The Scheduling and Wavelength Assignment Problem in Optical WDM Networks

Evripidis Bampis and George N. Rouskas, *Senior Member, IEEE*

***Abstract*—We consider a scheduling problem, which we call the *scheduling and wavelength assignment* (SWA) problem, arising in optical networks that are based on the wavelength-division-multiplexing (WDM) technology. We prove that the SWA problem is $\mathcal{NP}$-complete for both the preemptive and the nonpreemptive cases. Furthermore, we propose two efficient approximation algorithms. The first is for the preemptive case and is based on a natural decomposition of the problem to the classical multiprocessor scheduling and open-shop problems. For the nonpreemptive case, we prove that a naive implementation of list scheduling produces a schedule that can be m times far from the optimum, where m is the number of processors (equivalently, WDM channels). Finally, we give a more refined version of list scheduling and we prove it to be a 2-approximation algorithm for both the off-line and the on-line contexts.**

***Index Terms*—Optical networks, packet scheduling, wavelength assignment, wavelength division multiplexing.**

I. Introduction

THE spectacular growth in data traffic and the surging demand for diverse services has led to a dramatic increase in demand for data transmission capacity. Recent advances in wavelength-division-multiplexing (WDM) technology [4], [10] are expected to provide solutions to this challenge. WDM supports multiple simultaneous channels, each on a different wavelength, on a single fiber. WDM systems operating at aggregate rates exceeding one terabit per second have been demonstrated, while systems supporting rates of tens of gigabits per second are becoming commercially available.

As future networks based on WDM technology are developed to support data traffic and the Internet, they must be designed and optimized for that purpose. In particular, a number of new and challenging problems arise in the area of scheduling data packets over multiple wavelengths, both in a local area environment [1], [20] and in a backbone network consisting of IP routers [18]. In this paper, we consider a scheduling problem with applications to packet-switched optical WDM networks, and we prove that it is $\mathcal{NP}$-complete for both the preemptive and the nonpreemptive cases. We then present two efficient approximation algorithms for this problem. For the preemptive case, the approximation algorithm is based on a natural decomposition of the problem into the classical multiprocessor scheduling and open-shop problems. For the nonpreemptive case, we develop two list-scheduling algorithms, the second of which is a 2-approximation algorithm for both the on-line and off-line contexts.

The paper is organized as follows. In Section II, we define the scheduling problem under study, and we motivate it by describing its relationship to packet scheduling in WDM optical networks. In Section III, we present an off-line approximation algorithm for preemptive scheduling, and in Section IV, we present both off-line and on-line algorithms for the nonpreemptive case. We conclude the paper in Section V.

Manuscript received June 4, 2001; revised January 17, 2002. The work of E. Bampis was supported in part by the Thematic Network of the European Union Approximation and Online Algorithms for Optimization Problems (APPOL II). The work of G. N. Rouskas was supported by the Université d'Evry Val d'Essonne, by the National Science Foundation under Grant NCR-9701113 (CAREER), and by the Defense Advanced Research Projects Agency under Grant F-30602-00-C-0034. This work was performed while G. N. Rouskas was visiting the Université d'Evry Val d'Essone.

E. Bampis is with the Laboratoire de Méthodes Informatiques, Université d'Evry Val d'Essone, 91025 Evry Cedex, France (e-mail: bampis@lami.univ-evry.fr).

G. N. Rouskas is with the Department of Computer Science, North Carolina State University, Raleigh, NC 27695-7534 USA (e-mail: rouskas@csc.ncsu.edu).

Publisher Item Identifier S 0733-8724(02)03196-1.

II. Problem Definition and Applications

A. The Scheduling and Wavelength Assignment (SWA) Problem

We consider a set $\mathcal{M}$ of m processors and a set $\mathcal{J}$ of $n, n > m$ jobs. Each job $J_d \in \mathcal{J}$ is defined as a set of n operations, $J_d = \{O_{1d}, O_{2d}, \ldots, O_{nd}\}$, and p_{sd} denotes the processing time of operation O_{sd}. The objective is to schedule the n jobs on the m processors so as to minimize the *makespan*, or maximum finish time of the schedule, subject to the following constraints.

C1) All operations of job J_d are executed on the same processor.

C2) The operations O_{sd} and $O_{sd'}$ cannot be simultaneously executed, for all $s, d \neq d'$.

C3) A processor may execute at most one operation at any time instant.

Constraints C1–C3 define a set of *compatibility* constraints among the different operations. Specifically, two operations O_{sd} and $O_{s'd'}$ are said to be *incompatible* if either $s = s'$ or $d = d'$; otherwise, the operations are *compatible*. Incompatible operations cannot be executed simultaneously. Constraint C1 also implies that once an operation of job J_d has been executed on a processor x, then all operations $O_{sd}, s = 1, \ldots, n$, must be executed on the same processor x. However, the processor on which the operations of a job J_d are executed is not known in advance; rather, it is determined as part of the solution to the scheduling problem. Furthermore, the operations of a job J_d can be executed on processor x in any order.

The scheduling problem defined above can be logically decomposed into two sub-problems. Because of constraint C1, the first sub-problem is to assign each job J_d to a processor x, meaning that all operations of J_d will be executed on x. Given this assignment of jobs to processors, the second sub-problem is to schedule the operations on their assigned processors so as to minimize the makespan, while also satisfying the compatibility constraints C2 and the processor constraint C3. This decomposition leads to a natural way of solving the scheduling problem by applying existing algorithms to each sub-problem in isolation, as discussed later. However, we will also show that it is possible to design algorithms to simultaneously solve both sub-problems, and these algorithms are more efficient than the two-step approach described above.

