Near-Optimal Algorithm for Group Scheduling in OBS Networks

Vo Viet Minh Nhat, Nguyen Hong Quoc, and Nguyen Hoang Son

Group scheduling is an operation whereby control packets arriving in a time slot schedule their bursts simultaneously. Normally, those bursts that are of the same wavelength are scheduled on the same channel. In cases where the support of full wavelength converters is available, such scheduling can be performed on multiple channels for those bursts that are of an arbitrary wavelength. This paper presents a new algorithm for group scheduling on multiple channels. In our approach, to reach a near-optimal schedule, a maximum-weight clique needs to be determined; thus, we propose an additional algorithm for this purpose. Analysis and simulation results indicate that an optimal schedule is almost attainable, while the complexity of computation and that of implementation are reduced.

Keywords: OBS networks, group scheduling, interval graph, maximum-weight clique, optimization.

I. Introduction

Optical burst switching (OBS) is considered as an appropriate alternative model for optical packet switching, where the current optical technology has not really matured enough to produce optical buffers and optical packet switching fabrics with nanosecond switching speed [1]. A typical feature of OBS networks is the burst control packet (BCP), which is a packet that is transmitted separately (in terms of both space and time) from its associated data burst. This means that a BCP is sent on a control channel that is separate to the data channel transporting its associated data burst; an offset time is created, since the BCP is sent ahead of its associated data burst. This offset time must be predetermined sufficiently so that the BCP can reserve necessary resources and configure switches just before its associated data burst arrives at each node on the path from source to destination.

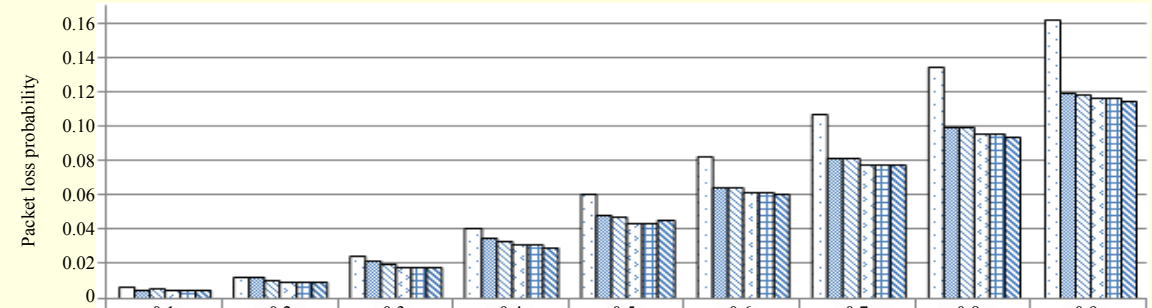
The action of resource reservation of a BCP at an OBS core node normally is a part of scheduling operation. There are two approaches for scheduling — online scheduling and group (offline) scheduling [2]–[3]. In the case of online scheduling, a BCP arriving at an OBS core node calls a scheduling algorithm (for example, LAUC [4] or LAUC-VF [5]) to reserve immediately the resources for its following burst. In the case of group scheduling, the BCPs arriving in a timeslot (or a time window) schedule their bursts simultaneously via one of the following algorithms: OBS-GS [4], MWIS-OS [5], LGS [6], heuristics [7], GreedyOPT [8], BATCHOPT [8], or LGS-MC [9]. As proved in [7]–[9] and illustrated in Fig. 1, group scheduling is more effective than online scheduling; however, group scheduling requires more updating operations. Therefore, its complexity is higher than that of online scheduling.

The first three of the aforementioned algorithms, OBS-GS,

Manuscript received Jan. 31, 2015; revised Aug. 12, 2015; accepted Aug. 21, 2015.

Vo Viet Minh Nhat (corresponding author, vvmnhat@hueuni.edu.vn) and Nguyen Hong Quoc (nhquoc@hueuni.edu.vn) are with the Department of Computer Science, Hue University, Vietnam.

Nguyen Hoang Son (nhson_dhkh@hueuni.edu.vn) is with the Department of Mathematics, Hue University, Vietnam.



	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
LAUC	0.006108162	0.011859506	0.024671186	0.040624383	0.060450020	0.081926322	0.107093528	0.134104531	0.162293011
LAUC-VF	0.004717927	0.011724684	0.021761243	0.035035346	0.048227715	0.064696445	0.081628514	0.099479622	0.119260345
SLV	0.005353886	0.010056264	0.019969178	0.033314518	0.047354428	0.064577464	0.080984653	0.098871889	0.118239700
MCF	0.004481291	0.009309929	0.017871092	0.030568790	0.043704086	0.060974804	0.077789469	0.095683022	0.116115489
SSF	0.004574959	0.009349653	0.017871902	0.030691498	0.043696648	0.060963215	0.077781870	0.095709545	0.116080683
LIF	0.004313674	0.009244926	0.018081137	0.029065915	0.044850450	0.060093648	0.077632881	0.093638987	0.114764696

Fig. 1. Comparison based on packet loss probability among LAUC, LAUC-VF, and heuristics with loads from 0.1 to 0.9 Erlangs.

MWIS-OS and LGS, are for group scheduling on a single channel if wavelength conversion is unavailable on OBS core nodes and the arriving bursts in a time slot are assumed to have the same wavelength; while, the remaining algorithms perform scheduling on multiple channels with the support of full wavelength converters. In this paper, we focus on group scheduling on multiple channels.

Our main contribution is to propose a near-optimal algorithm for group scheduling on multiple channels. Our algorithm is based on finding a maximum-weight clique (MWC), for which we propose a further new approach of finding such a clique on an interval graph that represents the scheduling possibilities of arriving bursts on data channels. Analysis and simulation results indicate that a near-optimal schedule with a minimal packet loss probability is achievable, while the complexity of computation and that of implementation are reduced.

The remainder of this paper is organized as follows. Section II presents some notations of graphs and related components. Section III summarizes previous proposals on group scheduling and their limitations. Section IV presents our algorithms for group scheduling based on finding an MWC. Analysis and simulation results are presented in Section V, and conclusions are given in Section VI.

II. Notations

Let $G = (V, E)$ be a graph, where V represents the set of vertices of G and E is the set of edges. For a vertex, u ($u \in V$), a positive weight, w_u, is associated with it, and its set of adjacent vertices is defined as $\text{adj}(u) = \{v \in V \mid (u, v) \in E\}$.

