Optimal Scheduling in Buffered WDM Interconnects with Limited Range Wavelength Conversion Capability

Zhenghao Zhang, *Student Member*, *IEEE*, and Yuanyuan Yang, *Senior Member*, *IEEE*

Abstract—All optical networking is a promising candidate for supporting high-speed communications because of the huge bandwidth of optics. In this paper, we study optimal scheduling in buffered WDM interconnects with limited range wavelength conversion capability. We formalize the problem of maximizing network throughput and minimizing total delay as a problem of finding an optimal matching in a weighted bipartite graph. We then give a simple algorithm, called the Scan and Swap Algorithm, that finds the optimal matching in $O(kB)$ time, where k is the number of wavelengths per fiber and B is the buffer length, as compared to directly adopting other existing algorithms that need at least $O(k^2N^2 + k^2BN)$ time, where N is the number of input fibers.

Index Terms—Wavelength-division-multiplexing (WDM), packet scheduling, wavelength conversion, limited range wavelength conversion, optical packet switching, optical interconnects, optical switching networks, optical buffering, packet loss probability.

1 Introduction and Background

All-optical networking with *wavelength-division-multiplexing (WDM)* has been proposed as a promising candidate for supporting high-speed communications [8], [4], [7] because of the huge bandwidth of optics. In WDM, the bandwidth of a fiber is divided into a number of independent channels, with each channel on a different wavelength. Several different technologies have been developed for transmitting data over WDM [8], such as broadcast-and-select, wavelength routing, optical packet switching (OPS), and optical burst switching (OBS). Broadcast-and-select networks and wavelength routing networks have been extensively studied. Optical packet switching and burst switching, especially optical packet switching, though still in the research phase, are attracting more and more interests because of their better flexibilities in utilizing the bandwidth [8]. In this paper, we focus on WDM packet switched networks.

In a WDM optical packet switched network, a data packet is modulated on a wavelength and may visit several intermediate nodes before reaching the destination. In each intermediate node, an interconnect (or a switch) is used to direct the incoming packets to the correct output fiber links. In an interconnect, there can be many input and output fiber links and, on each link, there can be multiple wavelength channels. Output contention arises when some packets on the same wavelength are destined for one output fiber at the same time. In general, there are three ways to combat output contention: deflection routing, buffering, and exploiting the wavelength domain [8]. Deflection routing is to send the contending packet to some other output link (which may or may not have a route to the destination). Buffering is to send the contending packet to some fiber delay lines in which the packet is delayed for a certain amount of time before being transmitted. Exploiting the wavelength domain is to convert the wavelength of a packet to some idle wavelength (if there is any) on the output fiber. In the first method, deflection routing, although the packet is not dropped, the end-to-end delay may be long and the packets arriving at the destination may be out of order. Therefore, in this paper, we study the combination of the second and the third method: buffering and exploiting the wavelength domain.

To translate a signal on one wavelength to another, *wavelength converters* are needed. If a wavelength converter is capable of converting a wavelength to any other wavelength in the optical system, it is called a *full range wavelength converter*. However, a full range wavelength converter is quite difficult and expensive to implement [9], [6]. Thus, to save the cost, it has been suggested that a *shared converter pool* can be used. A shared converter pool consists of a limited number of full range wavelength converters that can be accessed by all inputs. In this way, due to statistical multiplexing, fewer converters are needed in an interconnect. However, since it is difficult to convert a wavelength to another wavelength that is far apart from it, it may be suitable only for some applications and, in cases where the number of wavelengths is large and the wavelengths are far apart from each other, a more realistic wavelength converter is the *limited range wavelength converter*, which is only able to convert a given wavelength to a limited number of wavelengths. It was shown through analytical models and simulations that, in wavelength-routing networks as well as in OPS networks, with limited range wavelength converters, the network performance is close to those with full range wavelength converters even when the conversion degree is very small [9], [6], [10], [7].

• *The authors are with the Department of Electrical and Computer Engineering, State University of New York, Stony Brook, NY 11794-2350. E-mail: {zhhzhang, yang}@ece.sunysb.edu.*

Manuscript received 16 Nov. 2004; revised 22 Mar. 2005; accepted 19 Aug. 2005; published online 22 Nov. 2005.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0365-1104.

Thus, limited range converters can be considered as a practical, cost-effective choice for providing wavelength conversion ability and will be the main focus of this paper. It should be mentioned that full range wavelength converters can be considered as a special case of limited range wavelength converters.

Similarly to [8], [4], [7], we assume that the durations of optical packets are all one time slot long and the packets arrive at the interconnect at the beginning of time slots. In such an interconnect, a scheduling algorithm is needed to smartly assign the wavelength channels to the incoming packets to achieve high throughput and low packet delay. Note that if the wavelength conversion is full range, the scheduling is trivial because, in this case, the wavelengths of the packets do not affect the scheduling as full range wavelength converters are capable of converting any incoming wavelength to any outgoing wavelength. When the wavelength conversion is limited range, the scheduling becomes more complicated. The problem of maximizing network throughput in an unbuffered WDM interconnect with limited range wavelength converters was studied in [12]. In this paper, we study the more complex and difficult case when the interconnect is buffered. We show that the problem of maximizing network throughput and minimizing total delay in such an interconnect can be formalized as the problem of finding an optimal matching in a weighted bipartite graph called *request graph* and give an efficient algorithm called the *Scan and Swap Algorithm* that solves this problem in $O(kB)$ time, where k is the number of wavelengths per fiber and B is the buffer length.

Extensive research has been conducted on scheduling algorithms for electronic interconnects in the literature, for example, the well-known iSLIP algorithm [15] for input-buffered electronic interconnects in which the scheduling was carried out by augmenting a matching in parallel. However, this method cannot be applied to our problem since it is for input-buffered interconnects, while the interconnect considered in this paper is output-buffered. Moreover, in the output buffer, not all buffer locations can be assigned to a packet due to the constraint of limited wavelength conversion. In the past, buffered WDM interconnects with full range wavelength converters were studied and their performances were evaluated with analytical models by [4], [13]. Shen et al. [7] first considered buffered WDM interconnects with limited range wavelength converters and suggested to "store a packet in the output buffer that has the smallest number of packets" when contention arises. However, no proof was given as to whether this will either achieve maximum throughput or minimum total delay. In this paper, we consider the same interconnect as in [7], but will prove that our algorithm gives an optimal schedule that both maximizes network throughput and minimizes total delay.

Finally, it should be mentioned that the work in this paper is by no means an extension of the earlier work for unbuffered WDM interconnects [12] because, first, the objectives to be optimized are different, e.g., packet delay is a main concern in a buffered interconnect while there is no delay in an unbuffered interconnect. As a result, the optimal method developed for the latter cannot be used for the former. In particular, we consider optimal matchings in weighted bipartite graphs, while the earlier work [12] considered maximum matchings in unweighted bipartite graphs. An optimal matching must be a maximum matching, but the reverse may not be true. Finding an optimal matching is considerably harder than finding a maximum matching. In fact, many problems on weighted structures have completely different and, often, more complex algorithms than their unweighted versions.

The rest of this paper is organized as follows: Section 2 gives some basic terminologies on limited range wavelength conversion and the interconnect model. Section 3 gives the formalization of the scheduling problem. Section 4 briefly reviews the properties of request graphs. Section 5 presents the Scan and Swap Algorithm for finding optimal matchings in request graphs. Section 6 presents the simulation results. Finally, Section 7 concludes the paper.

2 Preliminaries

2.1 Wavelength Conversion

All-optical wavelength conversion is usually achieved by conveying information from the input light signal to a probe signal [18]. The probe signal is generated by a tunable laser tuned to the desired output wavelength. The tuning range of the laser is continuous, but, under limited range wavelength conversion, it is only part of the whole spectrum because of constraints such as tuning speed, loss, etc.

We can see that a wavelength can be converted to an interval of wavelengths because the tuning range of the laser covers an interval of wavelengths. Also, note that if the laser for the conversion of λ_1 can be tuned to λ_3, then the laser for the conversion of λ_2 should also be able to be tuned to λ_3 since λ_2 is closer to λ_3 than λ_1 is. Thus, we have the following two observations on wavelength conversion:

Observation 1. Wavelengths that can be converted from λ_i for $i \in [1,k]$ can be represented by interval $[Begin(i), End(i)]$, where $Begin(i)$ and $End(i)$ are positive integers in $[1,k]$ and $Begin(i) \leq i \leq End(i)$. Wavelengths that belong to this interval are the *adjacency set* of λ_i.