We will refer to this type of scheduling problem as the *scheduling and wavelength assignment* (SWA) problem, since it arises naturally in packet-switched optical WDM networks, as we explain in Section II-C. We have also chosen to use s and d as subscripts for the operations to reflect the fact that, in the network settings described in Section II-C, operation O_{sd} corresponds to the amount of traffic to be transmitted from a source s to a destination d in the network.

We now introduce notation that will be useful in later sections. Let T_{seq} denote the sum of the processing times of all operations, let $D_{\max}$ be the maximum amount of processing time required by any job, and let $S_{\max}$ denote the maximum amount of processing time associated with any source s:

$$T_{\text{seq}} = \sum_{s=1}^{n} \sum_{d=1}^{n} p_{sd} \tag{1}$$

$$D_{\max} = \max_{d=1,\ldots,n} \left\{ \sum_{s=1}^{n} p_{sd} \right\} \tag{2}$$

$$S_{\max} = \max_{s=1,\ldots,n} \left\{ \sum_{d=1}^{n} p_{sd} \right\}. \tag{3}$$

Let $C_{\max}^{\text{opt}}$ denote the optimal makespan. We obviously have that

$$C_{\max}^{\text{opt}} \geq \frac{T_{\text{seq}}}{m} \tag{4}$$

$$C_{\max}^{\text{opt}} \geq S_{\max} \tag{5}$$

$$C_{\max}^{\text{opt}} \geq D_{\max}. \tag{6}$$

B. Related Classical Scheduling Problems

There are two classical scheduling problems that are closely related to the SWA problem that we consider in this work. The first one is the *multiprocessor scheduling problem* [9], [14], in which we have a set of n tasks that must be scheduled on m machines in such a way that the makespan is minimized. As in the SWA problem, the tasks can be executed on any machine and their processing times are not machine dependent. It is well known that the multiprocessor scheduling problem is $\mathcal{NP}$-complete [9], and that any list-scheduling algorithm is a 2-approximation algorithm [13] (a list-scheduling algorithm considers a set of tasks in a given order and assigns to a processor that becomes idle the next unassigned task in the order). Note that this result is very important, since it remains valid in an *on-line* context. Of course, for the *off-line* case, there exist α-approximation algorithms which provide significantly better performance guarantees (an α-approximation algorithm produces a solution that is guaranteed to be at most a factor α away from the optimal one). These include the largest processing time (LPT) algorithm, which is a 4/3-approximation algorithm [13], and the MULTI-FIT algorithm, which guarantees a relative performance of 1.2 [8]. In addition, there exists a polynomial time approximation scheme (PTAS) for the multiprocessor scheduling problem that is due to Hochbaum and Schmoys [15]. The main difference between the multiprocessor scheduling problem and the SWA problem is that in the former, the n tasks are assumed to be pair-wise independent, and thus, any two tasks may be scheduled simultaneously on different processors. In the SWA problem, on the other hand, there is a set of compatibility constraints among the operations, as defined by constraints C1 and C2, which prevent the simultaneous execution of certain operations.

The second related problem is the *open-shop problem*, where we have a set of n jobs with m operations each [14], [12], [17], [11], such that the lth operation, $l = 1, \ldots, m$, of a job must be processed on machine l. Similar to the SWA problem, under open-shop scheduling, the operations of a job may be processed in any order, and only one operation of a given job can be processed at any given time. The preemptive version of the open-shop problem is solvable in polynomial time [12]. A polynomial-time algorithm also exists for the nonpreemptive open-shop problem with $m = 2$ processors [12]. However, the general nonpreemptive open-shop problem is known to be $\mathcal{NP}$-complete, and it has been shown that any list-scheduling algorithm is a 2-approximation algorithm [16]. Here also, it is interesting to point out that this result holds in the on-line context.

C. Applications

We now describe several network environments where the SWA problem arises.

Broadcast WDM optical networks [1]: Consider a WDM optical network with n nodes interconnected via a broadcast star that supports $m < n$ distinct wavelengths [19]. Nodes communicate by exchanging fixed-length packets, and the time it takes to transmit a packet is taken as the unit of time. Since there are fewer wavelengths than nodes, packet transmissions by several nodes may share a single wavelength, and the problem of scheduling these packet transmissions arises. At the same time, an important objective in such a network is load balancing across the different wavelengths, since it has been shown that network performance deteriorates significantly if the traffic load concentrates on a few wavelengths [22], [21], [2], [3].

Let us assume that the long-term traffic requirements of the nodes are known, let O_{sd} denote the operation of transmitting packets from node s to node d, and let p_{sd} be the number of packets that need to be transmitted between these two nodes in the network. Let us also assume that the nodes are equipped with fast-tunable transmitters, so that no cost is incurred when a transmitter switches from one wavelength to another, but that receivers are fixed tuned to a certain wavelength. (These are tunability characteristics of nodes in the Helios DARPA NGI project [1].) Let us also consider the operations of transmitting packets to a single receiver d as a job J_d. Then, scheduling the packet transmissions over the m wavelengths is equivalent to the *preemptive* SWA problem, since the transmission of the p_{sd} packets of operation O_{sd} can be preempted (at the end of any packet) and continued at a later time. Note that minimizing the makespan for this problem implies balancing the traffic across the various wavelengths by properly assigning the fixed-tuned receivers to wavelengths.

WDM IP routers employing multiprotocol lambda switching: Consider a backbone network of high-speed IP routers interconnected by fiber lines. The routers communicate using generalized multiprotocol label switching (GMPLS) [7], an extension of MPLS [6] also referred to as multiprotocol lambda switching (MPλS). Specifically, each router originates a number of lightpaths to other routers in the backbone network, and it forwards all packets it receives onto one of its outgoing lightpaths. To avoid packet reordering at the destination router, packets for a given destination d (i.e., those carrying the same MPLS label) must be sent over the same lightpath in a first-come–first-served order.