A *clique* in G is a set of vertices, C ($C \subseteq V$), such that $\forall u, v \in C$, $(u, v) \in E$. A *maximal* clique is a clique that is not a subset of any other clique. A *maximum* clique is a maximal clique that has maximum cardinality or weight. This paper focuses on the weighted maximum clique problem.

A graph, G, is defined to be an *interval* graph if there exists a one-one correspondence between its vertices and a family of intervals; any two vertices in G are deemed to be adjacent if their corresponding intervals overlap. This paper considers the opposite case in which two vertices in G are deemed to be adjacent if their corresponding intervals do not overlap.

III. Related Works

The problem of group scheduling, which is related to bursts arriving in a time slot on different data channels, can be formulated as a job scheduling problem, and an interval graph can be constructed to find an optimal scheduling solution [10]. As with job scheduling, the group scheduling of bursts arriving in a time slot is associated with the problem of scheduling with non-identical machines (S-NIM), because there may exist a burst that cannot be scheduled on a certain data channel if its start time is before the latest available unscheduled time (LAUT) of the channel in question. Because of the NP-hard complexity of S-NIM [10], a possible solution is to use heuristic approaches [7] or to transform the problem from S-NIM to scheduling with identical machines (S-IM) [10].

The heuristic approaches proposed in [7] consist of Smallest Start-time First (SSF), Largest Interval First (LIF), Smallest-Last Vertex (SLV), and Maximal Cliques First (MCF). The idea expressed in [7] is to first rearrange the order of arriving bursts and then to call LAUC-VF to schedule them.

SSF arranges the arriving bursts based on their start times. This approach is simple, but it does not facilitate the search for an optimal schedule [8]. LIF, an improvement over SSF, sorts the bursts according to their weight (length) and then selects the

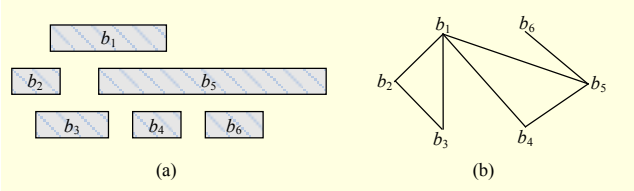


Fig. 2. Example of arriving bursts, in which burst b_1 overlaps more bursts (b_2, b_3, b_4, and b_5) than any other burst; however, it is not the longest (burst b_5 is in fact longest): (a) status of arriving bursts and (b) corresponding interval graph.

largest as the first to be scheduled. This approach may achieve a local optimal schedule on each individual channel but not a maximal weight of scheduled bursts over all channels.

In the case of SLV, an interval graph, G, is firstly constructed and any arriving bursts are then arranged in descending order of the degrees of the corresponding vertices. SLV believes that the burst corresponding to the vertex of largest degree is the longest as it overlaps more bursts than any other burst. However, this is not always true (as the example in Fig. 2 shows); so, the scheduling solution of SLV is no better than that for LIF.

Similarly, MCF firstly builds an interval graph, G, and then tries to color G by m colors (equal to the number of available data channels). To do this, MCF finds all maximal cliques, the sizes of which are greater than m. With a maximal clique having the earliest of start times, MCF respectively discards those vertices that have the earliest end times until the size of a maximal clique is equal to or smaller than m. This process is repeated until no maximal clique has a size that is greater than m. The efficiency of MCF is mainly determined by the way it discards those vertices that have the earliest end times in a maximal clique. However, such vertices do not correspond to the shortest bursts; thus, an optimal schedule, then, will not be achieved.

Different from heuristic algorithms, the approach common to GreedyOPT and BATCHOPT [8] is one that aims to transform S-NIM into S-IM. Instead of trying to allocate optimally the arriving bursts on the available bandwidths between the scheduled bursts, GreedyOPT and BATCHOPT remove all the scheduled bursts and reschedule them simultaneously with the arriving bursts.

In the case of GreedyOPT, group scheduling is based on a principle of greed, without regard for maximizing the total weight of scheduled bursts. If a burst cannot find a suitable resource on any of the available data channels, then GreedyOPT tries to replace it with the scheduled burst that has the latest end time. This action is designed to allow bursts the opportunity to seek suitable resources. However, a major drawback of GreedyOPT is that there is no mechanism to ensure that all removed bursts are rescheduled. Table 1 shows our simulation results, in which the rates of successful rescheduling of GreedyOPT never achieve 100%; noting that if a removed burst is rescheduled without a change in its placement in the schedule, then it is not considered as a rescheduling because no re-signaling is required.

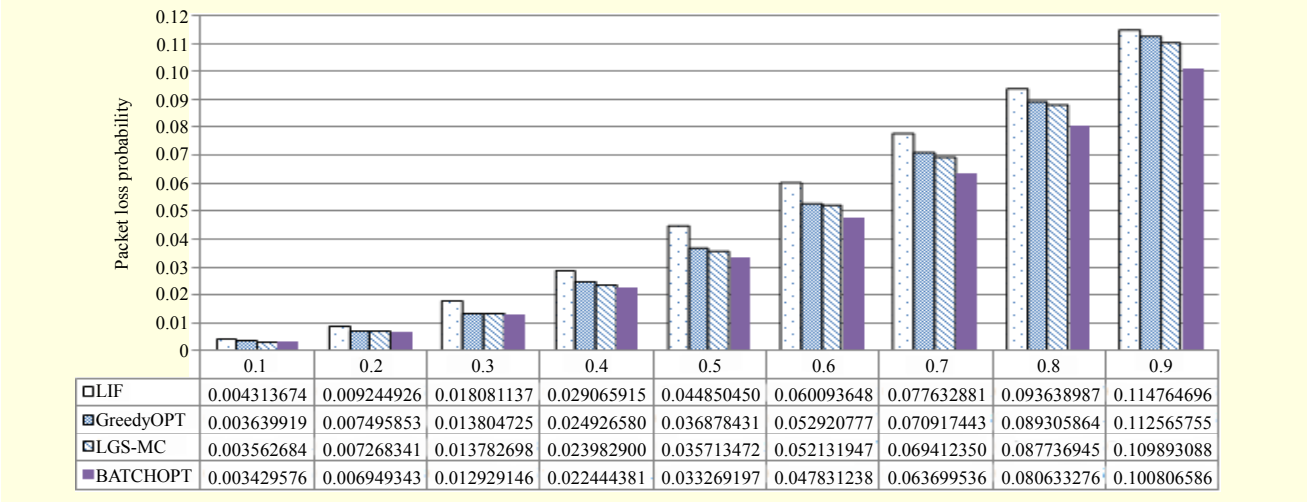
Table 1. Rates of successful rescheduling of GreedyOPT with loads from 0.1 to 0.9 Erlangs.

