Joint scheduling for optical grid applications

Yan Wang, Yaohui Jin, Wei Guo, Weiqiang Sun, and Weisheng Hu
State Key Laboratory on Fiber-Optic Local Area Networks and Advanced Optical Communication Systems, Shanghai Jiao Tong University, Shanghai, China
jinyh@sjtu.edu.cn

Min-You Wu
Grid Computing Center, Shanghai Jiao Tong University, Shanghai, China

Received August 22, 2006; revised November 23, 2006; accepted January 17, 2007; published February 28, 2007 (Doc. ID 74280)

Optical networking technologies are expected to play an important role in creating an efficient infrastructure for supporting advanced grid applications. Since both the scheduling methods in grid computing and optical networks are limited to be directly used to achieve optical grid scheduling, we propose a new, to the best of our knowledge, joint scheduling model by extending the classic list scheduling algorithm to achieve communication contention aware task scheduling for the optical grid applications. An effective adaptive routing scheme is also proposed to improve the performance of the extended list scheduling. The impacts of different routing schemes on the extended list scheduling are comparatively investigated by simulations. © 2007 Optical Society of America

1. Introduction

Grid applications for science research evolve into a new generation to solve more and more complex problems. It integrates geographically distributed resources such as instruments, data storage, and sensors as well as supercomputers by open middleware technologies [1]. Usually, bandwidth- and delay-guaranteed communication services are required for these large-scale, data-intensive, and real-time applications. Although today’s Internet bandwidth gets cheaper with the exponential advance in communication techniques, optical networking technologies may be better suited to fulfill these requirements, i.e., to offer huge capacity and relatively low latency, as well as dynamic control and allocation of bandwidth at various granularities [2,3]. The reason is because circuit-switched optical networks avoid hop-by-hop store-and-forwarding in IP networks. Thus optical networking is expected to play an important role in creating an efficient infrastructure for supporting such advanced grid applications, which is called optical grid or photonic grid [4].

An essential issue of optical grid applications is scheduling, which is the spatial and temporal assignment of the tasks of a grid application to the required resources while satisfying their precedence constraints. One of the primary objectives of scheduling is to minimize the scheduling span, i.e., the completion time of the last finished task. In task scheduling, the grid application can be modeled by a directed acyclic graph (DAG), where a node represents a task and an edge represents the communication between two adjacent tasks [5]. Previously, many algorithms have been proposed for this DAG scheduling [6–8]. However, most of them assume an ideal communication system in which the resources are fully connected and the communication between any two resources can be provisioned whenever they need. These assumptions are not consistent with those of the practical circuit-switched optical networks because the port and link communication contention is ignored. Recently, Sinnen and Sousa [9] proposed a task scheduling model with communication contention awareness for parallel computing. The purpose of their work is to focus on the scheduling accuracy when compared with the ideal models. However, they do not consider how to exploit the networks resources efficiently so as to improve the scheduling performance.

On the aspects of optical networking, previous efforts focused on resource optimization under static or dynamic traffic [10]. There has been recent interest in scheduled
traffic [11,12] that can provision scheduled dedicated channels or bandwidth pipes at a specific time with certain duration. Although these different traffic models are valid and useful in many circumstances, they still present challenges when used for optical grid applications. Because the traffic demands in these models are independent of each other in both time and space domains, they are not able to reflect the traffic characteristics of grid applications in which the communications between the tasks have precedence constraints in their execution sequence.

In this paper, we study optical grid scheduling problems. Using the framework in [9], we regard the optical network as a resource in a similar way as a computer or storage and propose a joint scheduling model for the optical grid applications by incorporating the link communication contentions of the optical network into task scheduling. As list scheduling is one of the most common heuristics for the DAG scheduling, we extend the classic list scheduling algorithm to implement (1) task scheduling that is to schedule the nodes of DAG onto the grid resources and (2) communication scheduling that is to schedule the edges of DAG onto the optical network’s links along the lightpaths. In the communication scheduling, we propose an adaptive routing scheme to determine the earliest start route for each edge scheduling and further reduce the total scheduling length. The performance of the joint scheduling algorithm under different routing schemes is comparatively evaluated by simulation. The results show that our proposed adaptive routing scheme can do a better job, in terms of schedule length and network utilization, to enhance the extended list scheduling performance compared with other routing schemes, e.g., fixed and alternative routing schemes.

The rest of the paper is organized as follows. Section 2 introduces the optical grid joint scheduling model. Section 3 describes the extended list algorithm to fulfill the implementation of the joint scheduling. This section also presents our adaptive routing scheme used to improve the scheduling performance. In Section 4, a simple scheduling example is given. Next in Section 5, the impacts of different routing schemes on the joint scheduling performance are comparatively investigated by extensive simulations. Finally, the paper concludes in Section 6.

2. Optical Grid Resource Scheduling Model
In this section, we describe our optical grid resource (OGR) scheduling model in detail. It is composed of three parts: resource model, task model, and the constraints for DAG scheduling.