Observation 2. For two wavelengths λ_i and λ_j, if $i < j$, $Begin(i) \leq Begin(j)$ and $End(i) \leq End(j)$.

We call this type of wavelength conversion "ordered interval" because the adjacency set of a wavelength can be represented by an interval of integers and the intervals for different wavelengths are "ordered." This type of wavelength conversion was also used in other research works, for example, [16], [17]. The cardinality of the adjacency set is called the *conversion degree* of the wavelength. The *conversion distance* of a wavelength is defined as the largest difference between a wavelength and a wavelength that can be converted from it.

Wavelength conversion can be visualized by a bipartite graph in which left side vertices represent input wavelengths and right side vertices represent output wavelengths and input wavelength λ_i is connected to output wavelength λ_j if λ_i can be converted to λ_j. For example, Fig. 1 shows such a graph for $k = 6$. In the example, the

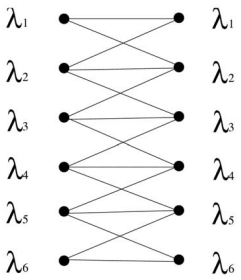


Fig. 1. Wavelength conversion on an optical fiber with six wavelengths.

adjacency set of λ_1 can be represented as $[1, 2]$ and the conversion distance is $2 - 1 = 1$. The conversion distances are the same for all wavelengths in this example; however, this is not required in our scheduling algorithms.

2.2 Interconnect Model

The WDM interconnect we consider is shown in Fig. 2. It has N input/output fibers and, on each fiber, there are k wavelengths that carry independent data. The wavelength channels on the input fibers are first demultiplexed, then each input channel is fed into a limited range wavelength converter to be converted to a proper wavelength. The output of the wavelength converter is then split into N copies of itself, one for each output fiber. The split signal is further split into $B + 1$ copies, one to each of the $B + 1$ optical delay lines (ODL) in front of the output fiber. The SOA gate following the splitter controls whether the signal can reach the ODL. The $B + 1$ ODLs for each output fiber are capable of delaying the optical packet for $0, 1, 2, \ldots, B$ time slots, respectively, and, similarly to the input/output fiber, a delay line can hold multiple signals if they are on different wavelengths. The outputs of these $B + 1$ delay lines are directly combined together and sent to the output fiber.

Fig. 2 only shows the switching fabric for data packets. The WDM interconnect also has a control component, though not shown in the figure, which makes the decision of whether an incoming packet can be accepted or not and, if accepted, which wavelength on which ODL it should be directed to. The switching fabric will be configured according to these decisions. The data packet has a header field preceding the data field, which contains the information of the destination of the packet. The packet header is converted to an electronic signal and sent to the control component. The data payload is not converted and remains in optical form; however, it will experience a fixed delay to give enough header processing time for the control component.

3 Problem Formalization

In this section, we show how the problem of maximizing network throughout and minimizing total delay in the WDM interconnect can be formalized into a weighted bipartite graph matching problem.

Consider the optical packets that have arrived at the WDM interconnect at a time slot. They can be partitioned into N subsets according to their destination fibers. Note that the decision of accepting or rejecting a packet in one subset does not affect the decisions for other subsets; hence, the scheduling for one output fiber can be done independently of other output fibers and, from now on, we will explain our algorithm for one output fiber only. The input to the scheduling algorithm is the packets destined to this fiber at this time slot. The output of the algorithm is the decisions of which packets are accepted and assigned to which wavelength channels on which ODLs.

At first, before the scheduling, the algorithm should find the *available wavelength channels*, which are the wavelength channels that can be assigned to the packets which have arrived at this time slot. At first glance, since there are $B + 1$ ODLs and, on each ODL, there are k wavelengths, there should be, in total, $k(B + 1)$ available wavelength channels. However, this is not true since not all of these $k(B + 1)$

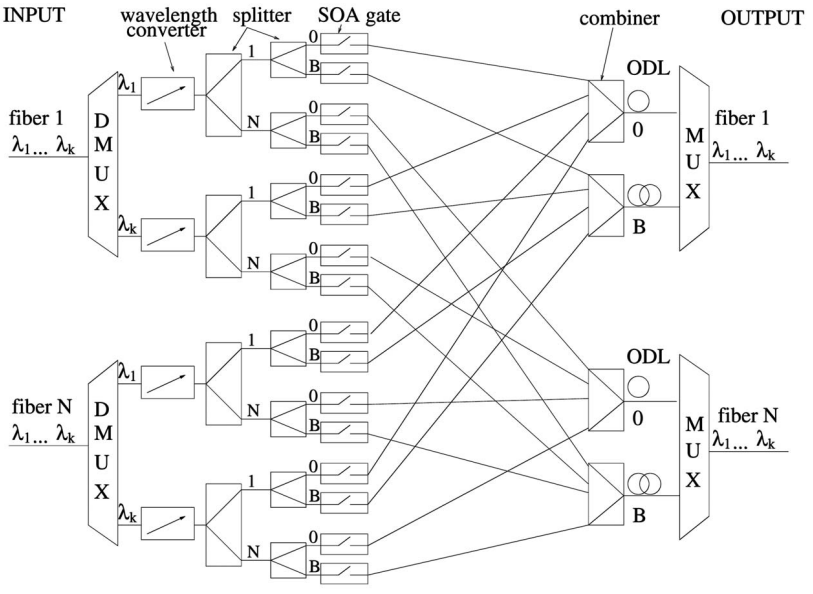


Fig. 2. A buffered wavelength convertible WDM interconnect.

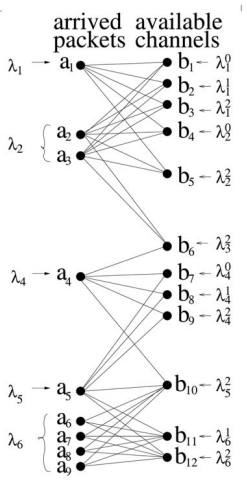


Fig. 3. A request graph where $N = 4$ and there are six wavelengths and three fiber delay lines and the conversion ability as defined in Fig. 1.

channels are available. From Fig. 2, we can see that the outputs of the $B+1$ ODLs are combined and sent to the output fiber. Hence, there should be no two packets on the same wavelength coming out of these ODLs at the same time slot; otherwise, collision will occur. For example, at the beginning of a time slot, if there is a packet coming out of ODL 1 on wavelength λ_1, channel λ_1 in ODL 0 should not be assigned to any newly arrived packets. More precisely, let λ_i^I represent wavelength channel λ_i on ODL I, for $1 \leq i \leq k$ and $0 \leq I \leq B$. At a time slot, channel λ_i^I is available if and only if there is no packet on wavelength λ_i that will come out of an ODL after I time slots. For example, λ_i^{B-1} is not available if a packet was directed to ODL B at the previous time slot and λ_i^{B-2} is not available if a packet on λ_i was directed to ODL $B-1$ at the previous time slot or to ODL B two time slots ago.

Given the buffer occupancy state, the set of available wavelength channels can be found in linear time. After this, we can draw a bipartite graph, which will be referred to as a *request graph*, as follows: In the request graph, left side vertices represent the arrived packets destined to this output fiber at this time slot. The vertices are sorted according to their wavelengths, lower wavelengths first. There might be several packets coming from different input fibers on the same wavelength and, in this case, their orders are arbitrary. The right side vertices represent the available wavelength channels, also sorted according to their wavelengths. There may also be more than one available wavelength channels on different ODLs of the same wavelength. In this case, we put the wavelength channel with a shorter delay in a higher position (though it is not necessary). Each vertex is given an index according to its position in the graph. A left side vertex will be denoted as a_u and a right side vertex will be denoted as b_i, where u and i are the indices of the vertices. There is an edge connecting a left side vertex a_u and a right side vertex b_i if the wavelength of the packet represented by a_u can be converted to the wavelength represented by b_i. Such a request graph is shown in Fig. 3, where $N = 4$ and there are three ODLs and the wavelength conversion is as defined in Fig. 1.