Now, consider an edge backbone router acting as the ingress node for n IP routers accessing the MPλS network. Let us assume that this ingress router originates m lightpaths to other backbone routers. The problem that arises is to schedule packets received by the n access routers for transmission on one of the m lightpaths such that packets with the same destination are sent on the same lightpath and the overall traffic from the n access routers is balanced across the m lightpaths. It can be seen that this is an instance of the SWA problem defined earlier.

Grooming packet traffic over WDM SONET/SDH rings: In this scenario, we consider a WDM ring network that carries either ATM fixed-size cells or IP variable-size packets over multiple wavelengths, with each wavelength employing SONET/SDH. We assume that each ring node is equipped with a switch which allows it to switch traffic from any of the wavelengths terminating at the node to any wavelength originating from the same node. In order to increase the utilization of wavelength and decrease the network equipment cost, each ring node must appropriately *groom* lower-rate streams into high-rate wavelengths [5].

Consider a ring node that originates m wavelengths. The ring receives traffic 1) from other ring nodes (arriving over wavelengths terminating at the node), or 2) from nonring access devices (i.e., ATM switches or IP routers) attached to the node. All traffic from this node to the other nodes in the ring must be carried by these m wavelengths. Furthermore, for the grooming to be effective, all traffic for a certain destination must be aggregated onto the same wavelength. Because of the synchronous nature of SONET/SDH, lower-rate traffic streams cannot be preempted, and a *nonpreemptive* SWA problem arises in this case.

While all the applications described in this section are in packet scheduling in WDM networks, for the rest of the paper we will use terminology from the multiprocessor scheduling literature. The reader should keep in mind, however, that the terms "processor," "processing time," and "job" correspond to "wavelength," "transmission time," and "destination," respectively, in the network environment.

III. Preemptive Scheduling

Let us first assume that the operations O_{sd} of any job J_d are preemptable, i.e., there is no cost in preempting an operation and resuming it later on the *same* processor (refer to constraint C1). We have the following result.

Lemma 1: The preemptive SWA problem is $\mathcal{NP}$-complete.

Proof: Consider an instance of the SWA problem with $m = 2$ processors and n jobs. Each job $J_d, d = 1, \ldots, n,$ of the instance consists of a single operation O_{dd} with nonzero processing time p_{dd} (i.e., all other operations $O_{sd}, s \neq d, s = 1, \ldots, n,$ of job J_d are such that $p_{sd} = 0$). Since an operation may not change processor after a preemption, this special case of the preemptive SWA is equivalent to the *PARTITION* problem [9], which is $\mathcal{NP}$-complete. ■

An approximation algorithm for the preemptive SWA problem may now be obtained by considering the problem decomposition described previously. The approximation algorithm consists of two steps. The first step assigns jobs to processors in a way that attempts to balance the amount of work (i.e., the total processing time) assigned to each processor. The second step is to apply existing approximation algorithms to the resulting open-shop problem.

Algorithm DA (Decomposition Algorithm):

Step 1) Let P_d be the total processing time required by job $J_d, P_d = \sum_{s=1}^{n} p_{sd}$. Considering each job J_d as an independent task with processing time equal to P_d, run an approximation algorithm for the resulting multiprocessor problem to assign each job J_d to a processor.

Step 2) Let $\mathcal{J}_x$ denote the set of jobs assigned to processor x at the end of Step 1). For each processor x, create new operations O'_{sx} with processing time $p'_{sx} = \sum_{d \mid J_d \in \mathcal{J}_x} p_{sd}$. Now, the original SWA problem has been transformed to a preemptive open-shop scheduling problem with m processors and operations O'_{sx}. Run the Gonzalez and Sahni algorithm [12] to obtain an optimal schedule for the new open-shop problem.

Lemma 2: Algorithm DA is an approximation algorithm for the preemptive SWA problem.

Proof: Correctness. Step 1) assigns each job to a certain processor, thus, all operations of a job will be executed on the same processor, satisfying constraint C1. In Step 2), a preemptive open-shop problem is constructed, such that the processing time of an operation O_{sx} for a source s on processor x is the sum of the processing times of operations O_{sd} for which job J_d has been assigned to processor x. Consider two jobs J_d and $J_{d'}$, $d \neq d'$. If the jobs have been assigned to two different processors, the Gonzalez and Sahni algorithm ensures that constraint C2 is satisfied. If the jobs have been assigned to the same processor, constraint C2 is also satisfied since, whenever operation O_{sx} is being processed, at most one of operations O_{sd} or $O_{sd'}$ (which are part of O_{sx}) is processed, but not both. Finally, the operation of the Gonzalez and Sahni algorithm does not violate constraint C3.