2.A. Resource Model
In addition to traditional grid resources such as computers, storages, and I/O devices, the optical network is also regarded as a grid resource in our model. In optical networks, possible resources include optical switch nodes and fiber links. Figure 1 depicts an example of the O-Grid extended resource model in which there are seven grid resources and four optical switches. The adjacent optical switches are connected via the WDM fiber links. Each grid resource is connected to the optical switch via the access link. Typically, the access link is a 1 Gbits/s Ethernet (GbE) or 10 Gbits/s Ethernet (10 GbE) link. Therefore, the traffic signals from and to the end grid resources can be mapped onto wide-area SONET/SDH circuits or all-optical light paths, depending on whether the optical switches are electronic, time-division multiplexed,
SONET/SDH cross connects, or all-optical, WDM switches. In our model, the optical switch is assumed to be equipped with all-wavelength converters, thus there is no wavelength continuity constraint for routing. So the wavelength can be randomly assigned as long as the bandwidth in corresponding links is enough.

Then our OGRs model can be formulated as an extended resource graph \( \text{OGR} = (N, L, \text{type}, \text{bw}, d) \), where

- \( N \) is a set of network nodes. \( N = R + S \), where a node \( r \in R \) represents a grid resource and a node \( s \in S \) represents an optical switch.
- \( L \) is a set of undirected links. \( L = L_A + L_T \), where a link \( l \in L_A \) represents the access link between a grid resource and an optical switch, and a link \( l \in L_T \) represents the transmission link between two optical switches.
- \( \text{type}(r) \), associated with a resource \( r \in R \), is a numeral scalar that represents the type of \( r \), for example, 1 represents computer, 2 storage, and 3 I/O device, etc.
- \( \text{bw}(l) \), associated with a link \( l \in L \), represents the link's bandwidth.
- \( d(l) \), associated with a link \( l \in L \), denotes the distance of the link.

It should be noted that an access link is dedicated to the traffic to and from the connected resource and cannot be shared by other traffic at the same time, while a transmission link can be shared by many traffic demands simultaneously.

### 2.B. Task Model
A grid application can be modeled by a workflow composed of several sequence-dependent tasks or jobs. So we can represent the grid application as a DAG [5]. We formulate the task model as \( G_{\text{DAG}} = (V, E, \text{type}, c, w) \), where

- \( V \) represents a set of vertices. A vertex \( v \in V \) denotes a task in the grid application.
- \( E \) represents a set of edges. An edge \( e_{mn} \in E \) denotes the communication from vertex \( v_m \) to vertex \( v_n \).
- \( \text{type}(v) \), associated with a task \( v \in V \), is a numeral scalar that represents the type of task \( v \), for example, 1 represents the computation task, 2 the storage task, and 3 I/O device operation task, etc.
- \( c(v) \), associated with a task \( v \in V \), denotes the average execution time required by \( v \) on a reference resource in the heterogeneous system.
- \( w(e) \), associated with an edge \( e \in E \), denotes the data volume transmitted on the edge \( e \).

Figure 2 shows an example DAG with each vertex assigned a type and an average execution cost, and each edge assigned a weight. In the DAG, all tasks are executed in sequential order, that is, a task vertex cannot begin execution until all the data from its predecessors have arrived and no output is available until the task has finished and at that time all outputs are available for communication simultaneously. The set of all direct predecessors of task \( v \) is denoted by \( \text{pred}(v) \) and the set of all direct successors of \( v \) is denoted by \( \text{succ}(v) \). A task vertex \( v \) without predecessors, \( \text{pred}(v) = \emptyset \), is named source node and if it is without successors, \( \text{succ}(v) = \emptyset \), it is named sink node.

### 2.C. Directed Acyclic Graph Scheduling
With the resource and task model defined above, the next step is to generate an execution schedule of DAG to the target optical grid extended resource system. The DAG
scheduling consists of two parts: one is task scheduling and the other is communication scheduling. Task scheduling is to schedule the task vertices of DAG onto the grid resources in the extended resource graph, while communication scheduling is to schedule the edges of DAG onto the links in the extended resource graph. Every task vertex or communication edge is assigned a start and a finish time after the scheduling. The objective of DAG scheduling considered in this paper is to minimize the schedule length, that is, the finish time of the last task vertex, without violating the precedence dependency among the tasks constrained by the DAG.

2.C.1. Task Scheduling

To describe a schedule of a task vertex \( v \in V \) on a grid resource \( r \in R \), the following terms are defined.