In a request graph G, let E denote the set of edges. Any wavelength assignment can be represented by E', which is a subset of E, where edge $a_u b_i \in E'$ if wavelength channel b_i is assigned to packet a_u. Under unicast traffic, any packet needs only one output channel. Also, an output channel can be assigned to only one packet. It follows that the edges in E' are vertex disjoint or E' is a *matching* in G since, if two edges share a vertex, either one packet is assigned to two wavelength channels or one wavelength channel is assigned to two packets. For a given set of packets, to maximize network throughput, we should find a maximum cardinality matching in the request graph. This problem for a bufferless WDM interconnect was studied and solved in [12].

However, in a buffered interconnect, although a maximum cardinality matching maximizes network throughput, it may not be a good schedule since the packet delay time should also be considered. Note that, in [12], the network is bufferless and there is no packet delay since a packet is either immediately transmitted or rejected. In the buffered interconnect, a packet that is granted, i.e., not dropped, may be able to be assigned to several ODLs and it is naturally preferred to assign it to the one with the shortest delay time. When considering all the arrived packets together, the total delay time of the granted packets should be minimized. Note that this is under the precondition that the maximum number of packets are granted since, otherwise, we can simply reject all arrived packets and the total delay time of the granted packets would be zero. Thus, the optimal schedule should have the following two properties: 1) The number of granted packets is maximum and 2) the total delay of the granted packets is minimum among all possible schedules where the same number of packets are granted.

To solve this problem, we can introduce weights to the right side vertices. We give weight $B - I + 1$ to right side vertices representing wavelength channels on ODL I, for $0 \leq I \leq B$. The weight of a matching then is defined as the total weight of the right side vertices it covers. We claim that a matching M with maximum weight will be the optimal schedule. This is because, first, M must be a maximum cardinality matching and, thus, the maximum number of packets is granted. If not, by graph theory, there must be an M augmenting path with which a new matching M' can be obtained which covers one more right side vertex than M while keeping all right side vertices covered by M still covered [20]. As a result, M' will have a larger weight than M since the weights of vertices are all positive, which contradicts the fact that M has maximum weight. Second, among all maximum matchings, M will give the smallest total delay time since, for a right side vertex, the larger the weight, the shorter the delay. For this reason, we will call M the *optimal matching* and will focus on methods for finding such optimal matchings in request graphs.

4 Properties of Request Graphs

Before moving on to solving the problem formalized in the previous section, we first briefly describe some properties of request graphs that have been shown in [12].

Property 1. *The adjacency set of any right side vertex is an interval. In the following, we use interval* $[begin(b_i), end(b_i)]$ *to represent the adjacency set of a right side vertex* b_i *and will*

TABLE 1
First Available Algorithm

```
First Available Algorithm
 for i := 1 to n do
    Find a_u which is the left side vertex adjacent to b_i with the
    smallest index and is not matched to any other vertex yet.
    if such a_u exists
       match a_u to b_i
    else
       b_i is not matched
    end if
 end for
```

call the left side vertices in this interval the neighbors *of* b_i. *The first and the last left side vertex in the adjacency set are called the "begin neighbor" and "end neighbor," respectively.*

Property 2. *If* $i < j$, *then* $begin(b_i) \leq begin(b_j)$ *and* $end(b_i) \leq end(b_j)$.

Property 3. *If edge* $a_u b_i \in E$, $a_v b_j \in E$ *while* $i < j$, $u > v$, *then* $a_v b_i \in E$, $a_u b_j \in E$.

Property 4. *Properties 1 and 2 also hold for left side vertices.*

Property 5. *Removing any vertex from the request graph, all the above properties still hold.*

The first two properties come directly from the observations of ordered interval wavelength conversion. (However, note that the intervals here are for the indices of the vertices, not for the wavelengths.) The fourth property says that left side vertices have the same properties as right side vertices. The fifth property says that any vertex induced subgraph has the same properties as the original request graph.

The third property can be called the *crossing edge property*, which will be frequently used in proving other properties of request graphs in this paper. It is so called because, if $i < j$ and $u > v$, $a_u b_i$ and $a_v b_j$ will appear crossing each other in the request graph. A direct consequence of this property is that there must be a maximum matching of a request graph with no crossing edges since any pair of crossing edges in the matching, say, $a_u b_i$ and $a_v b_j$, can be replaced with $a_v b_i$ and $a_u b_j$, which do not cross each other. Such a matching is called a noncrossing matching, in which the ith matched left side vertex is matched to the ith matched right side vertex.

A simple algorithm called the First Available Algorithm, described in Table 1, can be used for finding a maximum cardinality matching in a request graph. In the description of the algorithm, n is the number of right side vertices. This algorithm scans the right side vertices from the first to the last and matches a right side vertex to its first unmatched neighbor. This can be imagined as picking the "top" edge in the request graph and adding it to the matching in each iteration. Zhang and Yang [12] gave a proof for the following theorem.

Theorem 1. *The First Available Algorithm finds a maximum matching in a request graph.*

The time complexity of this algorithm is $O(n)$, where n is the number of right side vertices, since the loop is executed n times and, by keeping a pointer, the work in the loop can be done in constant time [12]. It can also be shown that the matching found by the First Available Algorithm is noncrossing. Since the left side vertices have the same properties as the right side vertices, this algorithm can also scan the left side vertices or "run on the left side."

5 The Scan and Swap Algorithm

In this section, we give a new algorithm, called the Scan and Swap Algorithm, for finding an optimal matching in the request graph or, equivalently, an optimal schedule for the buffered WDM interconnect.

5.1 Matroid Greedy Algorithm

Before presenting our algorithm, we first describe an algorithm for finding optimal matching in arbitrary bipartite graphs, which will be referred to as the matroid greedy algorithm since it can be derived from the matroid theory [19]. The matroid greedy algorithm serves as a guideline for finding optimal matching in all bipartite graphs. The Scan and Swap Algorithm can be regarded as an algorithm based on the matroid greedy algorithm specifically designed for request graphs.

A matroid is a structure defined on a finite whole set S and a family of subsets of S, with property usually referred to as independence defined on the elements of the subsets [19], [5]. For example, in a graph, the set of all vertices can be the whole set. A proper subset, which is a subset belonging to the matroid, is a group of vertices that can be covered by a matching. These vertices are said to be independent of each other in the matroid theory. Greedy algorithms can be used to find optimal solutions for problems defined on a matroid.

The idea of the matroid greedy algorithm is actually very simple. It will try to find a set of vertices that can be covered by an optimal matching by checking the weighted vertices one by one according to their weights, vertices with larger weights first. A vertex is added to the set if it can be covered along with vertices already in the set. To elaborate, the algorithm starts with an empty set Π. In Step s, it will check the weighted vertex with the sth largest weight, say, b. It checks whether there is a matching covering b and all the vertices added to Π previously. If yes, b will be added to Π and we say b is *selected*; otherwise, b is left uncovered. Update $s \leftarrow s + 1$ and repeat until all vertices have been checked. When the algorithm terminates, Π stores weighted vertices that can be covered by an optimal matching.

In general bipartite graphs, finding an optimal matching with the matroid algorithm needs $O(n^3)$ time since $O(n^2)$ time is needed to determine whether a vertex can be covered with the previously selected vertices by growing alternating paths, where n is the number of vertices. In bipartite graphs with some additional properties, other faster methods can be found to determine whether a vertex can be covered to reduce the complexity. For example, [23] showed that, in convex bipartite graphs, the optimal matching can be found in $O(m(m+n))$ time where a bipartite graph is convex if it satisfies Property 1 in Section 4 and m is the number of left side vertices and n is the number of right side vertices. The algorithm for convex bipartite graphs can be directly applied to our request graphs as well. However, since there can be up to

TABLE 2
List of Symbols

Π_I	:	the set of selected vertices after finishing stage I
π	:	the set of selected non-compulsory vertices after finishing a stage
$\pi(i)$	:	the set of vertices that have been added to π after step i
Ψ	:	the set of marked left side vertices
$\Psi(i)$	:	the set of vertices that have been added to Ψ after step i
Θ	:	the set of vertices matched to Ψ
$\Theta(i)$	:	the set of vertices that have been added to Θ after step i
M_i	:	the non-crossing matching that matches vertices in $\Psi(i)$ to vertices in $\Theta(i)$

Nk vertices on the left side and $(B+1)k$ vertices on the right side of a request graph, the running time will be $O(k^2N^2 + k^2BN)$, which may be too long and may not be suitable for a WDM interconnect. Next, we present a completely new algorithm, the Scan and Swap Algorithm, which can reduce the complexity to $O(Bk)$ by taking advantage of the unique properties of the request graphs. Some of the symbols used in the following sections are listed in Table 2.