Approximation Claim. Consider the preemptive open-shop problem in Step 2), and let $D'_{\max}$ (respectively, $S'_{\max}$) denote the maximum processing time associated with any processor (respectively, source). By construction of the open-shop problem we have that

$$S'_{\max} = S_{\max} \tag{7}$$

where the quantity $S_{\max}$ for the original SWA problem is defined in (3). Note, now, that an α-approximation algorithm is used in Step 1) for the multiprocessor scheduling problem. This approximation algorithm respects constraint C1 only, therefore the optimal for the multiprocessor scheduling problem is at most equal to the optimal for the SWA problem. Thus

$$D'_{\max} \leq \alpha C^{\text{opt}}_{\max} \tag{8}$$

where $C^{\text{opt}}_{\max}$ refers to the SWA problem. Since the Gonzalez and Sahni algorithm for the preemptive open-shop scheduling problem is optimal [12], the schedule produced by the decomposition algorithm has length

$$C^{\text{DA}}_{\max} = \max\{D'_{\max}, S'_{\max}\} \leq \alpha C^{\text{opt}}_{\max} \tag{9}$$

where the last inequality follows from (5). ■

The above lemma implies that, if we use LPT (respectively, MULTI-FIT) in Step 1), then the decomposition algorithm is a 4/3-approximation (respectively, 1.2-approximation) algorithm. On the other hand, the decomposition algorithm becomes a PTAS if the PTAS developed by Hochbaum and Schmoys for the multiprocessor scheduling problem is used in Step 1) to obtain the assignment of jobs to processors.

IV. Nonpreemptive Scheduling

Let us now consider the nonpreemptive SWA problem, whereby, once an operation O_{sd} has started processing on a certain processor x, it must be completed before the processor can start executing another operation. The following lemma proves that the nonpreemptive version of the problem is also $\mathcal{NP}$-complete.

Lemma 3: The nonpreemptive SWA problem is $\mathcal{NP}$-complete.

Proof: Consider an instance of the nonpreemptive SWA problem with m processors and n jobs. Each job $J_d, d = 1, \ldots, n$, of the instance consists of a single operation O_{dd} with nonzero processing time p_{dd} (i.e., all other operations $O_{sd}, s \neq d, s = 1, \ldots, n$, of job J_d are such that $p_{sd} = 0$). This instance consists of n independent tasks (the n operations O_{dd}) and $m < n$ processors. Thus, the well-known $\mathcal{NP}$-complete multiprocessor scheduling problem [9], [14] is a special case of the nonpreemptive SWA problem. ■

A. Algorithm Based on Problem Decomposition

An algorithm based on a problem decomposition, similar to the one developed for the preemptive case in Section III, can also be applied to this problem. However, nonpreemptive open-shop scheduling is an $\mathcal{NP}$-complete problem [12], but it is known that any list-scheduling algorithm is a 2-approximation algorithm for this problem [16]. Therefore, instead of the Gonzalez and Sahni optimal algorithm for preemptive open-shop scheduling, a 2-approximation list-scheduling algorithm for the nonpreemptive open-shop scheduling problem is applied in Step 2) of the decomposition algorithm DA in Section III. We now have the following lemma.

Lemma 4: If an α-approximation algorithm is applied to the corresponding multiprocessor scheduling problem in Step 1 of the decomposition, then the decomposition algorithm DA is a 2α-approximation algorithm for the nonpreemptive SWA problem.

Proof: Correctness. The proof of correctness is similar to the one given for the preemptive SWA problem in Lemma 2.

Approximation Claim. If an α-approximation algorithm is used in Step 1) for the multiprocessor scheduling problem, by construction of the open-shop problem we have that (refer also to the proof of Lemma 2)

$$D'_{\max} \leq \alpha C^{\text{opt}}_{\max} \tag{10}$$

$$S'_{\max} = S_{\max} \tag{11}$$

where the quantity $S_{\max}$ for the original SWA problem is defined in (3). Since list scheduling is a 2-approximation algorithm for the nonpreemptive open-shop scheduling problem [12], the schedule produced by the decomposition algorithm has length

$$C^{\text{DA}}_{\max} \leq 2\max\{D'_{\max}, S'_{\max}\} \leq 2\alpha C^{\text{opt}}_{\max} \tag{12}$$

where the last inequality follows from (5). ■

We now note that, while the decomposition algorithm DA will correctly schedule the operations of the nonpreemptive SWA problem (i.e., the resulting schedule will satisfy constraints C1, C2, and C3), it may actually do more than is needed to satisfy the nonpreemption requirement. Consider jobs J_d and $J_{d'}, d \neq d'$, if they exist, that have been assigned to the same processor x. In Step 2) of the algorithm, a new operation $O_{sx}, s = 1, \ldots, n$, is created for the new open-shop problem, with a processing time equal to the sum of the processing times of operations O_{sd} and $O_{sd'}$ (and, possibly, the operations of other jobs assigned to the same processor) of the original SWA problem. The list-scheduling algorithm in Step 2) applied to the open-shop scheduling problem will ensure that the operation O_{sx} will not be preempted, while the requirement for the original SWA problem is simply that operations O_{sd} and $O_{sd'}$ not be preempted.

B. Algorithms Based on List Scheduling

We now present list-scheduling algorithms for the nonpreemptive SWA problem. We first show that a naive implementation of list scheduling can produce schedules that are a factor of m from the optimal schedule. We then describe a more refined version of list schedule that yields a 2-approximation algorithm for the nonpreemptive SWA problem.

We will need the following definitions in our discussion. A job J_d is said to be *assigned* to processor x at time t if

1) no operation $O_{sd}, s = 1, \ldots, n$, has been executed (fully or partially) on any processor before time t, and
2) processor x is idle at time t and starts processing some operation $O_{sd}, s = 1, \ldots, n$, at that time.