- \( rsc(v) \) denotes the resource to which task \( v \) is allocated, \( rsc(v) \in R \).
- \( ts(v,r) \) denotes the start time of task vertex \( v \) on the resource \( r \).
- \( h(v,r) \) denotes the heterogeneity factor that is determined by measuring the difference in processing capabilities (e.g., speed) of resource \( r \) and the reference resource with respect to task \( v \) [13]. Then the actual execution time of \( v \) on \( r \) is given by \( c(v)h(v,r) \). In a homogeneous resource system, the execution time is equivalent to \( c(v) \), that is, the average execution time of \( v \).
- \( tf(v,r) = ts(v,r) + c(v)h(v,r) \) denotes the finish time of task \( v \) on resource \( r \).
- \( tfa(v) = \max_{v \in V, rsc(v) = v} [tf(v,r)] \) denotes the first available time of resource \( r \). This is the time at which all the former scheduled tasks on it have finished execution and after which new tasks can be scheduled on it.
- \( sl = \max_{v \in V} [tf(v)] \) denotes the schedule length, where \( tf(v) \) denotes the finish time of task \( v \), \( tfa(v) = tf(v,rsc(v)) \).

Type constraint. For any task \( v \in V \), \( v \) can only be scheduled onto the resource \( r \in R \) with \( type(r) = type(v) \).

Resource no time-overlap constraint. For any two tasks \( v_m,v_n \in V \), with \( type(v_m) = type(v_n) \), if they are assigned to the same resource \( r \in R \), i.e., \( rsc(v_m) = rsc(v_n) = r \), then

\[
   tf(v_m,r) \leq ts(v_n,r) \quad \text{or} \quad tf(v_n,r) \leq ts(v_m,r).
\]

This constraint ensures that, if a resource is utilized for processing one task, then it can be used for another task only after or before the task has been executed, but not during.

Data ready time. As shown in the DAG, a task vertex can start execution when all the data from its parent nodes has arrived. So the earliest start time of a task \( v_m \in V \) on resource \( r \in R \) critically depends on the task’s data ready time (DRT), which is defined as the last arrival time of edges from its predecessors [9].

\[
   t_{dr}(v_m,r) = \begin{cases} 
   \max_{e_{nm} \in E, v_a = \text{pred}(v_m)} \{tf(e_{nm})\} & \text{if } \text{pred}(v_m) \neq \emptyset \\
   0 & \text{otherwise}
   \end{cases}
\]

where \( tf(e_{nm}) \) is the edge finish time of the communication associated with edge \( e_{nm} \) (which will be discussed in Subsection 2.C.2). So the actual start time of task \( v_m \) on resource \( r \) is determined by

\[
   ts(v_m,r) = \max\{tfa(r), t_{dr}(v_m,r)\}.
\]

An alternative approach to determine the task’s start time considers idle time slots between already scheduled tasks in order to insert the task into a slot if appropriate. This technique has a higher complexity and we do not discuss here.

2.C.2. Communication Scheduling

If two adjacent tasks are scheduled on the same resource, the communication cost of the edge between them is assumed to be negligible and is set to zero. However, when a communication, represented by the edge \( e \in E \), is performed between two distinct resources \( r_{src} \) and \( r_{dst} \), the routing algorithm of the O-Grid system returns a route from \( r_{src} \) to \( r_{dst} \). The edge \( e \) is then scheduled on each link along the route. Corresponding to the task scheduling, \( ts(e,l) \) and \( tf(e,l) \) denote the start and finish time of edge \( e \in E \) on link \( l \in L \), respectively.
Unlike a task vertex that is only scheduled on one grid resource, an edge is scheduled on all the links along the route, so the situation about the communication scheduling is a bit more complicated. The scheduled communication should start and end on all the links along the route simultaneously for the route is a cutting-through path in the network without any store and forwarding.

Following the idea in [14], we associate each link \( l \in \mathbf{L} \) with a capacity availability function \( A(t, l) \) that records the available bandwidth of link \( l \) at time \( t \). This function is given by

\[
A(t, l) = \begin{cases} 
\pi_z & t_z \leq t < t_{z+1} \\
\mathbf{bw}(l) & t \geq t_{z_{\max}} 
\end{cases}
\]

(4)

where \( \pi_z \leq \mathbf{bw}(l) \) and \( z=1,2,\ldots,z_{\max} \). \( t_z \) is a time point where a change of the available bandwidth takes place. After \( t_{z_{\max}} \), all the bandwidth \( \mathbf{bw}(l) \) of link \( l \) remain available. Using \( A(t, l) \), we can determine when and how much link bandwidth is available for a new edge scheduling. It should be noted that after each edge scheduling, the capacity availability function \( A(t, l) \) of the concerned link must be updated for the next edge scheduling, thus the time point \( t_z \), for \( z=1,2,\ldots,z_{\max} \), is also changed in consistent with the update.