Load	Removed bursts	Rescheduled bursts	Rate (%) of successful rescheduling
0.1	319	276	86.52
0.2	392	327	83.42
0.3	560	481	85.89
0.4	620	537	86.61
0.5	739	646	87.42
0.6	797	692	86.83
0.7	929	803	86.44
0.8	1,072	805	75.09
0.9	1,286	904	70.30

In the case of BATCHOPT, an interval graph that represents both the arriving bursts and the scheduled bursts is firstly built. Next, the algorithm finds all maximal cliques of the graph and arranges them in ascending order of their start times. A flow graph of the arranged maximal cliques is then set in which each arc is characterized by the parameters of cost, throughput capacity, and weight (length). Finally, BATCHOPT finds the flow with minimum cost and eliminates the corresponding arcs.

Clearly, this approach helps BATCHOPT achieve an optimal solution with the remaining maximal flow. Moreover, by assigning an infinite weight value to the removed bursts, BATCHOPT ensures to reschedule all of them. However, the main drawbacks of BATCHOP as well as of GreedyOPT are as follows: system complexity is increased due to the rescheduling of previously scheduled bursts, more control packets are generated to notify the changes of scheduling status, and updates in the signaling protocol are required. The complexity of BATCHOPT is proven to be $O(N^2 \log(N))$ [8], where N is the total number of arriving and removed bursts.

Another algorithm for group scheduling, called LGS-MC [9], has a near-optimal scheduling result. With the hope of achieving an optimal schedule on all data channels, LGS-MC tries to optimize group scheduling on each channel. The scheduling optimization on each channel is performed by LGS [6]. Although the scheduling result is approximately optimal, the complexity of LGS-MC is less than that of BATCHOPT. As proved in [9], the complexity of LGS-MC is $O(m \times n\log(n))$,



	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
LIF	0.004313674	0.009244926	0.018081137	0.029065915	0.044850450	0.060093648	0.077632881	0.093638987	0.114764696
GreedyOPT	0.003639919	0.007495853	0.013804725	0.024926580	0.036878431	0.052920777	0.070917443	0.089305864	0.112565755
LGS-MC	0.003562684	0.007268341	0.013782698	0.023982900	0.035713472	0.052131947	0.069412350	0.087736945	0.109893088
BATCHOPT	0.003429576	0.006949343	0.012929146	0.022444381	0.033269197	0.047831238	0.063699536	0.080633276	0.100806586

Fig. 3. Comparison based on packet loss probability among LIF (best of heuristics), GreedyOPT, LGS-MC, and BATCHOPT with loads from 0.1 to 0.9 Erlangs.

where n is the number of arriving bursts ($n < N$) and m is the number of data channels ($m < N$). Another advantage of LGS-MC is that no rescheduling is required; thus, no complexity is added to the system.

Figure 3 shows a comparison among the group scheduling algorithms. The packets in this case refer to blocks of 64 bytes contained in each burst. The following is the detailed description of our proposed algorithms for group scheduling on multiple channels.

IV. Algorithm of MWC-GS

Consider a set of BCPs, $I = \{b_1, b_2, \ldots, b_n\}$, arriving in a time slot that schedules their respective bursts on a set of data channels, $W = \{1, 2, \ldots, m\}$. A burst is characterized by a triplet, (s_i, e_i, w_i), where s_i represents the start time, e_i the end time, and w_i the weight (length) of burst b_i, respectively. Two bursts, b_i and b_j, are scheduled on a channel, k, if they are after the LAUT of channel k ($s_i > \text{LAUT}^k$ and $s_j > \text{LAUT}^k$) and do not overlap each other ($s_i \geq e_j$ or $s_j \geq e_i$). Thus, depending on its position compared to the LAUT of data channels and its overlap with other bursts, an arriving burst can have many scheduling possibilities. For visualization, the scheduling possibilities can be modeled in the form of an interval graph.

Let $G(V, E)$ be an interval graph, where each vertex $b_i^k \in V$ presents the scheduling possibility of burst b_i on channel k and each edge $(b_i^k, b_j^h) \in E$ corresponds with one of the two following cases: (a) bursts b_i and b_j can be scheduled on the same channel ($k = h$) without being overlapped or (b) they can be scheduled on two separate channels ($k \neq h$).

With the arriving bursts in Fig. 4(a), the interval graph representing their scheduling possibilities on data channels is shown in Fig. 5(a). The problem of finding an optimal scheduling solution (for example, one with a maximal total of lengths) of the arriving bursts on data channels now becomes the problem of finding an MWC, called C_{max}, in G. The subset

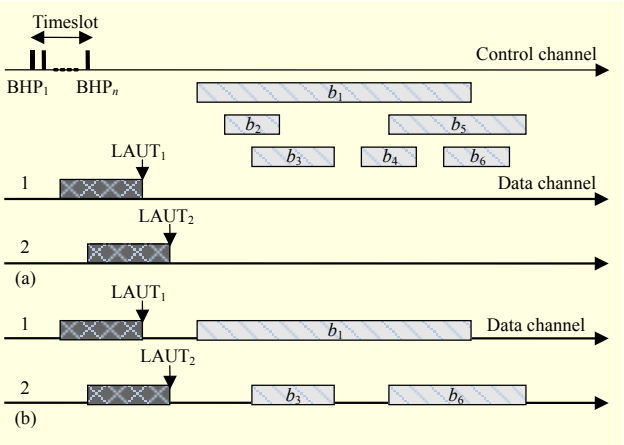


Fig. 4. Example of (a) arriving bursts and (b) optimal schedule on two data channels.