5.2 Scan and Swap Algorithm

Consider the matroid greedy algorithm when applied to the request graph. As mentioned earlier, it will check the weighted vertices one by one according to their weights. We say the matroid greedy algorithm is working on "stage I" when it is checking vertices representing channels on ODL I or with weight $B - I + 1$, for $0 \leq I \leq B$. Let Π_I be the set of vertices the algorithm has selected when it has finished working on stage I. Note that, by the matroid theory, when finished with stage I, the algorithm should have added the maximum number of vertices with weight $B - I + 1$ to Π_{I-1} to obtain Π_I. Also note that the order by which the matroid greedy algorithm checks the vertices with the same weight does not change the outcome. Thus, an algorithm is functionally equivalent to the matroid greedy algorithm at stage I if it can find the maximum number of vertices with weight $B - I + 1$ that can be covered by a matching along with Π_{I-1}.

From now on, we will focus on finding such an algorithm. At stage I, we call vertices in Π_{I-1} the *compulsory* vertices because all these vertices must be still covered after stage I. All vertices with weight $B - I + 1$ are called the *noncompulsory* vertices because some of them can be left uncovered. The algorithm works on a subgraph of the request graph G, denoted as G_I, with the left side vertices being all the left side vertices in G and the right side vertices being the union of the compulsory and noncompulsory vertices. For presentational convenience, if there are n right side vertices in G_I, we label the right side vertices in G_I as b_1 to b_n, based on the values of their indices in G. The algorithm should find a matching that covers the maximum number of noncompulsory vertices under the constraint that all the compulsory vertices are still covered. The algorithm is called the *Scan and Swap Algorithm* and is described in Table 3. The output of the algorithm, which is the set of selected noncompulsory vertices, is stored in set π.

The algorithm works as follows: Initially, π is empty and all the left side vertices are not marked. Then, it starts to scan the right side vertices from the first to the last and will mark some left side vertices if needed. At Step i, when scanning to vertex b_i, it checks whether there is an unmarked neighbor of b_i. If yes, it will mark such a left side vertex with the smallest index and add b_i to π if b_i is a noncompulsory vertex. If all b_i's neighbors have been marked, if b_i is a noncompulsory vertex, it will proceed to the next vertex. Otherwise, i.e., if b_i is a compulsory vertex, it will swap out the noncompulsory vertex with the largest index in π.

To use this algorithm for finding an optimal matching in the original graph, we need to run this algorithm $B+1$ times, one for each stage. When stage I finishes, let $\Pi_I = \Pi_{I-1} \cup \pi$. When stage B finishes, the output Π_B is the set of the weighted vertices that can be covered by an optimal matching.

As an example, Fig. 4 and Fig. 5 show how the Scan and Swap Algorithm finds an optimal matching in the request graph in Fig. 3. In the figures, the compulsory vertices and noncompulsory vertices are shown in black and white, respectively. Fig. 4a shows stage 0, in which there are no compulsory vertices, and the three noncompulsory vertices denoted as b_1, b_2, and b_3 are the vertices representing wavelength channels on ODL0. (In Fig. 3, they are denoted as b_1, b_4, and b_7, respectively.) According to the algorithm, they will mark a_1, a_2, and a_4, respectively, and, since they all have marked some left side vertices, they will be added to π and will be the compulsory vertices of the next stage. Fig. 4b shows stage 1, in which the compulsory vertices are $\Pi_0 = \{b_1, b_3, b_4\}$ ($\{b_1, b_4, b_7\}$ in Fig. 3) and the three noncompulsory vertices are $\{b_2, b_5, b_6\}$ ($\{b_2, b_8, b_{11}\}$ in Fig. 3). When scanning the right side, the algorithm finds that they will mark a_1 to a_6 and, thus, all noncompulsory vertices will

TABLE 3
Scan and Swap Algorithm

Scan and Swap Algorithm
Set all the left side vertices unmarked.
$\pi \leftarrow \emptyset$.
for $i := 1$ **to** n **do**
  Find the left side vertex adjacent to b_i with the smallest index that has not been marked.
  if there is such a left side vertex
    Mark this left side vertex.
    if b_i is a non-compulsory vertex
      $\pi \leftarrow \pi \cup \{b_i\}$
    end if
  else
    if b_i is a compulsory vertex
      Let b_q be the non-compulsory vertex with the largest index in π
      $\pi \leftarrow \pi \setminus \{b_q\}$
    end if
  end if
end while

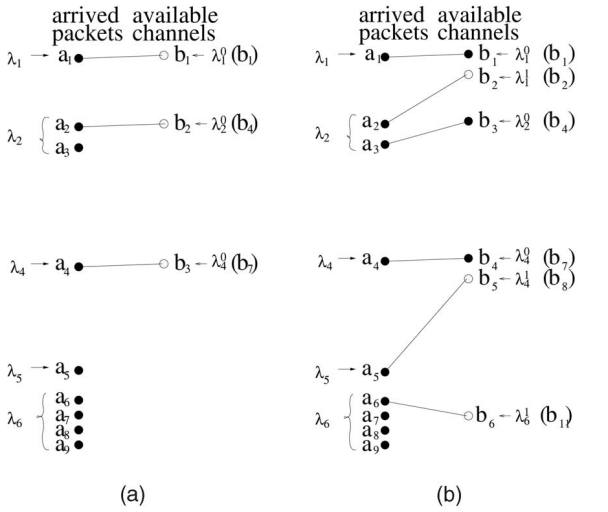


Fig. 4. Stages 0 and 1 of the Scan and Swap Algorithm when applied to the request graph in Fig. 3. Noncompulsory vertices are shown in white. The parenthesis to the right of the vertices contains the original notations for the same vertices in Fig. 3. (a) Stage 0. (b) Stage 1.

be added to π. Fig. 5 shows stage 2, in which the compulsory vertices are $\Pi_1 = \{b_1, b_2, b_4, b_7, b_8, b_{11}\}$ and the noncompulsory vertices are $\{b_3, b_5, b_6, b_9, b_{10}, b_{12}\}$ (the indices are the same as in Fig. 3). Note that, in the previous two stages, no vertex added to π was swapped out. However, such a swap occurs at Step 3 and Step 4 in stage 2, as shown in Fig. 5a and Fig. 5b. Fig. 5a shows Step 3 when the algorithm scans to b_3 and finds that a_3 is adjacent to b_3 and is not marked and, therefore, marks a_3 and adds b_3 to π. Fig. 5b shows Step 4 when the algorithm finds that all vertices adjacent to compulsory vertex b_4 have been marked and swaps b_3 out of π since it is the noncompulsory vertex in π with the largest index. When stage 2 finishes, $\pi = \{b_{10}, b_{12}\}$, and $\Pi_2 = \{b_1, b_2, b_4, b_7, b_8, b_{10}, b_{11}, b_{12}\}$, as shown in Fig. 5c. In fact, the edges in Fig. 5c are the optimal matching for the request graph in Fig. 3.

5.3 Correctness Proof of the Scan and Swap Algorithm

We now prove the correctness of the Scan and Swap Algorithm, i.e., that it finds the maximum number of noncompulsory vertices that can be covered along with all the compulsory vertices.

Some notions used in this section are defined as follows: After the algorithm has checked b_i, the set of vertices in π is denoted as $\pi(i)$. The set of marked left side vertices will be referred to as Ψ and vertices in Ψ after Step i will be denoted as $\Psi(i)$. As will soon be seen, there exists a matching, denoted as M_i, that matches vertices in $\Psi(i)$ to vertices in $\pi(i)$ and all the compulsory vertices with indices no more than i. The set of vertices matched to Ψ, including both the compulsory and noncompulsory vertices, will be referred to as Θ and the set of vertices in Θ after Step i will be denoted as $\Theta(i)$. We will also interchangeably use the terms "mark a left side vertex" and "add a left side vertex to Ψ."