Corresponding to the first available time of task \( t_{f_0}(v) \), we denote \( t_{f_0}(l, B) \) as the first available time of a link \( l \in \mathbf{L} \) for the new edge scheduling with communication bandwidth \( B \).

\[
t_{f_0}(l, B) = \min\{t_z|A(t, l) \geq B, \forall t \in [t_z, \infty)\}.
\]

(5)

Let \( Rt=\langle l_1, l_2, \ldots, l_k \rangle \), \( l_i \in \mathbf{L} \) for \( i=1,2,\ldots,k \), be the route for the communication of \( e_{mn} \in \mathbf{E}, v_m, v_n \in \mathbf{V} \), and \( \mathbf{rsc}(v_m) \neq \mathbf{rsc}(v_n) \). Because \( l_1, l_k \in \mathbf{LA} \) are the access links with much lower bandwidth than other transmission links along the route, the bandwidth of the whole route is determined by

\[
\mathbf{bw}(Rt) = \min\{\mathbf{bw}(l_1), \mathbf{bw}(l_k)\} \quad l_1, l_k \in Rt.
\]

(6)

The start time of \( e_{mn} \in \mathbf{E} \) on the route \( Rt \) is given by

\[
t_s(e_{mn}, Rt) = \max\{t_{f_0}(l, \mathbf{bw}(Rt)), t_f(v_m, \mathbf{rsc}(v_m))\}.
\]

(7)

Then the finish time of \( e_{mn} \in \mathbf{E} \) on \( Rt \) is

\[
t_f(e_{mn}, Rt) = t_f(e_{mn}, Rt) + c(e_{mn})/\mathbf{bw}(Rt).
\]

(8)

Finally we get the finish time of scheduled edge \( e_{mn} \in \mathbf{E} \),

\[
t_f(e_{mn}) = \begin{cases} 
t_f(v_m) & \text{if } \mathbf{rsc}(v_m) = \mathbf{rsc}(v_n) \\
t_f(e_{mn}, Rt) & \text{otherwise} 
\end{cases}
\]

(9)

\( t_f(e_{mn}) \) is then used to determine the DRT as defined in Eq. (2). After scheduling the edge \( e_{mn} \) on the route \( Rt \), the start and finish time of \( e_{mn} \) on each link along the route \( Rt \) can be determined, that is,

\[
t_s(e, l) = t_s(e_{mn}, Rt), \quad t_f(e, l) = t_f(e_{mn}, Rt), \quad \text{for } l \in Rt.
\]

(10)

Then the capacity availability function \( A(t, l) \) of each link \( l \) along the route \( Rt \) is updated by \( t_s(e, l) \) and \( t_f(e, l) \).

### 3. Extended List Scheduling Algorithm

Based on the O-Grid scheduling model, algorithms are required to be proposed to fulfill the scheduling. Our scheduling problem under communication contention model has been proved to be NP-hard which can be solved in non-deterministic polynomial time [9]. The heuristics therefore try to produce near optimal solutions in acceptable solving time. As list scheduling is one of the most common heuristics for the DAG scheduling, we extend the classic list scheduling algorithm to implement the joint scheduling based on our proposed O-Grid scheduling model. The key of the extension to the list scheduling algorithm lies in the introduction of the communication scheduling into the task scheduling as described in Subsection 2.C.2.
The description of the extended algorithm is given in Fig. 3. The algorithm is composed of two steps. In the first step, all the tasks are sorted into a scheduling list \( \text{LIST} \) according to their priorities. A common and usually good priority is the task's bottom level \( bl \), which is the length of the longest path leaving the task [9,15]. It is recursively defined as

\[
bl(v_n) = c(v_m) + \max_{v_n \in \text{succ}(v_m)} \{ w(e_{mn}) + bl(v_n) \}. \tag{11}
\]

In the second step, the algorithm iterates over the list. At each step of the first for-loop in Fig. 3, an unscheduled task \( v_n \) is selected from the list with the highest priority and then allocated onto the required resource with earliest finish time. To find the resource that allows the earliest finish time of \( v_n \), the second for-loop will tentatively schedule \( v_n \) on every resource with the same type as \( v_n \). After the tentative scheduling, we can find the required resource \( r_{\text{min}} \) and then we will schedule task \( v_n \) on \( r_{\text{min}} \). First, each incoming edge \( e_{mn} \) of task \( v_n, v_m \in \text{pred}(v_n) \), is scheduled on the determined route from \( rsc(v_m) \) to \( r_{\text{min}} \) as described in Subsection 2.C.2. Only after that, when the finish time of all the incoming edges of \( v_n \) are known, task \( v_n \)'s DRT can be determined and it can be finally scheduled on the found resource \( r_{\text{min}} \).

It should be noted that in the algorithm, although the route \( Rt = (l_1, l_2, \ldots, l_k) \) is denoted as the route from the source resource to the destination resource, the actual route calculation is based on the optical network, as the two end links \( l_1 \) and \( l_k \) are the access links and dedicated to their belonging resources respectively. In the third for-loop of the algorithm in Fig. 3, a definite processing order of the incoming edges is required. In our algorithm, we sort each \( v_m \in \text{pred}(v_n) \) in an increasing order by \( \max_{l \in L_A} \{ t_f(v_m), t_f(l') \} \), where \( l' \in L_A \) is the access link dedicated to the resource \( rsc(v_m) \).