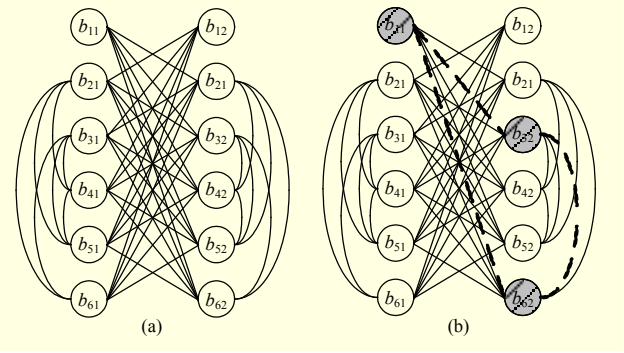


Fig. 5. (a) Interval graph representing scheduling possibilities of arriving bursts in Fig. 4(a) and (b) MWC corresponding to optimal schedule in Fig. 4(b).

of bursts $I' \subseteq I$ corresponding with the vertices in C_{max} is the optimal schedule. Figure 5(b) shows a case in which a found MWC corresponds with bursts b_1 (which is scheduled on channel 1) and b_3 and b_6 (both of which are scheduled on channel 2 (see Fig. 4(b)). Note that each vertex (b_i^k) in I' contains the full information about which burst is scheduled for which channel; the scheduling simply only distributes the bursts in I' on the channels in W in sequence.

Our algorithm for MWC-based group scheduling (MWC-GS) includes the following three main steps (functions).

MWC-GS Algorithm.
Input: A set of arriving bursts $I = \{b_1, b_2, \dots, b_n\}$.
A set of data channels $W = \{1, 2, \dots, m\}$.
Output: A set of scheduled bursts $I' \subseteq I$.
Begin
// Construct an interval graph of scheduling possibilities
1 G = constructGraph (I, W)
// Find a maximum-weight clique of G
2 C = findMWC(G).
// Schedule the bursts corresponding with the vertices in MWC
3 I' = scheduleBurstsfromMWC(C).
End

1. Interval Graph of Scheduling Possibilities

A burst $b_i \in I$ is considered to be scheduled on a channel $k \in W$ if its start time (s_i) is after the LAUT of channel k ($s_i >$ LAUT^k). The scheduling possibility is represented by a vertex on the interval graph G with a weight that is the length of burst b_i ($w_i^k = e_i - s_i$). An edge between two vertices shows the possibility in which two bursts are scheduled on two separate channels ($k \neq h$) or on the same channel ($k = h$) without being overlapped ($s_i > e_j$ or $e_i > s_j$). The function constructGraph(I, W) is described as follows.

Function constructGraph(I, W).
Input: A set of arriving bursts $I = \{b_1, b_2, \dots, b_n\}$.
A set of data channels $W = \{1, 2, \dots, m\}$.
Output: Graph $G(V, E)$.
Begin
1 For each burst b_i in I
2 For each channel k in W
3 If $s_i >$ LAUT^k
4 Create a vertex b_i^k with the weight $w_i^k = e_i - s_i$;
5 $V \leftarrow V \cup \{b_i^k\}$;
6 For each b_j^h in V and $i \neq j$
7 If ($k \neq h$) or (($k = h$) and ($s_i > e_j$ or $e_i > s_j$))
8 Create an edge (b_i^k, b_j^h);
9 $E \leftarrow E \cup \{(b_i^k, b_j^h)\}$;
10 End if;
11 End for;
12 End if;
13 End for;
14 End for;
15 Return $G(V, E)$;
End

The function constructGraph(I, W) has complexity $O(n \times m \times |V|)$ or $O(|V|^2)$.

2. MWC

The popular idea of finding an MWC is based on a comparison of the weights of all found maximal cliques; however, the problem of finding all maximal cliques is difficult. With a regular interval graph, it is possible to find an MWC by using some of the algorithms proposed in [11]; these algorithms have a complexity of only O($n$log(n)). However, the above constructed interval graph of scheduling possibilities has a particular feature; that is, each burst is represented by m vertices that correspond to m intervals with the same start/end time and the same length. Therefore, there is no criterion to choose which of the m intervals is the most suitable if the algorithms in [11] are used. This paper, therefore, proposes a new algorithm to find an MWC with the following lemma.

Lemma. Given an edge, (u, v), and a set V_{cad} = adj(u) ∩ adj(v), if a set $V' \subseteq V_{cad}$ exists to be a clique then $V' \cup \{u, v\}$ is also a clique.

Proof. According to the definition of a clique given in Section II, all the vertices in a clique are adjacent. So, if $V' \subseteq$ adj(u) ∩ adj(v) is a clique, then all vertices in V' are adjacent and also adjacent to two vertices $\{u, v\}$; therefore, $V' \cup \{u, v\}$ is also a clique. ■

Corollary. Given an edge (u, v) and a set V_{cad} = adj(u) ∩ adj(v), if a set $V' \subseteq V_{cad}$ exists to be a maximal clique, then $V' \cup \{u, v\}$ is also a maximal clique.

Our proposed algorithm comes from an arbitrary edge, (u, v), which is also the original clique $C_{(u,v)} = \{u, v\}$, and we calculate its set of adjacent vertices V_{cad} = adj(u) ∩ adj(v). First, the algorithm takes a vertex $u' \in V_{cad}$ and adds it to $C_{(u,v)}$. After that, the algorithm removes all vertices $v' \in V_{cad}$ that are not adjacent to u'. This action ensures that all remaining vertices in V_{cad} are adjacent to all of the vertices in $C_{(u,v)}$. The process is repeated until $V_{cad} = \varnothing$, which signifies that all the adjacent vertices in V_{cad} have been added to $C_{(u,v)}$. The clique $C_{(u,v)}$ is now maximal. The detailed description of the function findMWC(G) is as follows.