In most cases, our theoretical results are simpler to state using set Θ than using π because of the one to one matching between vertices in Θ and Ψ. Therefore, in this section, we will mainly use Θ instead of π except for a few cases. Accordingly, the Scan and Swap Algorithm can be considered as working in the following way, which is equivalent to what was described in Section 5.2: Initially, Θ is empty and no left side vertices are marked. Then, the algorithm starts scanning the right side vertices from the first one to the last. At Step i, when scanning b_i, it checks whether there is an unmarked neighbor of b_i. If yes, add b_i to Θ and mark such neighbor with the smallest index. Otherwise, it checks whether b_i is a compulsory vertex and, if so, it will add b_i to Θ and swap out a noncompulsory vertex in Θ with the largest index. Then, it will proceed to the next vertex until all vertices have been scanned.

We claim that, after any Step i of the algorithm, the following three invariants hold:

1. There is a perfect matching from vertices in $\Theta(i)$ to vertices in $\Psi(i)$, i.e., there is a matching that matches all vertices in $\Theta(i)$ to all vertices in $\Psi(i)$.

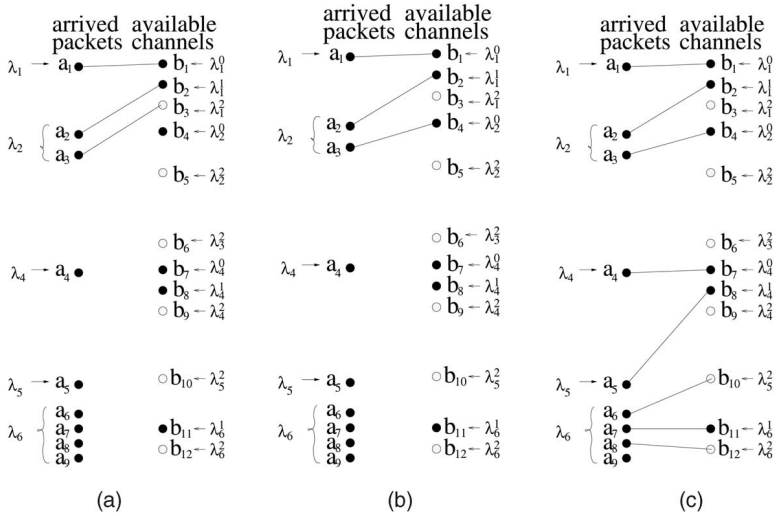


Fig. 5. Stage 2 of the Scan and Swap Algorithm when applied to the request graph in Fig. 3. Noncompulsory vertices are shown in white. (a) After Step 3. (b) After Step 4. (c) After stage 2 is finished.

2. All compulsory vertices with indices no more than i are in $\Theta(i)$. In addition, the maximum number of noncompulsory vertices with indices no more than i that can be covered by a matching along with these compulsory vertices is also in $\Theta(i)$.
3. Let a_u be the marked left side vertex with the largest index. Suppose there is another matching M' that covers all the compulsory vertices and $|\pi(i)|$ noncompulsory vertices among right side vertices with indices no more than i (where $|\pi(i)|$ is the cardinality of $\pi(i)$). Among all left side vertices that are matched to right side vertices with indices no more than i under M', let $a_{u'}$ be the one with the largest index. We have $u \leq u'$.

The first invariant guarantees that the vertices added to Θ can be matched. The second invariant guarantees that the maximum number of noncompulsory vertices have been added to Θ. If these two invariants are true, the Scan and Swap Algorithm must be correct. The third invariant says that the algorithm will not mark a left side vertex if it can mark another vertex with a smaller index and is used in the proof for the second invariant.

We will show the invariants are true by induction on i, the steps of the algorithm. Note that if the first right side vertex is isolated, it can be removed from the graph, hence, from now on, we only consider request graphs where the first right side vertex is not isolated. In this case, it is clear that, after the first step, the three invariants are true. Thus, we will assume they are true for all steps before i. Also note that, due to the crossing edge property of the request graph, we can assume matching $M_1, M_2, \ldots, M_{i-1}$ are all noncrossing. That is, for any $j < i$, under M_j, the vertex in $\Psi(j)$ with the lth smallest vertex is matched to the vertex in $\Theta(j)$ with the lth smallest vertex.

The following propositions can be obtained without much difficulty:

Proposition 1. *If a_u was added to Ψ before a_v, then $u < v$.*

Proposition 2. *The indices of left side vertices matched to the same right side vertex will not increase. That is, suppose, at Step j, b_j is added to Θ and is matched to a_u under M_j. If b_j is not swapped out in later steps, under $M_{j+1}, M_{j+2}, \ldots, b_j$ will only be matched to vertices with indices no more than a_u.*

Proposition 3. *The indices of right side vertices matched to the same left side vertex will not decrease.*

We also have:

Lemma 1. *Suppose, after Step $i-1$, a_{u-1} is not marked but a_u is marked and a_u is matched to b_j under M_{i-1}. Then, a_u must be the begin neighbor of b_j.*

Proof. By contradiction. If a_u is not the begin neighbor of b_j, b_j is adjacent to a_{u-1}. Consider when b_j was first added to Θ at Step j. Since a_{u-1} is not marked, there must be some unmarked neighbor of b_j with an index less than a_{u-1} since, otherwise, the algorithm will mark a_{u-1}. Suppose the one with the smallest index is a_v, then, under M_j, b_j was matched to a_v. By Proposition 2, $u \leq v$. But, this contradicts the fact that $u - 1 > v$. □

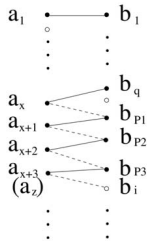


Fig. 6. A possible M_{i-1}-alternating path, where the solid lines are edges in M_{i-1} and dashed lines are edges not in M_{i-1}. The black nodes are vertices in $\Theta(i-1)$ and $\Psi(i-1)$.

We are now in a position to show that the invariants are true.

Lemma 2. *The first invariant of the Scan and Swap Algorithm is true after Step i.*

Proof. At Step i, the algorithm checks b_i. First, consider when b_i has some unmarked neighbor. In this case, b_i can be added to Θ and the algorithm will mark one of its neighbors, which will be added to Ψ. Since one vertex is added to Θ and Ψ each and these two vertices can be matched to each other, the invariant is true.

If all b_i's neighbors have been marked and b_i is a noncompulsory vertex, b_i will not be added to Θ and Ψ will not be changed, thus the invariant is still true. Hence, we only need to consider when all b_i's neighbors have been marked while b_i is a compulsory vertex. In this case, we prove the invariant is true by establishing an M_{i-1} alternating path starting at b_i and ending at b_q, where b_q is the noncompulsory vertex in $\Theta(i-1)$ with the largest index. An M_{i-1} alternating path is a path that alternates between edges in M_{i-1} and not in M_{i-1}. If such a path can be found, we can do a "flip" operation to the edges in the alternating path, that is, remove the edges that are in M_{i-1} and add in edges that are not in M_{i-1}. After doing this change to M_{i-1}, the new matching covers b_i and all vertices that were covered by M_{i-1} except b_q. Thus, if the alternating path can be found, the invariant will still be true. As an example, Fig. 6 shows a possible alternating path, where the solid lines are edges in M_{i-1} and dashed lines are edges not in M_{i-1}. Clearly, if the vertices are matched according to the dashed edges, b_i will be matched and b_q will not.

We now show that such an alternating path must exist. Suppose there are l right side vertices in $\Theta(i-1)$ between b_q and b_i, denoted as $b_{P1}, b_{P2}, \ldots, b_{Pl}$. In Fig. 6, $l = 3$. By the choice of b_q, they must all be compulsory vertices. Also suppose, under M_{i-1}, b_q is matched to a_x. Suppose the end neighbor of b_i is a_z. a_z must have been marked and, due to Property 2 of the request graph, a_z is the marked vertex with the largest index.

We first claim that left side vertices with indices in interval $[x, z]$ are all marked after Step $i-1$. To see this, suppose there are some unmarked vertices among them and let a_y be the one with the largest index. Note that, due to the choice of a_y, a_{y+1} must be marked and suppose it is matched to b_{Ph} under M_{i-1}. By Lemma 1, a_{y+1} is the begin neighbor of b_{Ph}. Let V_l be the set of left side vertices with indices in interval $[y+1, z]$. Let V_r be the set of right

side vertices in $\Theta(i-1)$ with indices no less than b_{Ph}. Let A_r be the union of the adjacency sets of vertices in V_r. Note that vertices in V_l are all marked and that they are only matched to right side vertices with indices no less than b_{Ph} since M_{i-1} is noncrossing, therefore $V_l \subseteq A_r$. On the other hand, since a_{y+1} is the begin neighbor of b_{Ph} and a_z is the end neighbor of b_i, $V_l \supseteq A_r$. Thus, $A_r = V_l$. However, if $A_r = V_l$, there cannot be a matching covering all vertices in $V'_r = V_r \cup \{b_i\}$, which contradicts the fact that such a matching must exist since vertices in V'_r are all compulsory vertices.