Next we discuss the routing scheme in the joint scheduling algorithm. There are generally three approaches to establish a lightpath in the optical network [16]. The first one is the fixed routing scheme in which a single fixed route is predetermined for each traffic demand. The second one is the fixed-alternate routing scheme in which

---

**Step 1:** Determine the scheduling list

Determine the task vertices' bottom level

Sort each task \( v \in V \) into a list \( \text{LIST} \) by decreasing order of their bottom levels

**Step 2:** Sequential scheduling over the list

For each task vertex \( v_n \in \text{LIST} \) do

For each resource \( r \in R \) with \( \text{type}(v_n) = \text{type}(r) \) tentatively do

For each \( v_m \in \text{pred}(v_n) \) in a definite order do

If \( rsc(v_m) \neq r \) then

Find a route \( Rt = (l_1, l_2, \ldots, l_k) \) from \( rsc(v_m) \) to \( r \)

Schedule \( e_{mn} \) on \( Rt \)

Else

Neglect the communication cost of \( e_{mn} \)

End if

End for

Schedule node \( v_n \) on resource \( r \)

End for

Record \( r_{\text{min}} \) so that \( t_f(v_n, r_{\text{min}}) = \min_{r \in R} \{ t_f(v_n, r) \} \)

Schedule edge \( e_{mn} \) on the route from \( rsc(v_m) \) to \( r_{\text{min}} \) for each \( v_m \in \text{pred}(v_n) \)

Schedule the node \( v_n \) on resource \( r_{\text{min}} \)

End for

---

Fig. 3. Extended list scheduling algorithm.
multiple fixed routes are precomputed for each traffic demand and stored in an ordered list at the source node’s routing table. The last one is the adaptive routing scheme that increases the likelihood of establishing a connection by taking network state information into account. For better utilization of the network resources, we adopt an adaptive routing approach. The routing algorithm is proposed by modifying the Dijkstra shortest path algorithm [17]. In the modification, we associate each optical switch node with a time-occupation cost so that we can establish a route that allows the earliest start time. The adaptive routing algorithm is named earliest start route first (ESRF) and is illustrated in Fig. 4.

Some terms used in the ESRF algorithm are defined as follows.
• \( G_{ON} = (S, L_T, d) \) denotes the optical network which is the subgraph of the O-Grid extended resource graph \( OGR = (N, L, \text{type}, \text{bw}, d) \), where \( S \subseteq N \), \( L_T \subseteq L \).
• \( \text{time}(s) \) stores the best estimate of the minimum finish time from the source to node \( s \in S \).
• \( \text{distance}(s) \) stores the best estimate of the shortest distance from the source to node \( s \).
• \( \text{pred}(s) \) stores the predecessor of node \( s \) on the shortest path from the source.
• \( F \) is the set of settled nodes whose shortest distances from the source have been found.
• \( U \) is the set of unsettled nodes.

Differing from the traditional Dijkstra shortest-path algorithm, the ESRF algorithm we proposed includes a metric \( \text{time}(s) \), prior to \( \text{distance}(s) \), to select the best choice of the intermediate nodes \( s \). In Fig. 4(a), lines 2–10 (originating from “begin”) performs a usual initialization. In each iteration of the while-loop (lines 11–15), extract-minimum procedure extract node \( s_i \) with the smallest \( \text{time}(s_i) \) from \( U \). If there is more than one smallest node defined by \( \text{time} \), the smallest one defined by \( \text{distance} \) is extracted. After the extraction, the node \( s_i \) with the smallest time-occupation cost is

\[
\text{ETF}(G_{ON}, s_{src}, s_{dst}, B) \\
// s_{src} is source switch node, \\
// s_{dst} is destination switch node \\
// B is the request bandwidth \\
begin:
\text{for all } s \in S \text{ do}
\quad \text{time}(s) = \infty \\
\quad \text{distance}(s) = \infty \\
\quad \text{pred}(s) = \text{NULL} \\
\text{end for}
\text{set } F = U = \text{empty}
\text{add } s_{src} \text{ to } U
\text{distance}(s_{src}) = 0 \\
\text{time}(s_{src}) = 0 \\
\text{while } U \text{ is not empty do}
\quad s_i = \text{extract - minimum}(U)
\quad \text{add } s_i \text{ to } F
\text{relax - neighbors}(s_i)
\text{end while}
\text{according to } F \text{ and } \text{pred}(\cdot),
\text{output the route from } s_{src} \text{ to } s_{dst}
end

relax - neighbors(s_u)
begin
\text{for each } s_j \text{ adjacent to } s_f, s_j \text{ not in } F \text{ do}
\quad \text{if } \text{time}(s_j) > \text{max} \{\text{time}(s_i), \text{time}(l_{ij}, B)\} \text{ then}
\quad \quad \text{time}(s_j) = \max \{\text{time}(s_i), \text{time}(l_{ij}, B)\}
\quad \quad \text{distance}(s_j) = \text{distance}(s_i) + d(l_{ij})
\quad \quad \text{pred}(s_j) = s_i
\quad \text{add } s_j \text{ to } U
\text{end if}
\quad \text{if } \text{time}(s_j) = \text{max} \{\text{time}(s_i), \text{time}(l_{ij}, B)\} \text{ then}
\quad \quad \text{if } \text{distance}(s_j) > \text{distance}(s_i) + d(l_{ij}) \text{ then}
\quad \quad \quad \text{distance}(s_j) = \text{distance}(s_i) + d(l_{ij})
\quad \quad \quad \text{pred}(s_j) = s_i
\quad \quad \text{add } s_j \text{ to } U
\quad \text{end if}
\text{end if}
\text{end for}
end