Because of the problem constraints, once a job J_d is assigned to processor x, all operations $O_{sd}, s = 1, \ldots, n$ must be executed on x. An operation O_{sd} is called *schedulable* on processor x at time t if

1) job J_d has not been assigned to a different processor y at some time $t' < t$, and
2) an incompatible operation $O_{s'd'}$ (i.e., such that $s = s'$ or $d = d'$) is not being executed by any processor y at time t.

$$
\begin{array}{c|cccccc|cccc|}
 & J_1 & J_2 & J_3 & \cdots & J_m & & J_{m+1} & J_{m+2} & \cdots & J_{2m} \\
1 & M-\frac{m+1}{m} & 0 & 0 & \cdots & 0 & & 0 & 0 & \cdots & 0 \\
2 & \frac{1}{m} & M-\frac{m+1}{m} & 0 & \cdots & 0 & & 0 & 0 & \cdots & 0 \\
3 & 0 & \frac{1}{m} & M-\frac{m+1}{m} & \cdots & 0 & & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\
m & 0 & 0 & 0 & \cdots & M-\frac{m+1}{m} & & 0 & 0 & \cdots & 0 \\
m+1 & 0 & 0 & 0 & \cdots & \frac{1}{m} & & \frac{1}{m} & \frac{1}{m} & \cdots & \frac{1}{m} \\
m+2 & 0 & 0 & 0 & \cdots & 0 & & \frac{1}{m} & \frac{1}{m} & \cdots & \frac{1}{m} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\
2m & 0 & 0 & 0 & \cdots & 0 & & \frac{1}{m} & \frac{1}{m} & \cdots & \frac{1}{m}
\end{array}
$$

Fig. 1. Problem instance that asymptotically achieves the upper bound of Lemma 5.

When a processor x becomes idle at some time t, it can only start processing an operation that is schedulable at time t.

1) List Scheduling, Version 1:

Algorithm LS1 (List Scheduling, Version 1): All the tasks $O_{sd}, s, d = 1, \ldots, n$, are initially arranged in an arbitrary list L. Consider a processor x which becomes idle at time t. If list L is empty, the algorithm terminates. Otherwise, processor x starts processing the first schedulable operation O_{sd} in L, and the operation is removed from the list. If no schedulable operation is found in L, processor x remains idle until time $t' > t$ at which another processor y that was busy at time t completes its operation. At time t', processors x and y each scan list L to select a new schedulable operation to process. (Ties are broken arbitrarily).

Lemma 5: Algorithm LS1 is an m-approximation algorithm for the nonpreemptive SWA problem.

Proof: Correctness. By construction, the algorithm ensures that a processor may execute at most one operation at any time instant, thus satisfying constraint C3. The definition of a "schedulable" operation above, and the requirement that a processor, upon becoming idle, selects a schedulable operation for execution, also ensure that the algorithm will never violate constraints C1 or C2.

Approximation Claim: Let $C_{\max}^{\mathrm{LS1}}$ denote the makespan of a schedule produced by algorithm LS1, and let T_{idle} denote the total idle time (over all m processors) in this schedule. We first observe that at no point in time can all m processors be idle in a schedule produced by using algorithm LS1, thus, $T_{\mathrm{idle}} \leq (m-1)C_{\max}^{\mathrm{LS1}}$. Because of (4), we obtain

$$C_{\max}^{\mathrm{LS1}} \leq \frac{T_{\mathrm{idle}} + T_{\mathrm{seq}}}{m} \leq \frac{m-1}{m} C_{\max}^{\mathrm{LS1}} + C_{\max}^{\mathrm{opt}} \Rightarrow C_{\max}^{\mathrm{LS1}} \leq m C_{\max}^{\mathrm{opt}}. \quad (13)$$

■

The following problem instance shows that, asymptotically, the bound of the lemma is a *tight one*. The instance consists of a number m of processors and a number $2m$ of jobs whose operations are shown in Fig. 1. The first m jobs consist of two operations, one long operation of length $M-(m+1)/(m)$ and one short operation of length $(1/m)$. The last m jobs consist of exactly m short operations each, of length $(1/m)$. Let $M = m(m-1)$. The optimal schedule, shown in Fig. 2(a), is such that processor $x, x = 1, \ldots, m$, executes exactly two jobs, say, jobs x and $x+m$. This schedule has length $C_{\max}^{\mathrm{opt}} = M = m(m-1)$. On the other hand, it is easy to see that under algorithm LS1, it is possible for one processor, say, processor 1, to be assigned jobs 1 through $m+1$, while processor $x, x = 2, \ldots, m$, is assigned only one of the last $m-1$ jobs. Such a schedule is illustrated in Fig. 2(b), and has length

$$C_{\max}^{\mathrm{LS1}} = 2 + m\left(M - \frac{m+1}{m}\right) = mM - (m-1) = m^2(m-1) - (m-1). \quad (14)$$

Then

$$\frac{C_{\max}^{\mathrm{LS1}}}{C_{\max}^{\mathrm{opt}}} = \frac{m^2(m-1)-(m-1)}{m(m-1)} = m - \frac{1}{m}. \quad (15)$$

2) List Scheduling, Version 2: We now present a different version of list scheduling which yields a 2-approximation algorithm for the nonpreemptive problem.

Algorithm LS2 (List Scheduling, Version 2): The n jobs $J_d, d = 1, \ldots, n$, are initially arranged in a list L. The operations of each job J_d are also arranged in a list L_d. A list l_x is also associated with each processor x, and it is initialized to the empty list. Consider a processor x that becomes idle at time t, and let O_{sd} be the operation that was just completed by the processor. Processor x selects an operation to process next by taking the following three steps in the order presented.