Function findMWC(G).
Input: a graph $G(V, E)$.
Output: the clique $C_{max} \subseteq G$.
Begin

```
1    For each u ∈ V
2      adj(u) ← {v ∈ V | (u, v) ∈ E};
3    End for;
4    For each (u, v) ∈ E
5      L(u,v) ← false;
6    End for;
7    Cmax ← ∅; Wmax ← 0;
8    For each (u, v) ∈ E and L(u,v) = false
9      C(u,v) ← {u, v}; W(u,v) ← wu + wv;
10     L(u,v) ← true;
11     Vcad ← adj(u) ∩ adj(v);
12     While Vcad ≠ ∅
13       u′ ← the first element of Vcad;
14       C(u,v) ← C(u,v) ∪ {u′}; W(u,v) ← W(u,v) + wu′;
15       L(u′,u) ← true; L(u′,v) ← true;
16       Vcad ← Vcad\{u′};
17       For each v′ ∈ Vcad
18         If (u′, v′) ∈ E
19           L(u′,v′) ← true;
20         Else
21           Vcad ← Vcad\{v′};
22         End if;
23       End for;
24     End while;
25   If Wmax < W(u,v)
26     Cmax ← C(u,v), Wmax ← W(u,v);
27     End if;
28   End for;
29 For each u ∈ V and adj(u) = ∅
30   If wu > Wmax
31     Cmax ← {u}; Wmax ← wu;
32   End if;
33 End for;
34 Return Cmax;
End
```

The complexity of the function findMWC(G) depends on the following four steps:

1) Find the set of adjacent vertices of all vertices in G (lines 1–3) that have a complexity of $O(|V|^2)$.
2) Initiate the label for all edges in G (lines 4–6) that have a complexity of $O(|E|)$.
3) Find all maximal cliques in G (lines 8–28) that have a complexity of $O(|E| \times |V| \log|V|)$.
4) Compare C_{max} with the weight of all independent vertices (lines 29–33) that have a complexity of $O(|V|)$.

Then, the final complexity of function findMWC(G) is $O(|E| \times |V| \log|V|)$.

Note that each edge (u, v) of G is attached a label, $L_{(u,v)}$. By setting $L_{(u,v)}$ = true for each validated edge, the number of available edges for constructing the maximal cliques hereafter is much less. For a graph G with $|V|$ vertices, the total number of maximal cliques of G, in which each maximal clique contains at least one edge, is at most $|V|/2$. Therefore, the code part of finding all maximal cliques in G (lines 8–28) actually has a complexity of $O(|V|^2 \log|V|)$.

3. Scheduling Bursts Corresponding with Vertices in MWC

Since each vertex (b_i^k) of C_{max} contains the full information on which channel its corresponding burst will be scheduled, the function scheduleBurstsfromMWC(C) only distributes burst b_i on channel k. Therefore, its complexity is $O(|V|)$. In summary, the complexity of MWC-GS is $O(|V|^2 \log|V|)$.

4. Extension of MWC-GS with Void Filling

In the MWC-GS algorithm, the scheduling is performed without void filling, in which the start time of an arriving burst is compared with the LAUT of data channels. However, in the case with void filling, the idle bandwidth (void), which is

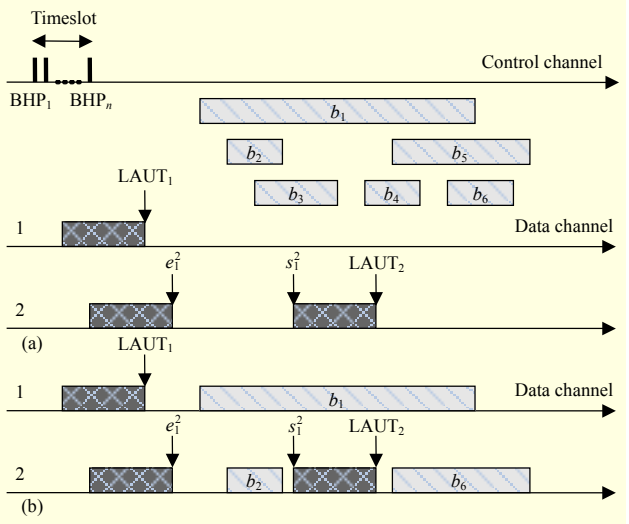


Fig. 6. Example of (a) arriving bursts and (b) optimal schedule with void filling on Channel 2.

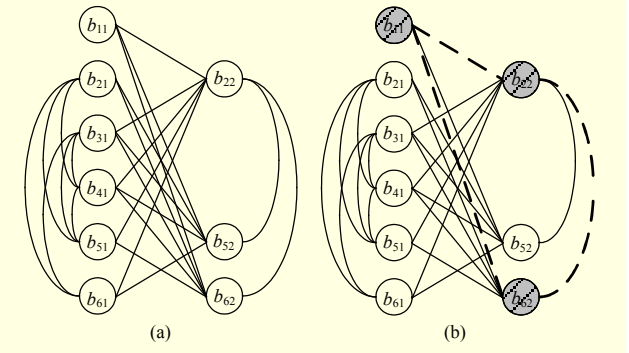


Fig. 7. (a) Interval graph, which represents scheduling possibilities with void filling of arriving bursts in Fig. 6(a), and (b) MWC that corresponds to optimal schedule in Fig. 6(b).

generated between the scheduled bursts, is also considered. Namely, the MWC-GS with void filling, called MWC-GS-VF, first tries scheduling an arriving burst on one of the voids. In the unsuccessful case, the condition in line 3 of function constructGraph(I, W) is considered.

With regards to the arriving bursts in Fig. 6(a), an interval graph representing the scheduling possibilities with void filling is constructed (see Fig. 7(a)). An MWC is then found (see Fig. 7(b)) that corresponds with the optimal schedule featured in Fig. 6(b).

The extension of MWC-GS-VF mainly affects the construction of the interval graph, in which the condition in line 3 of function constructGraph(I, W) is replaced by ($s_i > e_1^k$ and $(s_i + w_i) < s_2^k$) or ($s_i >$ LAUT^k). The functions of findMWC(G) and scheduleBurstsfromMWC(C) do not have any changes.

V. Simulation and Analysis

MWC-GS and MWC-GS-VF are implemented in NS2, with the support of the package obs0.9a [12], on a PC of 2.4 GHz Intel Core 2 CPU, 2G RAM. We compare their efficiency, based on the packet loss probability and the computational time, with that of BATCHOPT. The simulation parameters are as follows: the timeslot is 700 μs; the arrival of bursts is modeled as a Poisson process with loads from 0.1 to 0.9 Erlangs; and the burst length, which is an integer multiple of the block (packet) of 64 bytes, has an exponential distribution.

We consider two cases of simulation networks. The first one is a dumbbell (see Fig. 8) that has ten edge nodes (E_i, i = 0, 1, … , 9) linked to two core nodes (C_0, C_1). The links connecting the edge nodes to the two cores have a bandwidth of 10 Gbps; whereas, the single link that connects the two cores has a 30 Gbps bandwidth. All the links include one control and four data channels. We set the bursty traffics between the pair of edges (E_i, E_{i+5}), i = 0, 1, … , 4, to investigate the efficiency of the group scheduling algorithms at C_0.