Fig. 4. ESRF algorithm. (a) Main procedure; (b) relaxing neighbor procedure.
removed from \(U\) and inserted into \(F\). Then the procedure relax-neighbors relaxes each link \(l_{ij}\) leaving \(s_i\) to \(s_j\), thus updating the estimate time \(s_j\), distance \(s_j\), and the predecessor \(pred(s_j)\) if the earliest start route to \(s_j\) can be improved by going through \(s_i\). The description of procedure relax-neighbors is given in Fig. 4(b). So our ESRF algorithm can find a route that allows the earliest start time from source to destination node. Moreover, the selected route is also a shortest path, if there are more than one earliest start paths.

**Complexity.** The time occupation cost introduced in the modification of Dijkstra algorithm is treated in a similar way the distance cost is utilized, which does not lead to further complexity. So the complexity of our ESRF algorithm is \(O(|S| \log |S| + |L_T|)\), the same as the traditional Dijkstra algorithm [17], where \(|S|\) is the number of optical switch nodes and \(|L_T|\) is the number of optical transmission links. The complexity of the first and second stage of the extended list scheduling algorithm is \(O(|V| \log |V| + |E|)\) and \(O(|R|(|V| + |E|)O(\text{routing})))\), respectively [9], where \(O(\text{routing})\) is the complexity of the routing scheme and \(|V|\), \(|E|\), \(|R|\) are the number of DAG vertices, DAG edges, and grid resources, respectively. Consequently, the total complexity of our proposed extended list scheduling algorithm with ESRF routing scheme is \(O(|V| \log |V| + |E|) + O(|R|(|V| + |E|(|S| \log |S| + |L_T|)))).

## 4. Simple Schedule Example

In this section, we give a simple example to illustrate the optical grid scheduling process and the impact of different routing schemes on the joint scheduling performance. Figure 5(a) shows a DAG of seven task vertices representing a grid application. The tasks have three different types numbered as 1, 2, and 3, respectively. In the DAG, each task vertex is assigned a type and an average execution cost, and each edge is assigned a weight, as shown in the figure. Figure 5(b) shows an extended resource graph in which there are six grid resources interconnected by a four-node optical network. The grid resources also have three different types corresponding to the types of the tasks in the DAG. We assume all the access and transmission links in the resource graph have only one unit bandwidth. The purpose of minimizing the link capacity is to maximize the communication contention. Moreover, it is also assumed that the grid resources of the same type are homogeneous.

The task vertices in the DAG are first sorted into a list according to their bottom levels in decreasing order (see Table 1). Then the task vertices are scheduled to the grid resources one after another. We employ four routing schemes in the extended list scheduling algorithm to evaluate how the routing scheme influences the schedule performance.

- No-routing scheme \(NR\). All the resources are assumed to connect to one optical switch, so that routing is not considered and only access link contention is concerned. Therefore this scheme has less communication contention and is expected to produce the shortest scheduling length.
• Routing scheme $F1R$ represents the fixed routing scheme.
• Routing scheme $F2R$ represents the fixed two-alternate routing scheme.
• Routing scheme $AR$ represents our proposed adaptive routing scheme as described in the ESRF algorithm.

From the results given in Fig. 6 we can see that the no-routing scheme NR has the shortest scheduling length as we expected, since it does not include the transmission link contentions. The schedule lengths under the other three routing schemes are in the order $F1R > F2R > AR$. The scheduling details for the three routing schemes are given in Fig. 7. The left side of each Gantt chart records the time allocations of tasks and communications on all the resources and their corresponding access links, the right side is the time allocations of communications on all the transmission links. The schedules under the three routing schemes have the same schedule list as shown in Table 1. We can observe from Fig. 7 that the first six task vertices have the same scheduling results. The difference happened in the scheduling for the last task $v_6$. All the incoming edges of $v_6$ are scheduled in the order by $e_{56}, e_{16}, e_{06}, e_{26},$ and $e_{46}$, according to the algorithm in Fig. 3. The task $v_6$ can only be scheduled on the resource $r_0$ and $r_3$ for the type constraint. Under routing scheme $F1R$, when $v_6$ is scheduled on $r_3$, the edge $e_{56}$ will use the fixed route $(l_{01}, l_{13})$. Because the link $l_{01}$ has been already occupied by edge $e_{04}, e_{56}$ has to start communication after 15 unit time, which results in the final schedule length of 110. When $v_6$ is scheduled on $r_0$, the final schedule length will be 109, shorter than that of scheduling on $r_3$. Under routing scheme $F2R$, $v_6$ can get a best schedule on $r_3$ with the final schedule length of 99. The reason why $F2R$ does a better job than $F1$ lies in the fact that $e_{56}$ has an alternative route $(l_{02}, l_{23})$ to choose. Although there is still a contention from edge $e_{12}$ along the second route, there is only a 4 unit time delay before $e_{56}$ began to communicate, since the finish time of $e_{12}$ is much earlier than $e_{04}$. Under routing scheme $AR$, $e_{56}$ can be scheduled onto a new route $(l_{02}, l_{21}, l_{13})$, avoiding the contentions both in the link $l_{01}$ and in $l_{23}$, which makes $e_{56}$ communicate immediately after task $n_5$ without any delay. Then routing scheme $AR$ can acquire the same schedule result as no-routing scheme NR.