1) If there exists an operation $O_{sd'}, d' \neq d$, in the processor's list l_x, it is removed from the list and x starts processing this operation at time t (note that operation $O_{sd'}$ is schedulable on x at time t, since scheduling it does not violate constraint C2).
2) If no operation $O_{sd'}$ is found in Step 1), x starts processing any other schedulable operation in its list l_x, and the operation is removed from the list.
3) If no operation is found in the first two steps, list L is scanned. If L is empty, the algorithm terminates. Otherwise, let $J_{d'}$ be the first job in L with an operation that is schedulable on processor x at time t, if one exists. Job

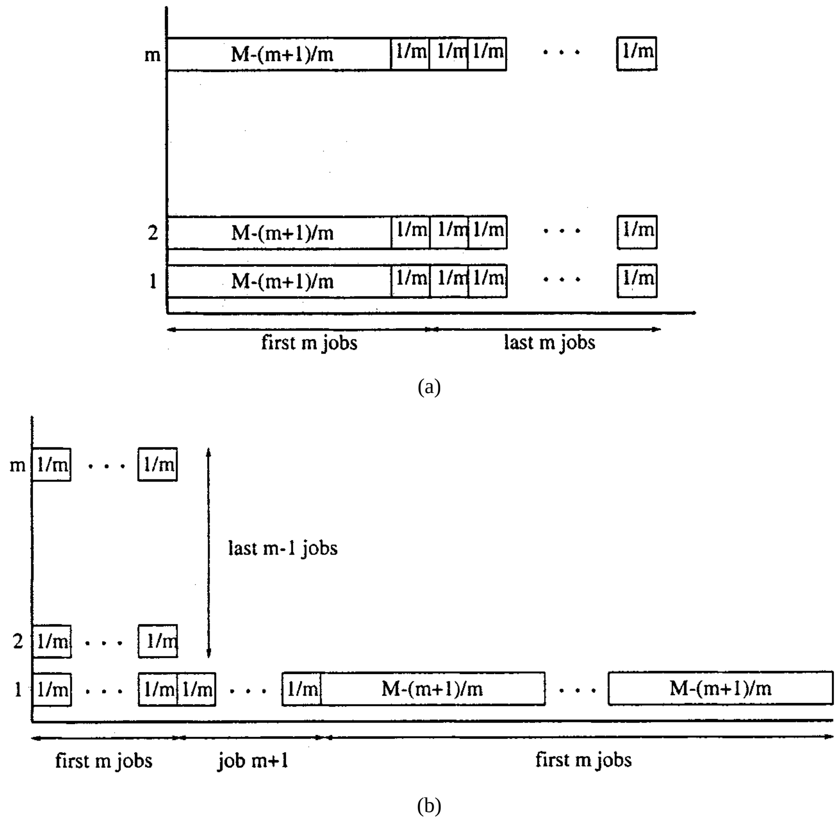

Fig. 2. (a) Optimal schedule and (b) worst-case schedule produced by algorithm LS1 for the instance shown in Fig. 1.

$J_{d'}$ is removed from L, and its list of operations $L_{d'}$ is appended to processor x's list l_x. Processor x starts processing the first schedulable operation in its new list l_x, and the operation is removed from l_x.

If no schedulable operation is found at the end of the third step, processor x remains idle until time $t' > t$ at which another processor y that was busy at time t completes its operation. At time t', processors x and y each repeat the above procedure to select a new schedulable operation to process. Ties are resolved using the following rule.

- Consider two processors x and y which start the process of selecting a new operation at the same time t. Let us assume that processor x finds an operation in Step 1) while processor y finds an operation at Step 2) or 3), but the two operations cannot be executed simultaneously. Then, processor x is allowed to proceed while processor y must either find another compatible operation or remain idle until a future time t'. Otherwise, ties are broken arbitrarily.

Lemma 6: Algorithm LS2 is a 2-approximation algorithm for instances of the nonpreemptive SWA problem for which no operation has zero processing time (i.e., $p_{sd} > 0$, $\forall s, d$).

Proof: Correctness. By construction, the algorithm satisfies constraint C3. The fact that when the first operation of a job J_d is schedulable on a processor x, the job is removed from list L and is appended to processor x's list, ensures that constraint C1 is not violated. Finally, the requirement that a processor, upon becoming idle, selects a schedulable operation for execution, guarantees that the resulting schedule will satisfy constraint C2.

Approximation Claim. Consider a schedule $\mathcal{S}$ built using algorithm LS2, and let J_d be the job that is the last to be assigned to a processor (i.e., all other jobs have been assigned to processors before J_d). Let t be the time when job J_d is assigned to a processor. We claim that no processor is idle before time t. To see that the claim is true, assume that a processor y became idle at time $t' < t$. Note now that at most $m-1$ different operations were being processed at time t', one at each of the other processors. By assumption, J_d includes $n > m-1$ nonzero operations, and $n-m+1$ of these operations are schedulable on y at time t'. Thus, under algorithm LS2 and because of the way ties are resolved, job J_d should have been assigned to y at that point, contradiction.

At time t, all jobs have been assigned to processors. Let $\mathcal{P} = \{\mathcal{J}_1, \ldots, \mathcal{J}_m\}$ be the partition of the job set $\mathcal{J}$ such that $\mathcal{J}_x$ is the subset of jobs that have been assigned by the algorithm to processor $x, x = 1, \ldots, m$. Let $p_{sd}(t)$ be the amount of processing that operation O_{sd} has received up to time t. Consider a new scheduling problem with m processors and n jobs J'_s, where each job is a set of operations, $J'_s = \{O'_{s1}, \ldots, O'_{sm}\}$, and the processing time of operation O'_{sx} is given by

$$p'_{sx} = \sum_{d \in \mathcal{J}_x} (p_{sd} - p_{sd}(t)). \tag{16}$$

This new problem is an instance of open-shop scheduling [12] with m processors and n jobs J'_s, where each job has a single operation O'_{sx} to be executed on each processor x. We now observe that algorithm LS2 is also a list-scheduling algorithm for this open-shop problem. To see this, consider a processor x that becomes idle at time $t' > t$. Since the last job was assigned to some processor at time t, no new jobs will be assigned to x at or after time t'. Let O_{sd} be the operation that was just completed by x at time t'. Because of the first step in selecting a new job under algorithm LS2, if there is another job $O_{sd'}$ in list l_x, x will start processing $O_{sd'}$ at time t'. When x becomes idle again, the same selection process will be repeated until all operations $O_{sd'}$ with the same source s in list l_x are completed (note that each of these operations are schedulable on x at the instant the previous one is completed). Thus, these operations $O_{sd'}$ will be executed back-to-back, in some order, without interruption, and no operations with the same source will be added to list l_x after time t'. But all these operations $O_{sd'}$ are part of the operation O'_{sx} of job J'_s in the open shop problem. Consequently, operation O'_{sx} will be executed without preemption on processor x by algorithm LS2. Therefore, algorithm LS2 is a list-scheduling algorithm for the open-shop problem.