As a result of this fact, a_z must be a_{x+l} and a_{x+w} must be matched to b_{Pw} for all $1 \le w \le l$. Using a similar method, we can also show that b_{Pw} must be adjacent to a_{x+w-1} for all $1 \le w \le l$. Thus, the alternating path can be $b_i, a_{x+l}, b_{Pl}, a_{x+l-1}, \ldots, a_x, b_q$, as shown in Fig. 6. □

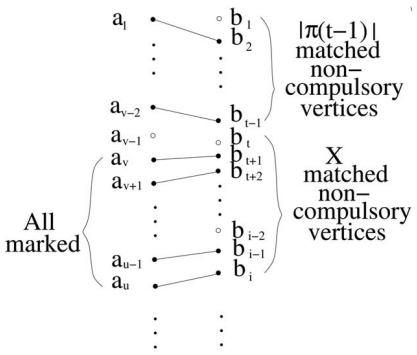


Fig. 7. Illustration of Lemma 4. The lines are edges in M_i. Black nodes are vertices in $\Theta(i)$ and $\Psi(i)$. a_v to a_u are all marked. a_v was first marked by b_t. In this example, b_t was swapped out later and a_v is matched to b_{t+1} in M_i.

Lemma 3. *Invariant 2 of the Scan and Swap Algorithm is true after Step i.*

Proof. First, consider when not all b_i's neighbors are marked. If b_i is a noncompulsory vertex, it can be added to Θ. Invariant 2 is true since, otherwise, there is another matching that covers more than $|\pi(i-1)|+1$ noncompulsory vertices from b_1 to b_i and the same matching must cover more than $|\pi(i-1)|$ noncompulsory vertices from b_1 to b_{i-1}, which is a contradiction to our inductive hypothesis that Invariant 2 is true after Step $i-1$. For similar reasons, if b_i is a compulsory vertex, Invariant 2 is also true.

Therefore, we only need to consider when all b_i's neighbors are marked. In this case, if b_i is a noncompulsory vertex, it cannot be added to Θ. If Invariant 2 is not true, then there exists a matching that covers all compulsory vertices from b_1 to b_i and more than $|\pi(i)|$ noncompulsory vertices from b_1 to b_i. For any such matching M'', consider a subset of it denoted as M', where an edge in M'' is also in M' if it covers a right side vertex with an index no more than b_i. Note that M' must cover b_i since, otherwise, Invariant 2 cannot be true after Step $i-1$. Let a_u be the end neighbor of b_i and, first, suppose it is matched under M'. Let the vertices matched to a_u and b_i under M' be b_j and a_v, respectively. Note that $j < i$ and $u > v$, therefore, due to the crossing edge property, we can match a_v to b_j and match a_u to b_i. After this change to M', all right side vertices covered by M' with smaller indices than b_i are matched to left vertices with smaller indices than a_u. But, this contradicts the inductive hypothesis that Invariant 3 is true after Step $i-1$. A similar contradiction can be found if a_u is not matched under M'. Therefore, Invariant 2 is true after Step i if b_i is a noncompulsory vertex. Similarly, it can be shown that Invariant 2 is true if b_i is a compulsory vertex. □

Lemma 4. *Invariant 3 of the Scan and Swap Algorithm is true after Step i.*

Proof. To show that Invariant 3 is true after Step i, we need to use the fact that Invariants 1 and 2 are true, that is, after Step i, the Scan and Swap Algorithm finds a matching M_i that covers the maximum number of noncompulsory vertices from b_1 to b_i and all the compulsory vertices from b_1 to b_i. Let a_u be the left side vertex covered by M_i with the largest index. This situation is shown in Fig. 7.

Note that if all left side vertices from a_1 to a_u are marked after Step i, the invariant is obviously true. Hence, from now on, we only consider when some left side vertices with indices between 1 and u are not marked at Step i. Among such vertices, let a_{v-1} be the one with the largest index. Suppose the vertex following a_{v-1}, a_v, was first marked at Step t, when the algorithm scanned to b_t. We claim that the vertices added to Θ prior to b_t, i.e., vertices in $\Theta(t-1)$, will not be swapped out of Θ after Step $t-1$.

This claim is true because of two facts. First, due to Proposition 3, a_v will only be matched to a vertex with an index no smaller than b_t in $M_t, M_{t+1}, \ldots, M_i$. Second, due to Lemma 1, a_v must be the begin neighbor of b_t. As a result of these two facts, after Step t, when a compulsory vertex needs to swap out a noncompulsory vertex, the alternating path will never reach right side vertices with indices less than b_t. The claim is thus proven.

Suppose the invariant is not true, then there is matching M', as described in the invariant, that satisfies $u' < u$, where $a_{u'}$ is the left side vertex with the largest index that is matched to a right side vertex with an index no more than i under M'. Note that, after Step i, vertices from a_v to a_u are all marked. In M_i, they are matched to some of the right side vertices from b_t to b_i, hence, in M_i, there are $u-v+1$ matched vertices from b_t to b_i. Suppose, among these vertices, X are noncompulsory vertices (from previous discussions, we know that $X = |\pi(i)| - |\pi(t-1)|$). Note that, in M', there must be less than X matched noncompulsory vertices from b_t to b_i because only vertices from a_v to a_{u-1} can be matched to b_t to b_i, thus, from b_t to b_i, M' covers at most $u-v$ vertices. As a result, from b_1 to b_{t-1}, there must be more vertices covered by M' than by M_i. But, this contradicts the fact that, after Step $t-1$, $\Theta(t-1)$ contains the maximum number of noncompulsory vertices from b_1 to b_{t-1} that can be covered and they will not be swapped out of Θ later. That completes our proof. □

Combining these lemmas, we have:

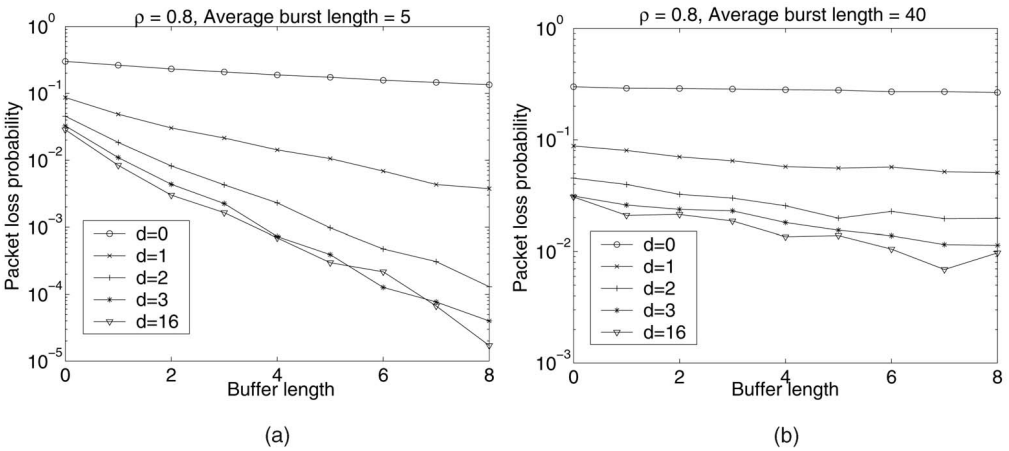


Fig. 8. Packet loss probability of the WDM interconnect under bursty traffic. The load is $\rho = 0.8$. (a) Average burst length is five time slots. (b) Average burst length is 40 time slots.

Theorem 2. *The Scan and Swap Algorithm finds the maximum number of noncompulsory vertices that can be covered along with all the compulsory vertices in a request graph.*

5.4 Implementation Issues and Complexity Analysis

In this section, we discuss the implementation and complexity of the Scan and Swap Algorithm. Note that, according to Table 3, the complexity of the Scan and Swap Algorithm should be $O(Bk)$, where B is the number of ODLs and k is the number of wavelengths. However, we will show in this section that, by using some special techniques and data structures, it can be reduced to $O(k)$.