The second simulation network is an NSFNet (see Fig. 9) with 10 Gbps bandwidth links in which each also has one control and four data channels. The distances in kilometers are considered to be the propagation delays (μs). The bursty traffics are set between the pairs of nodes. Without loss of generality, node 9 is chosen to investigate the efficiency of the group scheduling algorithms.

Figures 10 and 11 show that MWC-GS has a packet loss probability approaching that of BATCHOPT, while the efficiency of MWC-GS-VF is equivalent to that of BATCHOPT. The reason for this is that in BATCHOPT all the previous scheduled bursts are removed and rescheduled with the new arriving bursts simultaneously. This approach helps rearrange the position of scheduled bursts more optimally and

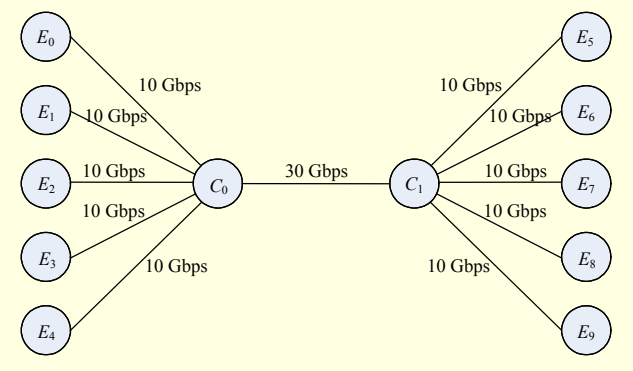


Fig. 8. Simulation network of dumbbell.

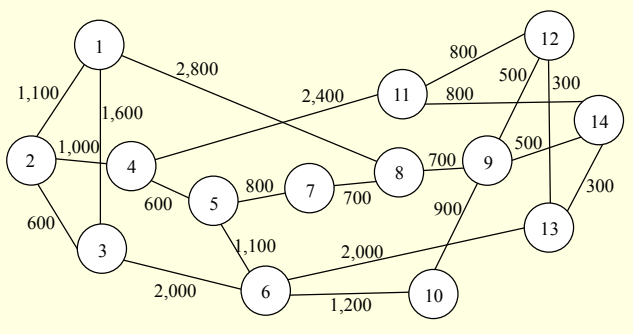
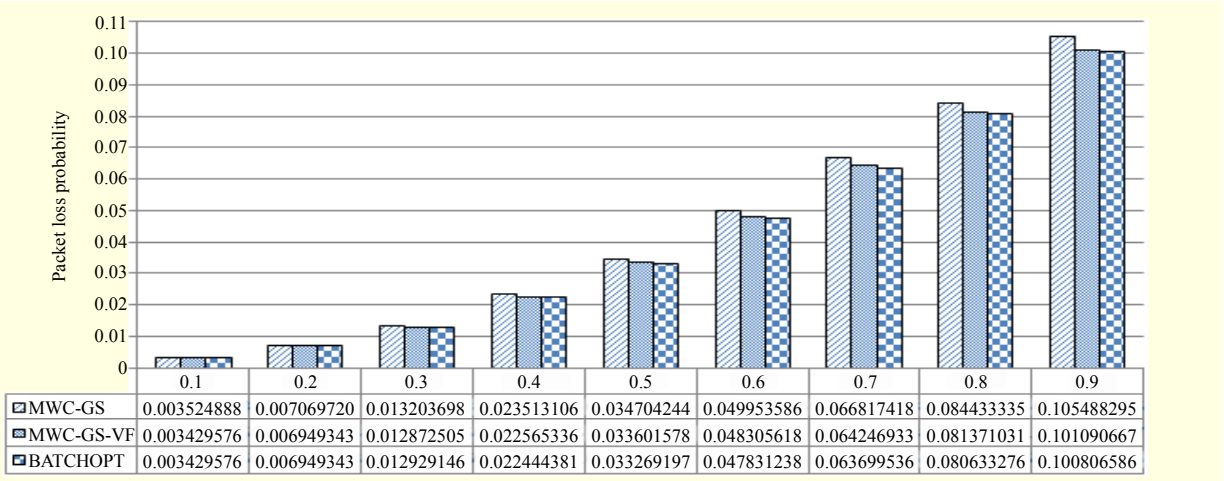


Fig. 9. Simulation network of NSFNet with 14 nodes.

generate new idle spaces for newly arriving bursts; whereas, both MWC-GS and MWC-GS-VF do not consider scheduled bursts. Namely, MWC-GS only considers the available bandwidth after the LAUT of data channels. With MWC-GS-VF, it firstly attempts to schedule arriving bursts into the voids[0] between the scheduled bursts; if it is unable to do so, then MWC-GS is called instead for scheduling. Therefore, BATCHOPT reaches the minimal packet loss probability, whereas our proposals only obtain near-minimal ones.

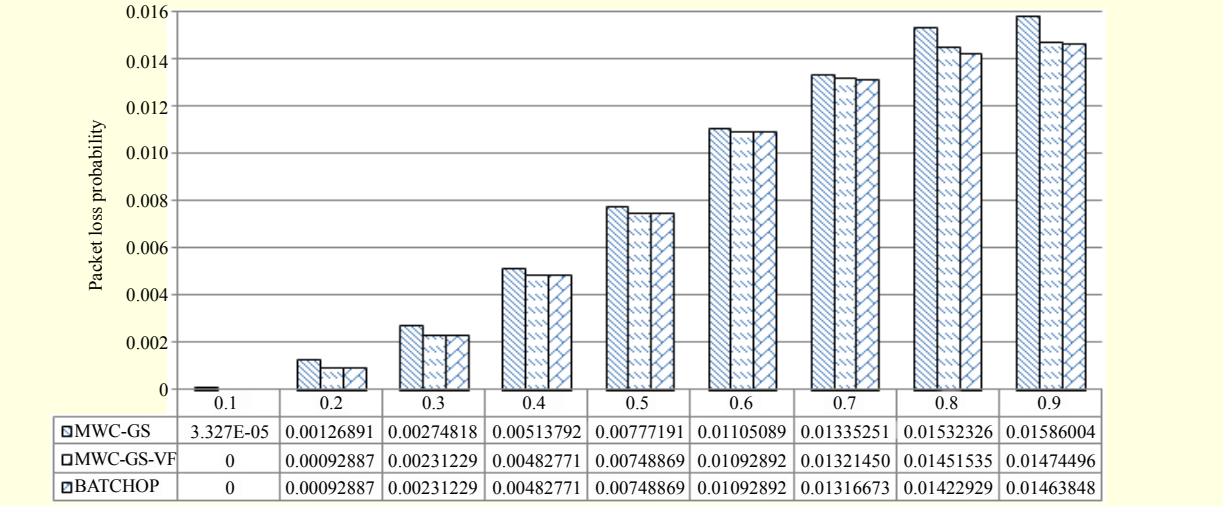
However, in BATCHOPT, if a burst is rescheduled to a new position that differs from the initial scheduled position (that is, on a different wavelength channel), then it is considered as a rescheduling. Then, a BCP must be generated and sent to its downstream nodes to remove the resources that were reserved by the previous BCP. After that, the BCP looks for a new schedule for its burst. This means that more BCPs would be generated to notify the changes of scheduling status, and more updates in the existing signaling protocol would then be required; thus, system complexity is increased. On the contrary, our proposals do not have these drawbacks.