Let $D'_{\max}$ (respectively, $S'_{\max}$) denote the maximum processing time associated with any processor (resp., source) in the open-shop problem. Since there is no idle time in the schedule constructed by algorithm LS2 until time t, we have that

$$C_{\max}^{\text{opt}}\,(\text{SWA}) \geq t + \max\{D'_{\max}, S'_{\max}\}. \tag{17}$$

(Note that we give the problem to which the various parameters are related in parentheses: OS for open-shop, and SWA for the original SWA problem.) Since LS2 is a list-scheduling algorithm for the open-shop problem, and any list-scheduling algorithm is a 2-approximation algorithm for the nonpreemptive open shop, we also have that

$$C_{\max}^{\text{LS2}}\,(\text{OS}) \leq 2\max\{D'_{\max}, S'_{\max}\} \leq 2C_{\max}^{\text{opt}}\,(\text{OS}). \tag{18}$$

Finally, we get the desired result as follows:

$$\begin{aligned} C_{\max}^{\text{LS2}}\,(\text{SWA}) &= t + C_{\max}^{\text{LS2}}\,(\text{OS}) \\ &\leq t + 2\max\{D'_{\max}, S'_{\max}\} \\ &\leq t + \max\{D'_{\max}, S'_{\max}\} + C_{\max}^{\text{opt}}\,(\text{SWA}) \\ &\leq 2C_{\max}^{\text{opt}}\,(\text{SWA}). \end{aligned} \tag{19}$$

■

Now, assume that some of the operations in the original nonpreemptive SWA problem are zero. Without loss of generality, assume that each job J_d consists of at least one operation of nonzero length (otherwise, we can remove this job reducing the problem into an equivalent one with m processors and $n-1$ jobs). We can obtain a schedule that is at most twice the optimum one for this problem by taking the following three steps. First, replace all zero-length operations with ones of length equal to $\epsilon > 0$. Then, run algorithm LS2 on this new instance of the SWA problem for which no operation has zero processing time. Finally, remove from the schedule obtained by algorithm LS2 all operations that are of zero processing time in the original problem.

To see that this schedule is at most twice the optimum for the original SWA problem (the one with some operation of zero length), we observe that at most $n(n-1)$ operations are added to the problem instance in the first step. Now, note that a schedule for the new problem can be obtained by sequentially processing the $n(n-1)$ operations on one processor after the end of the optimal schedule for the original problem, thus

$$C_{\max}^{\text{opt}}\,(\text{new}) \leq C_{\max}^{\text{opt}}\,(\text{original}) + n(n-1)\epsilon. \tag{20}$$

The optimum schedule for the new problem is at most equal to this schedule. Since at the third step of the algorithm we remove some operations from the schedule produced by LS2, we have that

$$\begin{aligned} C_{\max}^{\text{LS2}}\,(\text{original}) &\leq C_{\max}^{\text{LS2}}\,(\text{new}) \\ &\leq 2C_{\max}^{\text{opt}}\,(\text{new}) \\ &\leq 2C_{\max}^{\text{opt}}\,(\text{original}) + 2n(n-1)\epsilon. \end{aligned} \tag{21}$$

By selecting an appropriate value for ϵ, we see that the schedule obtained in this way for the original problem is at most twice the optimal schedule.

Finally, we note that algorithm LS2 is valid in an on-line context, where the jobs are not known in advance but appear one after the other.

V. Concluding Remarks

We considered the scheduling and wavelength assignment (SWA) problem which has applications in packet-switched optical WDM networks. We proved that the SWA problem is $\mathcal{NP}$-complete for both the preemptive and the nonpreemptive cases. Furthermore, we proposed two efficient approximation algorithms. For the preemptive case, we described an efficient approximation algorithm based on a natural decomposition of the problem. For the nonpreemptive case, we presented two list-scheduling algorithms, the second of which is a 2-approximation algorithm for both the off-line and the on-line contexts.

References

[1] The NGI Helios project. Defense Advanced Research Projects Agency (DARPA). [Online]. Available: http://www.anr.mcnc.org/projects/Helios/Helios.html

[2] I. Baldine and G. N. Rouskas, "Dynamic load balancing in broadcast WDM networks with tuning latencies," in *Proc. IEEE INFOCOM'98*, Mar. 1998, pp. 78–85.

[3] ——, "Traffic adaptive WDM networks: A study of reconfiguration issues," *J. Lightwave Technol.*, vol. 19, pp. 433–455, Apr. 2001.

[4] "Special issue on optical networks," *J. Lightwave Technol.*, vol. 18, Dec. 2000.

[5] R. Dutta and G. N. Rouskas, "On optimal traffic grooming in WDM rings," in *Proc. ACM SIGMETRICS/PERFORMANCE 2001*, June 2001, pp. 164–174.