The input to the Scan and Swap Algorithm is the set of compulsory vertices, the set of noncompulsory vertices, and the set of left side vertices. They can be represented by $1 \times k$ vectors, where component i in the vector is the number of vertices on wavelength λ_i. From now on, they will be referred to as the "compulsory vector," the "noncompulsory vector," and the "left side vector" and be denoted as C, N, and L, respectively. Accordingly, their components will be written as c_i, n_i, and l_i, respectively. For example, corresponding to Fig. 5, $C = [2, 1, 0, 2, 0, 1]$, $N = [1, 1, 1, 1, 1, 1]$, and $L = [1, 2, 0, 1, 1, 4]$. Note that the noncompulsory vector must be a binary vector.

A stack can be used to store the indices of vertices in π. When a noncompulsory vertex is added to π, its index will be pushed into this stack. Later, some of the vertices might have to be swapped out. Based on the algorithm, they will be the ones that were most recently added to π, therefore they are at the top of the stack and to swap them out is simply to perform several pop operations. Note that a vertex that is swapped out of π will never be added to π again. Thus, the overall time spent in stack operations is $O(k)$.

The key observation concerning the complexity of the algorithm is that the vertices on the same wavelength need not be treated one by one because they have the same connection patterns. As a result, rather than scanning the individual vertices, the algorithm only needs to scan the *wavelengths* one by one, i.e., scan the $1 \times k$ vectors, which leads to the $O(k)$ time complexity. To elaborate, suppose the algorithm is scanning the compulsory vertices on λ_i. A pointer p pointing to λ_p will be maintained, where λ_p can be converted to λ_i and is the first such wavelength on which there are some unmarked left side vertices. Let l'_p be the number of unmarked left side vertices on λ_p. According to the algorithm, the first c_i unmarked left side vertices need to be found for the c_i compulsory vertices on λ_i. It can be done by adding l'_p with $l_{p+1}, l_{p+2}, \ldots, l_q$ until the summation result exceeds c_i or until λ_q is the last wavelength convertible from λ_i. In the latter case, some vertices should be popped out of the stack. Then, update p and l'_p accordingly, which is equivalent to marking some left side vertices. The noncompulsory vertex on λ_i can be treated in a similar way. Note that l'_p needs to be updated no more than $2k$ times and a component in L can be involved in the addition no more than once. Thus, the complexity of the algorithm is $O(k)$. After finishing stage I, to get Π_I, one simply increments some components in the compulsory vector according to the content of the stack, which also takes $O(k)$ time.

In our applications, the Scan and Swap Algorithm needs to run $B + 1$ times. Thus, when using this algorithm for optimal matching, $O(kB)$ time is needed, where B is the length of the longest delay line and k is the number of wavelengths per fiber. Note that the complexity is linear to the input size, thus the algorithm has the lowest possible order of complexity since any algorithm needs to scan the input at least once. Also note that the Scan and Swap Algorithm is very simple and can be implemented in hardware. These features make it a practical algorithm for optical interconnects.

6 Simulation Results

We implemented Scan and Swap Algorithm in software and conducted simulations. The interconnect in simulations has 16 input fibers and 16 output fibers with 16 wavelengths on each fiber. An input channel alternates between two states, the "busy" state and the "idle" state. When in the "busy" state, it continuously receives packets and all the packets, called a "burst," go to the same destination; when in the "idle" state, it does not receive any packets. The length of the busy and idle periods follows geometric distribution. For each experiment, the simulation program was run for 100,000 time slots.

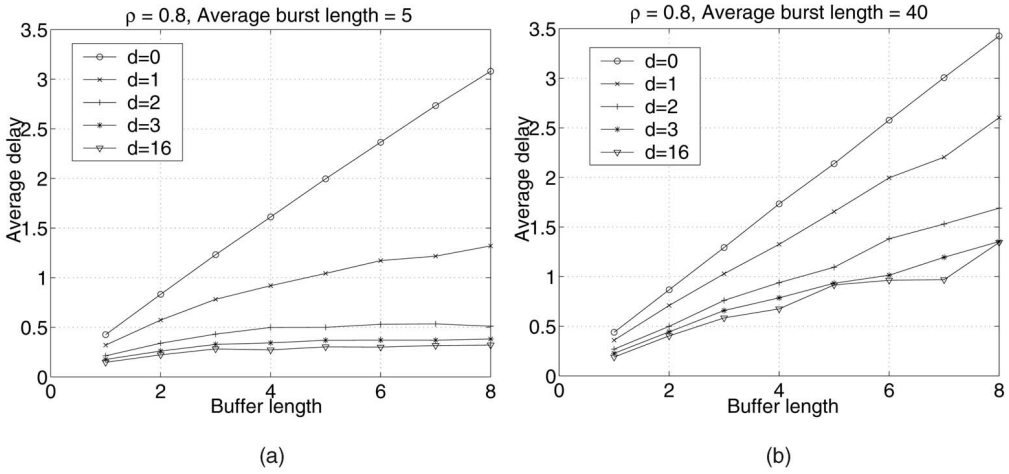


Fig. 9. Average delay of the WDM interconnect under bursty traffic. The load is $\rho = 0.8$. (a) Average burst length is five time slots. (b) Average burst length is 40 time slots.

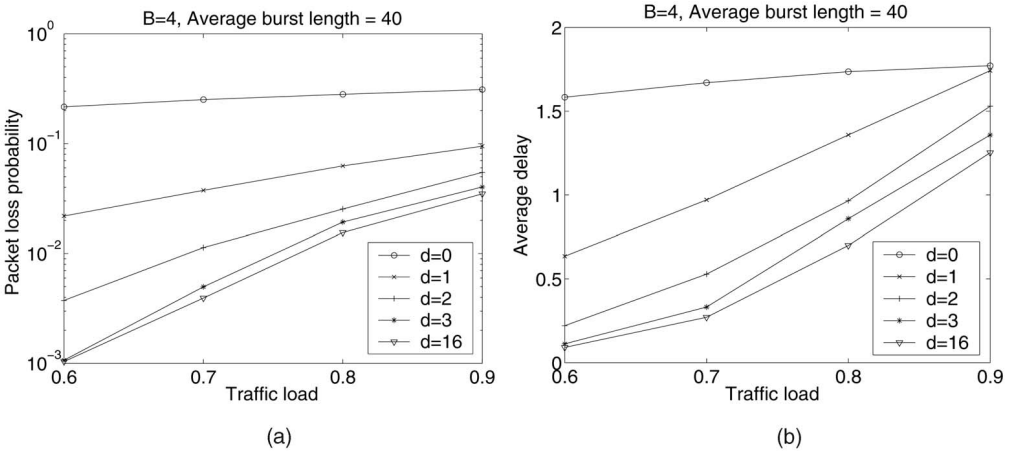


Fig. 10. Packet loss probability and average delay under different traffic load when the average burst length is 40. (a) Packet loss probability. (b) Average delay.

Fig. 8 shows the packet loss probability as a function of the number of fiber delay lines when the traffic load is $\rho = 0.8$. The average burst lengths in Fig. 8a and Fig. 8b are five time slots and 40 time slots, respectively. As expected, packet loss probability decreases as the number of delay lines increases. For example, in Fig. 8a, for conversion distance $d = 2$, when $B = 0$ (no buffer), the packet loss probability is about $10^{-1.3}$; however, when $B = 4$, it is reduced to about 10^{-3}. As the traffic becomes more bursty, i.e., as the average burst length increases, the packet loss probability decreases much more slowly with the buffer depth, as can be observed in Fig. 8b, where the curves are almost flat. This is because, when the burst is too long, the buffer capacity will always be exceeded.

It can also be seen that, with the same buffer length, a larger conversion distance always results in a smaller packet loss probability. Also, when the burst is too long, increasing buffer length does not yield too much benefit, but increasing conversion distance always does. For example, in Fig. 8b, when $d = 1$, increasing buffer length does not reduce much of the packet loss probability, but, when d is increased to 2, the packet loss probability drops by almost $10^{-0.4}$. This suggests that wavelength conversion ability is more important than buffering in a WDM interconnect. However, we observe that only a relatively small conversion distance is needed to achieve good performance. As can be seen in Fig. 8, the packet loss probability for $d = 3$ is already very close to that for $d = 16$ (full range conversion). This is exactly the reason to use limited range wavelength converters instead of full range wavelength converters.