Each routing scheme not only produces a different schedule length, but different network resource utilization as well. The network utilization $U$ after scheduling is defined below as the ratio of the total occupied bandwidths over the total supplied bandwidth of all the transmission links during the whole scheduling span.

$$U = \frac{\sum_{l \in L_T} \sum_{z=1}^{z_{\text{max}}-1} [bw(l) - A(t, l)][t_{z+1} - t_z]}{sl \times \sum_{l \in L_T} bw(l)}. \quad (12)$$

In the above example, the network utilizations under routing scheme $F1R$, $F2R$, and $AR$ are 0.12, 0.15, and 0.18, respectively. The routing scheme $AR$ gets the highest net-

<table>
<thead>
<tr>
<th>List order</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAG node</td>
<td>$v_0$</td>
<td>$v_1$</td>
<td>$v_5$</td>
<td>$v_2$</td>
<td>$v_4$</td>
<td>$v_3$</td>
<td>$v_6$</td>
</tr>
</tbody>
</table>

![Graph](image-url)
work utilization as the schedule under scheme AR has smaller schedule span and more link usage.

5. Simulation Results
In this section, we evaluate the average performance of the extended list scheduling under different routing schemes for randomly generated DAGs. Three typical networks are used in the extensive simulations: 16-node network (Fig. 8(a), average node
degree is 2.125], NSFNET network [Fig. 8(b), average node degree is 3.125], and Mesh Torus network [Fig. 8(c), average node degree is 4]. The number on the links in Fig. 8 represents the length of the links in kilometers, and each link is bidirectional.

To generate DAG randomly, two fundamental characteristics of the DAG are considered: (1) the communication–computation ratio (CCR) and (2) the average number of edges per node. The CCR is defined as the sum of all communication costs divided by the sum of all computation costs. CCRs of 0.1, 1, and 10 are used to simulate low, medium, and high communication, respectively. For the average number of edges per vertex, we utilize two in the following simulation. For each CCR value, DAGs are generated with sizes of 10, 50, 250, and 1250 nodes. We use the same DAG generator in [15]. Every possible edge is created with the same probability, calculated based on the average number of edges per node. To obtain the desired CCR for a DAG, node weights are taken randomly from a uniform distribution [1,19] around 10, thus the average node weight is 10. Edge weights are also taken from a uniform distribution, whose mean value depends on the CCR and on the average number of edge weights. The relative deviation of the edge weights is identical to that of the node weights.

We made the following assumptions for simulation.

- Every optical switch is assumed to have a connection to only one resource.
- Each access link has only one unit bandwidth.
- All the tasks and resources are assumed to have three different types.
- Each scheduling result is the average over 100 simulations.
- We compare the joint scheduling performances under three different routing schemes: $F_1R$, $F_2R$, and $AR$ as described in Section 4.
- We define normalized schedule length (NSL) to evaluate the schedule performance under different routing schemes.

\[
\text{NSL}(X) = \frac{SL(X)}{SL(NR)}, \quad X \in \{F_1R, F_2R, AR\},
\]

where $\text{NSL}(X)$ is the normalized schedule length of routing scheme $X$, $SL(X)$ is the schedule length of routing scheme $X$, and $SL(NR)$ is the schedule length of no-routing scheme $NR$ under the same simulation conditions as $X$. The closer to 1 the normalized
schedule length, the better the scheduling performance. On the contrary, the larger than 1 the normalized schedule length, the worse the scheduling performance.

5.A. Impacts of Directed Acyclic Graph Size and Communication–Computation Ratio

In the first simulation, we study the impacts of DAG size (the number of task vertices in the DAG) and CCR on the schedule results under different routing schemes. The simulations are carried out over NSFNET network [see Fig. 8(b)]. It is assumed that each transmission link has only one unit bandwidth.