[6] "Special issue on multiprotocol label switching," *IEEE Commun. Mag.*, vol. 37, no. 12, Dec. 1999.

[7] P. Ashwood-Smith *et al.*, "Generalized MPLS—Signaling functional description," IETF Draft, <draft-ietf-mpls-generalized-signaling-06.txt>, Apr. 2001.

[8] D. K. Friesen, "Tighter bounds for the multifit processor scheduling algorithm," *SIAM J. Computation*, vol. 13, no. 1, pp. 170–181, 1984.

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability*. New York: Freeman, 1979.

[10] "Special issue on protocols and architectures for next-generation optical WDM networks," *IEEE J. Select. Areas Commun.*, vol. 18, Oct. 2000.

[11] T. Gonzalez, "A note on open-shop preemptive schedules," *IEEE Trans. Comput.*, vol. C-28, pp. 782–786, 1979.

[12] T. Gonzalez and S. Sahni, "Open-shop scheduling to minimize finish time," *J. ACM*, vol. 23, no. 4, pp. 665–679, Oct. 1976.

[13] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM J. Appl. Math.*, vol. 17, no. 2, pp. 416–429, Mar. 1969.

[14] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey," *Annals Discrete Math.*, vol. 5, pp. 287–326, Jan. 1979.

[15] D. S. Hochbaum and D. B. Schmoys, "Using dual approximation algorithms for scheduling problems," *J. ACM*, vol. 34, no. 1, pp. 144–162, Jan. 1987.

[16] T. Fiala and I. Bárány, "Tobbgepes, utemezesi problemak kozel optimalis megoldasa," *Szigma-Mat.-Kozgazdasagi Folyoirat*, vol. 15, pp. 177–191, 1982.

[17] E. L. Lawler and J. Labetoulle, "On preemptive scheduling of unrelated parallel processors by linear programming," *J. ACM*, vol. 25, pp. 612–619, 1978.

[18] E. Modiano, "WDM-based packet networks networks," *IEEE Commun. Mag.*, vol. 37, pp. 130–135, Mar. 1999.

[19] R. Ramaswami and K. N. Sivarajan, *Optical Networks*. San Francisco, CA: Morgan Kaufmann, 1998.

[20] G. N. Rouskas and V. Sivaraman, "Packet scheduling in broadcast WDM networks with arbitrary transceiver tuning latencies," *IEEE/ACM Trans. Networking*, vol. 5, pp. 359–370, June 1997.

[21] V. Sivaraman and G. N. Rouskas, "HiPeR-ℓ: A High Performance Reservation protocol with ℓook-ahead for broadcast WDM networks," in *Proc. IEEE INFOCOM'97*, Apr. 1997, pp. 1272–1279.

[22] ——, "A reservation protocol for broadcast WDM networks and stability analysis," *Comput. Networks*, vol. 32, no. 2, pp. 211–227, Feb. 2000.

Evripidis Bampis received the Diploma in electrical engineering from the National Technical University of Athens (NTUA), Athens, Greece, in 1989 and the M.Sc. and Ph.D. degrees in computer science from the University of Paris-Sud (Orsay), France, in 1990 and 1993, respectively.

He joined the Department of Computer Science, University of Evry Val d'Essonne, France, in September 1993, as an Assistant Professor. In January 1998, he received the Habilitation in computer science from the University of Evry, and since September 1998, he has been a Professor and Director of the team "Algorithms, Discrete Optimization and Applications" of Laboratoire de Méthodes Informatiques (LaMI) at the University of Evry. His research interests are in designing efficient algorithms and studying the (non)approximabilty for various problems arising in modern computer and telecommunication systems, such as scheduling, graph theory, and parallel algorithms. He is the coordinator of a national program of the French Ministry of Education and Research on *Approximability and Local Search*. He participates in a Thematic Network of the European Union on *Approximation and On-line algorithms for Optimization problems* (APPOL-IST-2001-32007), and he is the French Coordinator of the French–German research exhange program PROCOPE and a France–Berkeley project on multiobjective optimization. He was a Co-Chair of the Workshop WEA'2001 on Efficient Algorithms (in conjunction with FCT'2001), Riga, Latvia. He served as co-guest editor for a special issue of *Parallel Computing* on task scheduling problems for parallel and distributed systems.

George N. Rouskas (S'92–M'95–SM'01) received the Diploma in electrical engineering from the National Technical University of Athens (NTUA), Athens, Greece, in 1989 and the M.S. and Ph.D. degrees in computer science from the College of Computing, Georgia Institute of Technology, Atlanta, in 1991 and 1994, respectively.

He joined the Department of Computer Science, North Carolina State University, Raleigh, in August 1994, where he has been an Associate Professor since July 1999. During the 2000–2001 academic year, he spent a sabbatical term at Vitesse Semiconductor, Morrisville, NC, and in May–June 2000, he was an Invited Professor at the University of Evry, France. His research interests include network architectures and protocols, optical networks, multicast communication, and performance evaluation.

Dr. Rouskas is a recipient of a 1997 NSF Faculty Early Career Development (CAREER) Award, and a coauthor of a paper that received the Best Paper Award at the 1998 SPIE conference on All-Optical Networking. He also received the 1995 Outstanding New Teacher Award from the Department of Computer Science, North Carolina State University, and the 1994 Graduate Research Assistant Award from the College of Computing, Georgia Institute of Technology. He was a Co-Guest Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, Special Issue on Protocols and Architectures for Next Generation Optical WDM Networks, published in October 2000, and is on the editorial boards of the IEEE/ACM TRANSACTIONS ON NETWORKING and the *Optical Networks Magazine*. He is a member of the Association for Computing Machinery and the Technical Chamber of Greece.