Fig. 9 shows the average delay of a packet as a function of the number of fiber delay lines, where the traffic is the same as in Fig. 8. It can be seen that, as the buffer length increases, the average packet delay also increases since fewer packets are dropped and, thus, more are directed to a buffer before being actually transmitted. For the same buffer size, a larger conversion distance results in a shorter average delay. As in Fig. 9a, when $B = 4$, the average delay for $d = 1$ is about 0.9 time slot and the average delay for $d = 3$ is only about 0.3 time slot.

Fig. 10 shows the packet loss probability and average delay as a function of the traffic load when $B = 4$ and the burst length is 40. Similar facts can be observed as in the previous figures, that is, increasing the conversion ability will greatly improve the performance and good performance can be achieved with a relatively small conversion distance.

7 Conclusions

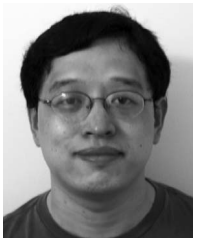
In this paper, we have studied optimal scheduling in buffered WDM optical interconnects with limited range wavelength conversion ability. We formalized the problem as a weighted bipartite graph matching problem and showed that maximum network throughput and minimum delay can be achieved by finding an optimal matching in the bipartite graph. We gave the Scan and Swap Algorithm for finding the optimal matching in $O(kB)$ time, where k is the number of wavelengths per fiber and B is the buffer length, compared to $O(k^2N^2 + k^2BN)$ time by directly adopting other existing algorithms for more general weighted bipartite graphs where N is the number of input fibers.

Acknowledgments

This research was supported in part by the US National Science Foundation under grant numbers CCR-0073085, CCR-0207999, and ECS-0427345.

References

[1] B. Mukherjee, "WDM Optical Communication Networks: Progress and Challenges," *IEEE J. Selected Areas in Comm.*, vol. 18, no. 10, pp. 1810-1824, Oct. 2000.
[2] D.K. Hunter, M.C. Chia, and I. Andonovic, "Buffering in Optical Packet Switches," *J. Lightwave Technology,* vol. 16, no. 12, pp. 2081-2094, 1998.
[3] M. Kovacevic and A. Acampora, "Benefits of Wavelength Translation in All-Optical Clear-Channel Networks," *IEEE J. Selected Areas in Comm.*, vol. 14, no. 5, pp. 868-880, June 1996.
[4] S.L. Danielsen, C. Joergensen, B. Mikkelsen, and K.E. Stubkjaer, "Analysis of a WDM Packet Switch with Improved Performance under Bursty Traffic Conditions Due to Tuneable Wavelength Converters," *J. Lightwave Technology,* vol. 16, no. 5, pp. 729-735, May 1998.
[5] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms.* The MIT Press, 1990.
[6] T. Tripathi and K.N. Sivarajan, "Computing Approximate Blocking Probabilities in Wavelength Routed All-Optical Networks with Limited-Range Wavelength Conversion," *IEEE J. Selected Areas in Comm.*, vol. 18, pp. 2123-2129, Oct. 2000.
[7] G. Shen, S.K Bose, T.H. Cheng, C. Lu, and T.Y. Chai, "Performance Study on a WDM Packet Switch with Limited-Range Wavelength Converters," *IEEE Comm. Letters,* vol. 5, no. 10, pp. 432-434, Oct. 2001.
[8] L. Xu, H.G. Perros, and G. Rouskas, "Techniques for Optical Packet Switching and Optical Burst Switching," *IEEE Comm. Magazine,* pp. 136-142, Jan. 2001.
[9] R. Ramaswami and G. Sasaki, "Multiwavelength Optical Networks with Limited Wavelength Conversion," *IEEE/ACM Trans. Networking,* vol. 6, pp. 744-754, Dec. 1998.
[10] X. Qin and Y. Yang, "Nonblocking WDM Switching Networks with Full and Limited Wavelength Conversion," *IEEE Trans. Comm.*, vol. 50, no. 12, pp. 2032-2041, Dec. 2002.
[11] Y. Yang, J. Wang, and C. Qiao, "Nonblocking WDM Multicast Switching Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 11, no. 12, pp. 1274-1287, Dec. 2000.
[12] Z. Zhang and Y. Yang, "Optimal Scheduling in WDM Optical Interconnects with Arbitrary Wavelength Conversion Capability," *IEEE Trans. Parallel and Distributed Systems,* vol. 15, no. 11, pp. 1012-1026, Nov. 2004.
[13] S.L. Danielsen, B. Mikkelsen, C. Joergensen, T. Durhuus, and K.E. Stubkjaer, "WDM Packet Switch Architectures and Analysis of the Influence of Tunable Wavelength Converters on the Performance," *J. Lightwave Technology,* vol. 15, no. 2, pp. 219-227, Feb. 1998.
[14] N. McKeown, P. Varaiya, and J. Warland, "Scheduling Cells in an Input-Queued Switch," *IEEE Electronic Letters,* pp. 2174-2175, Dec. 1993.
[15] N. McKeown, "The iSLIP Scheduling Algorithm Input-Queued Switch," *IEEE/ACM Trans. Networking,* vol. 7, pp. 188-201, Apr. 1999.
[16] H. Qin, S. Zhang, and Z Liu, "Dynamic Routing and Wavelength Assignment for Limited-Range Wavelength Conversion," *IEEE Comm. Letters,* vol. 5, no. 3, pp. 136-138, Mar. 2003.
[17] X. Masip-Bruin, S. Sanchez-Lopez, J. Sole-Pareta, J. Domingo-Pascual, and D. Colle, "Routing and Wavelength Assignment under Inaccurate Routing Information in Networks with Sparse and Limited Wavelength Conversion," *Proc. IEEE 2003 Global Telecomm. Conf. (IEEE GLOBECOM '03),* vol. 5, pp. 2575-2579, Dec. 2003.
[18] R. Ramaswami and K.N. Sivarajan, *Optical Networks: A Practical Perspective,* first ed. Academic Press, 2001.
[19] E.L. Lawler, *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, 1976.
[20] D.B. West, *Introduction to Graph Theory.* Prentice Hall, 1996.
[21] J. Hopcroft and R. Karp, "An $n^{\frac{5}{2}}$ Algorithm for Maximum Matchings in Bipartite Graph," *SIAM J. Computing,* vol. 2, no. 4, pp. 225-231, Dec. 1973.
[22] F. Glover, "Maximum Matching in Convex Bipartite Graph," *Naval Research Logistics Quarterly,* vol. 14, pp. 313-316, 1967.
[23] W. Lipski Jr. and F.P. Preparata, "Algorithms for Maximum Matchings in Bipartite Graphs," *Naval Research Logistics Quarterly,* vol. 14, pp. 313-316, 1981.



Zhenghao Zhang received the BEng and MS degrees in electrical engineering from Zhejiang University, People's Republic of China, in 1996 and 1999, respectively. From 1999 to 2001, he worked in industry as a software engineer in embedded systems design. Since 2001, he has been working toward the PhD degree in the Department of Electrical and Computer Engineering at the State University of New York at Stony Brook. His research interests include scheduling and performance analysis of optical networks and wireless networks. He is a student member of the IEEE and the IEEE Computer Society.

Yuanyuan Yang received the BEng and MS degrees in computer science and engineering from Tsinghua University, Beijing, China, and the MSE and PhD degrees in computer science from Johns Hopkins University, Baltimore, Maryland. She is a professor of computer engineering and computer science at the State University of New York at Stony Brook. Her research interests include parallel and distributed computing and systems, high-speed networks, optical and wireless networks, and high-performance computer architecture. Her research has been supported by the US National Science Foundation (NSF) and US Army Research Office (ARO). Dr. Yang has published extensively in major journals and refereed conference proceedings and holds six US patents in these areas. She is an editor for the *IEEE Transactions on Parallel and Distributed Systems* and for the *Journal of Parallel and Distributed Computing*. Dr. Yang has served on NSF review panels and program/organizing committees of numerous international conferences in her areas of research. She is a senior member of the IEEE and a member of the IEEE Computer Society.