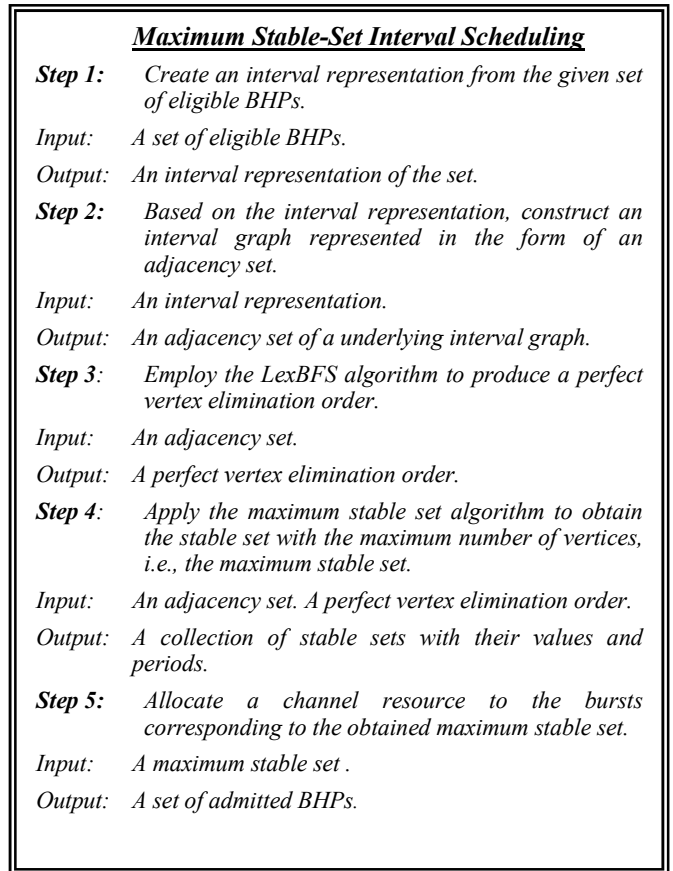


Figure 9 Maximum Stable-Set Interval Scheduling Algorithm

V. NUMERICAL RESULTS

We carried out simulation of this scheduling scheme assuming a single data channel per output and assuming no buffers. An immediate scheduling scheme (*JET-like* scheduling with a single channel [2]), the counterpart, is used as a baseline scheme representing one of the best currently known schemes. A channel bit-rate is 10 Gb/s and a mean burst size is 20 kbytes. We fix scheduling time window to 200 μ s. DBs are

generated by an *on-off* source model. Two performance metrics are studied: burst loss probability and channel utilization. 600 scheduling windows (approximately 6000 bursts) were simulated at each offered load.

A. Burst Loss Probability

Fig. 10 compares the burst-loss probability of the group scheduling with that of the immediate scheduling. Both probability distributions of the source “on” and “off” states are exponentially distributed. The results show that the group scheduling outperforms the immediate scheduling over the entire range of the mean offered load. The gain increases slightly with load. Up to 5% improvement in burst loss probability can be achieved, particularly in the full load to overload region. A similar trend is also observed when the lengths of DBs are deterministic (not shown here). Again, the improvement by group scheduling is up to about 5%.

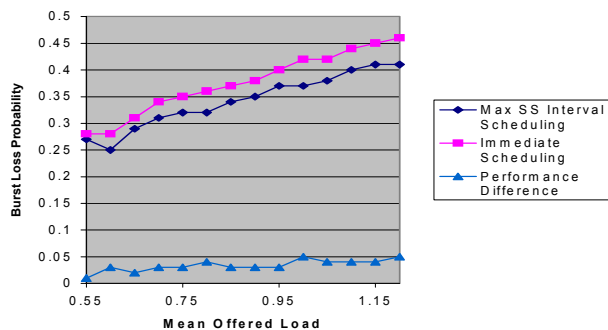


Figure 10 Burst loss probability for exponential inter-arrival time and burst length

B. Channel Utilization

Fig. 11 shows a comparison between the two schemes in regards to channel utilization. Both the “on” and “off” states are exponentially distributed. OBS group scheduling performs better also in terms of channel utilization. The improvement is close to 5% over the entire range of mean offered load from 0.55 to 1.2. Again similar trend is observed when burst size is fixed with exponentially distributed inter-arrival time (not shown here). An advantage of our proposed OBS group scheduling over the immediate scheduling is shown, specifically at high loads, i.e., from 0.95 to 1.2.

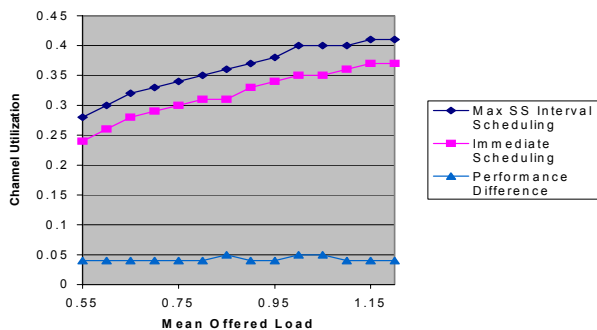


Figure 11 Channel utilization: exponential inter-arrival time and burst length

VI. CONCLUSION

Group scheduling is a novel burst scheduling approach for OBS networks. In this approach, DBs are scheduled in groups. The problem of DB scheduling is mapped to a combinatorial optimization problem of scheduling DB time intervals on a channel time line. The maximum stable set algorithm is proposed to obtain the maximum number of non-overlapping bursts. Performance comparison of group scheduling with existing scheduling schemes was performed. The results showed the advantages of group scheduling in terms of burst loss probability and channel utilization. Up to 5% improvement is possible in both categories. We argue that this improvement is quite a hard-pressed improvement, and we anticipate difficulty in pushing it further unless, for example, the characteristics of the DBs can be altered.

REFERENCES

- [1] J. Turner, “Terabit burst switching,” *Journal of High Speed Networks*, Vol. 8, No. 1, pp. 3-16, 1999.
- [2] C. Qiao and M. Yoo, “Optical Burst Switching (OBS)—a new paradigm for an optical internet,” *Journal of High Speed Networks*, Vol. 8, No. 1, pp. 69-84, 1999.
- [3] J.Y. Wei and R.I. McFarland, “Just-In-Time signaling for WDM optical burst switching networks,” *IEEE Journal of Lightwave Technology*, Vol. 18, No. 12, pp. 2019-2037, December 2000.
- [4] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York, 1979.
- [5] F. Gavril, “Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph,” *SIAM Journal of Computing*, Vol 1, No. 2, pp180-187, 1972.
- [6] R. Karp, “Reducibility among combinatorial problems,” In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, Plenum Press, 1972.
- [7] R. Kopf, G. Ruhe, “A Computational study of the weighted independent set problem for general graphs,” *Foundations of Control Engineering*, Vol. 12, No. 4, 1987.
- [8] Martin C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.
- [9] J.R. Ramirez Alfonsin, and Bruce A. Reed, *Perfect Graph*, Wiley-Interscience Series, John Wiley & Sons, Chichester, England, 2001.
- [10] D.J. Rose, R.E. Tarjan, and G.S. Lueker, “Algorithmic aspects of vertex elimination on graphs,” *SIAM Journal of Computing*, (5) pp 266-283, 1976.