With regards to the computational complexity, as proved in Section IV, MWC-GS (and MWC-GS-VF) has a complexity of O($|V|^2 \log|V|$), $|V| = m \times n$, where n is the number of arriving bursts and m is the number of data channels. Since m is constant, MWC-GS then has a complexity of O($n^2 \log(n)$).



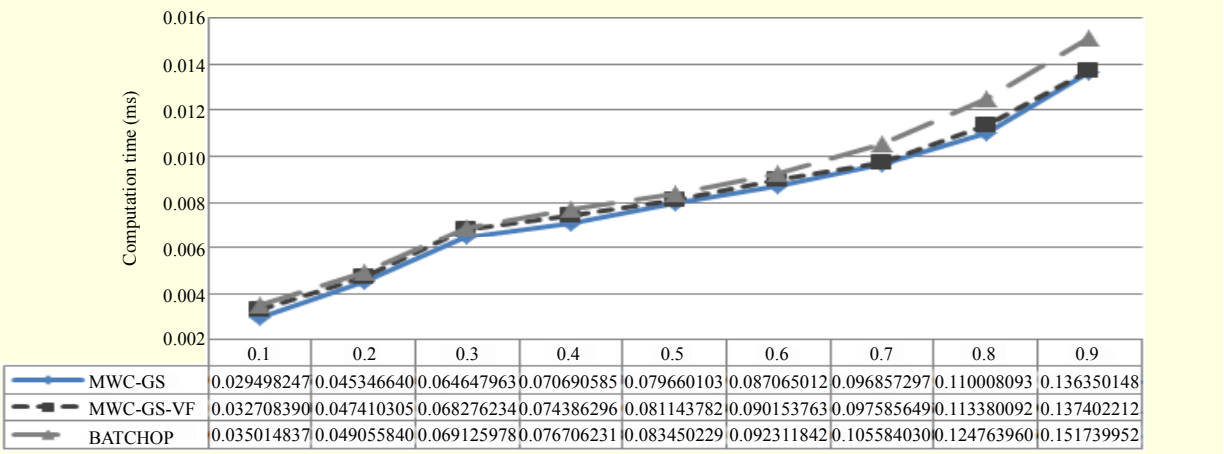
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
MWC-GS	0.003524888	0.007069720	0.013203698	0.023513106	0.034704244	0.049953586	0.066817418	0.084433335	0.105488295
MWC-GS-VF	0.003429576	0.006949343	0.012872505	0.022565336	0.033601578	0.048305618	0.064246933	0.081371031	0.101090667
BATCHOPT	0.003429576	0.006949343	0.012929146	0.022444381	0.033269197	0.047831238	0.063699536	0.080633276	0.100806586

Fig. 10. Comparison based on packet loss probability among MWC-GS, MWC-GS-VF, and BATCHOPT with dumbbell network and loads from 0.1 to 0.9 Erlangs.



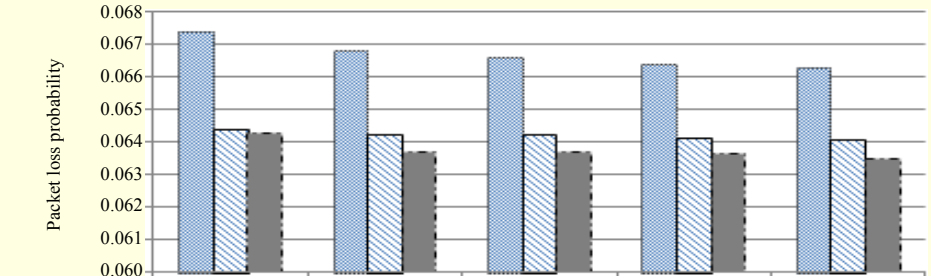
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
MWC-GS	3.327E-05	0.00126891	0.00274818	0.00513792	0.00777191	0.01105089	0.01335251	0.01532326	0.01586004
MWC-GS-VF	0	0.00092887	0.00231229	0.00482771	0.00748869	0.01092892	0.01321450	0.01451535	0.01474496
BATCHOP	0	0.00092887	0.00231229	0.00482771	0.00748869	0.01092892	0.01316673	0.01422929	0.01463848

Fig. 11. Comparison based on packet loss probability among MWC-GS, MWC-GS-VF, and BATCHOPT with NSFNet network and loads from 0.1 to 0.9 Erlangs.



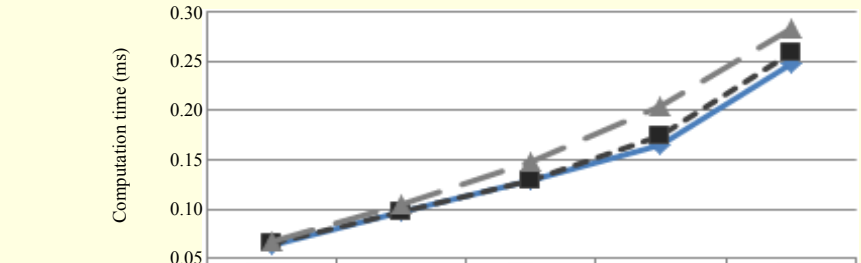
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
MWC-GS	0.029498247	0.045346640	0.064647963	0.070690585	0.079660103	0.087065012	0.096857297	0.110008093	0.136350148
MWC-GS-VF	0.032708390	0.047410305	0.068276234	0.074386296	0.081143782	0.090153763	0.097585649	0.113380092	0.137402212
BATCHOP	0.035014837	0.049055840	0.069125978	0.076706231	0.083450229	0.092311842	0.105584030	0.124763960	0.151739952

Fig. 12. Comparison based on computational time (ms) among MWC-GS, MWC-GS-VF, and BATCHOPT with loads from 0.1 to 0.9 Erlangs.