Figure 9 depicts the normalized scheduling length versus DAG size under different routing schemes, with CCR=0.1, 1, and 10. When CCR=0.1, we observe that three routing schemes have almost no difference in performance [see Fig. 9(a)]. It is because the load of the communication is low and the communication contention can be ignored in DAG scheduling. When CCR=1, the total amount of communications becomes equivalent with that of computations, so that the communication contention cannot be ignored in the scheduling. We find that the normalized schedule length of each routing scheme increases as DAG size increases [see Fig. 10(b)]. It can also be found that the three routing schemes have more difference in performance when the DAG size increases. AR is the best in performance, F2R is moderate, and F1R is the worst, as expected. When CCR=10, the load communication becomes intense, and then the normalized schedule lengths of all three routing schemes increase accordingly, as shown in Fig. 9(c). Hence, increasing the DAG size and CCR value provides more communication, and consequently results in more contentions in the scheduling.

Figure 10 plots the network utilization (defined in Section 4) of different routing schemes corresponding to Fig. 9. We can observe that the network utilization of each routing scheme and their difference in performance increase as the DAG size and CCR increase. AR still has the best performance among the three routing schemes. Another observation is that the network utilization increases more and more slowly when the DAG increases to a certain degree. The explanation for this is that the release of the resources occupied by the former scheduled tasks and the allocation of resources required by the new tasks have gradually come to balance.

Fig. 9. Normalized scheduling length versus DAG size with different CCR in NSFNET network. (a) CCR=0.1; (b) CCR=1.0; (c) CCR=10.

Fig. 10. Network resource utility versus DAG size with different CCR in NSFNET network. (a) CCR=0.1; (b) CCR=1; (c) CCR=10.
5.B. Impact of Different Network Topologies

In this simulation, the impacts of different network topologies on the schedule results are compared. We employ three different network topologies as shown in Fig. 8. It is assumed that each transmission link has only one unit bandwidth.

Figure 11 plots the normalized scheduling length versus DAG size with CCR=1 in different networks. Figure 12 plots the network utilization corresponding to Fig. 11. We can observe that the higher the average node degree of the network is, the lower the normalized schedule length and network utilization for each routing scheme becomes. This is because the network with higher average node degree has more abundant link resources that can relieve the link communication contention during the scheduling process. We also find that in the network with higher average node degree, the routing scheme AR performs better than F2R in both normalized schedule length and network utilization. The reason is that scheme AR can utilize more additional links to determine the route than scheme F2R due to the abundant link resources in the network with higher average node degree.

5.C. Impact of the Number of Unit Bandwidths per Transmission Link

In this simulation, we randomly generate DAGs with the size of 1250 vertices and CCR=1, and obtain the simulation results over the NSFNET topology. In Fig. 13, the scheduling results under the three routing schemes versus the number of unit bandwidths per transmission link are compared.

It can be observed that both the normalized schedule length and network utilization of each routing scheme decrease as the number of unit bandwidths per link increase. This is because increasing the bandwidth per transmission link is to increase the optical network resources and alleviate the resource contentions in the same link for each routing scheme. Therefore, more bandwidths per link lead to better scheduling performance and lower network utilization. To achieve better scheduling performance, routing scheme AR requires fewer bandwidths per link with higher network utilization. However, when the links contain abundant bandwidths, there will be no difference in scheduling results for all the routing schemes, which means only considering access link (i.e., access port) contention is feasible for the optical grid scheduling.
6. Conclusions

In this paper, a practical joint scheduling model for optical grid applications is proposed. The grid resources and the optical network interconnecting them are combined and modeled as an extended resource graph. The optical grid application is abstracted as a set of precedence-dependent tasks represented by a DAG. The joint scheduling for the O-Grid application then becomes a communication contention aware DAG scheduling problem. The contention awareness is achieved by scheduling the edges of the DAG onto the links of the extended resource graph in the similar way as the vertices of the DAG are scheduled onto the grid resources. Based on the optical grid scheduling model, we extend the classic list scheduling algorithm to implement the resource mapping from DAG to O-Grid extended resource system.

To improve the scheduling performance, we proposed an efficient adaptive routing scheme named earliest start route first by modifying the traditional Dijkstra shortest path first algorithm. As can be seen in the simulation part, the different optical network routing schemes do have impacts on the joint scheduling performance. Our proposed ESRF algorithm can do a better job of enhancing the scheduling performance compared with the fixed and alternative routing schemes, especially in the optical network with higher average node degree. We also observed that when there are adequate communication resources that can be utilized in the optical network, the scheduling results under different routing schemes have no difference, which leads to the fact that only considering the access link contention is feasible for the optical grid application scheduling.

The proposed extended list scheduling algorithm is a greedy algorithm that, at each step, selects the best choice without considering the future consequences. Hence, it may not produce the shortest schedule length all the time even using an adaptive routing scheme. Therefore we need to devise a more optimal and reliable scheduling algorithm.

Acknowledgments

This work is supported by the China 863 Program and the National Nature Science Foundation Committee of China under grants 60672016, 60502004, 60632010, and 60602010.

References