	600	700	800	900	1,000
MWC-GS	0.067381333	0.066817418	0.066610286	0.066390601	0.066283236
MWC-GS-VF	0.064410458	0.064246933	0.064239005	0.064121399	0.064092327
BATCHOPT	0.064284263	0.063699536	0.063683679	0.063668483	0.063485467

Fig. 13. Comparison based on packet loss probability among MWC-GS, MWC-GS-VF, and BATCHOPT with timeslot sizes from 600 μs to 1,000 μs.



	600	700	800	900	1,000
MWC-GS	0.063635273	0.096857297	0.128667413	0.165035091	0.248370326
MWC-GS-VF	0.064823035	0.097585649	0.129723244	0.174608602	0.259688062
BATCHOPT	0.067392921	0.105584030	0.147224444	0.204120929	0.284303139

Fig. 14. Comparison based on computational time (ms) among MWC-GS, MWC-GS-VF, and BATCHOPT with timeslot sizes from 600 μs to 1,000 μs.

Therefore, the complexity of MWC-GS is smaller than that of BATCHOPT, $O(N^2\log(N))$, where N ($N > n$) is the total number of arriving and removed bursts. This is also illustrated in our simulation results in Fig. 12.

To investigate the impact of timeslot size on the efficiency of the scheduling algorithms, we implement timeslot sizes ranging from 600 μs to 1,000 μs. The simulation results show that there is a slight decrease in the packet loss probability (see Fig. 13) and an increase in the computational time (see Fig. 14) as the timeslot size is increased. This is understandable, because when there are more bursts to be scheduled then there are more burst arrangements and then more opportunities to reduce the lost packets. However, timeslot size cannot be expanded arbitrarily, because it depends on the allowed delay in which a burst transits at each node. Regardless of the timeslot size, MWC-GS and MWC-GS-VF always approach an optimal solution.

VI. Conclusion

In this paper, near-optimal algorithms for group scheduling on multiple channels have been proposed. Our approach was to formulate the group scheduling in terms of the problem of finding a maximum-weight clique (MWC) from an interval graph. Our proposed algorithms (MWC-GS and MWC-GS-VF) almost reach an optimal solution without rescheduling, as in BATCHOPT. Additionally, we proposed a new method to find such an MWC. The approach has a complexity of $O(|V|^2 \log|V|)$, which is equivalent to that of the current best algorithms.

References

[1] Y. Chen, C. Qiao, and X. Yu, "Optical Burst Switching: A New Area in Optical Networking Research," *IEEE Netw.*, vol. 18, no. 3, June 2004, pp. 16–23.

[2] S. Charcranoon et al., "Group-Scheduling for Optical Burst Switched (OBS) Networks," *IEEE Global Telecommun. Conf.*, vol. 5, Dec. 1–5, 2003, pp. 2745–2749.

[3] H. Zheng, C. Chen, and Y. Zhao, "Optimization Scheduling for Optical Burst Switching Networks with Wavelength

Conversion," *Int. Conf. Commun. Technol.*, Guilin, China, Nov. 27–30, 2006, pp. 1–4.
[4] J.S. Turner, "Terabit Burst Switching," *J. High Speed Netw.*, vol. 8, 1999, pp. 3–16.
[5] Y. Xiong, M. Vandenhoute, and H.C. Cankaya, "Control Architecture in Optical Burst-Switched WDM Networks," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, Oct. 2000, pp. 1838–1851.
[6] N.H. Quoc, V.V.M. Nhat, and N.H. Son, "A New Algorithm of Group Scheduling in OBS Core Nodes." *Int. Conf. Adv. Technol. Commun.*, Ho Chi Minh, Vietnam, Oct. 16–18, 2013, pp. 592–596.
[7] A. Kaheel and H. Alnuweiri, "Batch Scheduling Algorithms: A Class of Wavelength Schedulers in Optical Burst Switching Networks," *IEEE Int. Conf. Commun.*, vol. 3, May 16–20, 2005, pp. 1713–1719.
[8] G.B. Figueiredo, E.C. Xavier, and N.L.S. da Fonseca, "Optimal Algorithms for the Batch Scheduling Problem in OBS Networks," *Comput. Netw.*, vol. 56, no. 14, Sept. 28, 2012, pp. 3274–3286.
[9] N.H. Quoc, V.V.M. Nhat, and N.H. Son, "Group Scheduling for Multichannels in OBS Networks," *REV J. Electron. Commun.*, vol. 3, no. 3–4, Dec. 2013, pp. 134–137.
[10] E.M. Arkin and E.B. Silverberg, "Scheduling Jobs with Fixed Start and End Times," *Discrete Appl. Math.*, vol. 18, no. 1, Sept. 1987, pp. 1–8.
[11] A.A. Bertossi and A. Gori, "Total Domination and Irredundance in Weighted Interval Graphs," *SIAM J. Discrete Math.*, vol. 1, no. 3, Aug. 1988, pp. 317–327.
[12] *OBS-ns Simulator*, Optical Internet Research Center. Accessed Oct. 2007. http://www.wine.icu.ac.kr/obsns/index.php

Vo Viet Minh Nhat received his MS degree in communication engineering from the Francophone Institute for Computer Science, Hanoi, Vietnam, in 2000 and his PhD degree in cognitive informatics from the University of Quebec, Montreal, Canada, in 2007. His research interests include optical packet/burst-based switching, quality of service, optimization, and traffic engineering.

Nguyen Hong Quoc received his MS degree in computer science from the Faculty of Sciences, Hue University, Vietnam, in 2010. His research interests are in the fields of all-optical networks with emphasis on packet/burst-based switching, scheduling, and quality of service.

Nguyen Hoang Son received his PhD degree in database theory from the Information Technology Institute, Ho Chi Minh, Vietnam, in 2006. His main research interests include computation and combinatorial problems in database theory; computational complexity theory; and discrete mathematics.