Workflow Scheduling in e-Science Networks

Eun-Sung Jung
Samsung Advanced Institute of Technology, ICL Lab
Samsung Electronics
Yongin-si, 446-712, South Korea
Email: es73.jung@samsung.com

Sanjay Ranka and Sartaj Sahni
Department of CISE
University of Florida
Gainesville, FL 32611, USA
Email: {ranka,sahni}@cise.ufl.edu

***Abstract*—We solve workflow scheduling problems in e-Science networks, whose goal is minimizing either makespan or network resource consumption by jointly scheduling heterogeneous resources such as compute and network resources. We formulate the workflow scheduling problem incorporating multiple paths as a mixed integer linear programming (MILP) and develop several linear programming relaxation heuristics based on this formulation. Our algorithms allow dynamic multiple paths for data transfer between tasks and more flexible resource allocation that may vary over time. We evaluate our algorithms against a well-known list scheduling algorithm in e-Science networks whose size is relatively small. Our simulation results show that our heuristics are fast and work well when communication-to-computation ratios (CCRs) are small. Also, these results show that use of dynamic multiple paths and malleable resource allocation is useful for data intensive applications.**

Keywords: Workflow scheduling, e-Science, dynamic path

I. Introduction

The advances in communication, networking and computing technologies is dramatically changing the way scientific research is conducted. A new term, *e-Science*, has emerged to describe the "large-scale science carried out through distributed global collaborations enabled by networks, requiring access to very large-scale data collections, computing resources, and high-performance visualization" [1]. e-Science (and the related grid computing [2]) examples include high-energy nuclear physics (HEP), radio astronomy, geoscience and climate studies. To support e-Science activities, a new generation of high-speed research and education networks have been developed. These include Internet2 [3], the Department of Energy's ESnet [4], National Lambda Rail [5] etc. These networks carry a large amount of data traffic for e-Science applications. However, the size of the networks is very small.

Many e-Science applications can be effectively modeled as a Directed Acyclic Graph (DAG). The general form of DAG scheduling has been shown to be NP-hard [6], and a number of heuristics have been proposed. Early research regarded communication costs as small [7], [8] or assumed a very simple interconnection network model, i.e., a fully-connected network model without contention [9], [10], [11], [12], [13]. The heterogeneous-earliest-finish-time (HEFT) algorithm extended for heterogeneous computing resources was proposed in [13].

The extended list scheduling algorithms in [14] and [15] targeted for heterogeneous cluster architectures address this network contention issue by various priority attributing schemes. They assumed the path between any two processors is determined and fixed by the target system using conventional algorithms such as a breadth first search (BFS). In the literature of grid computing, similar work regarding DAG scheduling has been done. Generally, the term, *workflow*, is used interchangeably with DAG in the context of grid computing.

For e-Science applications, the amount of data that needs to be transferred between tasks may be of the order of hundreds of gigabytes to multiple terabytes. Thus, the key challenge is to be able to schedule a workflow such that the total execution time and the communication costs are minimized. Following the taxonomy of [16], the network resource mapping can have the following characteristics:

Rigid vs. malleable: Rigid mapping is fixed bandwidth mapping over the time period of data transfer whereas malleable mapping allows for variable bandwidth. If there is no quality of service requirements such as constant data rate, malleable mapping is a viable option to utilize network resources efficiently since solutions can be flexible as long as total amount of data transmission over time meets the data transmission requirement.

Single path vs. multiple paths: For many transfers, multiple paths can be effectively used to reduce the transfer time. However, finding a set of multiple paths requires more computation time, and thus efficient algorithms are needed.

Static vs. dynamic paths: In static mapping, paths determined at the start time of data transmission do not change until the end of data transmission, while in dynamic path mapping, paths can change dynamically over time.

The recent work on workflow scheduling in optical grids can provision network resources dynamically with guarantee of specified bandwidth [17], [18], [19], [20], [21]. According to the above taxonomy, all those methods use rigid and single path mapping. Moreover, the paths are assumed to be static. In contrast, we propose efficient algorithms allowing

malleable resource allocation using dynamic network paths.

The rest of the paper is organized as follows. We describe the workflow scheduling problem in Section II. We present the optimal problem formulation in Section III. We present the our LP relaxation algorithms in Section IV. We show experimental results in Section V and finally conclude with summary in Section VI.

II. Workflow Scheduling in e-Science Networks

A. System Model and Data Structure

1) Time Model: The uniform time slice model is represented by τ and M where τ is the size of a time slice and M is the maximum number of time slices the system would consider. The start and end time of the time slice m is denoted by T_m and T_{m+1}, respectively.

2) Network Resource Model: A network resource model is represented by $G_n = (\mathbf{V}, \mathbf{E}, \mathbf{r}, \mathbf{TR}, \mathbf{TB})$, where V and E are a set of nodes and a set of edges, respectively, r_v denotes the resource type of a node v, and TR_v and TB_e denote the data structures for the resource availability of node v and edge e, respectively, over time. More specifically, we use time-resource (TR) or time-bandwidth (TB) arrays as data structures for managing resource availability over time. A TR or TB array is a set of (a_m) where m is the index of a time slice and a_m is the available amount of a resource over the period $[T_m, T_{m+1})$. These data structures are necessary for effective in-advance reservation of resources. Basically, the data structures of a TR array and a TB array are same except the fact that the resource type represented by a TB array is only network resource type whereas other resource types are represented by a TR array. Thus, a TB array is assigned on each edge and a TR array is assigned on each node in a network resource graph.

Figure 1 shows an example of network resource graph. Each node represents one resource, and is associated with a resource type and a TR array, which tracks the resource availability over time. In Figure 1, nodes V_1 through V_3 are of resource type 1, and nodes V_4 and V_5 are of resource type 2. We can assign a unique number to a different resource type excluding the network resource. For example, resource type 1 is pure compute resource and resource type 2 is database service resource. Each edge represents a physical link connecting two nodes, and is associated with a TB array, which tracks the network resource availability over time. In this paper, we assume the uniform time slice model where the periods of all time slices are same and reasonably small compared to the makespans of workflows.

3) Workflow Model: A workflow can be represented by a task graph, which is a directed graph and formally defined as $G_t = (\mathbf{N}, \mathbf{L}, \mathbf{r}, \mathbf{R_N}, \mathbf{R_L}, ST, Deadline)$. $\mathbf{N}$ and $\mathbf{L}$ represent a node set and an edge set, respectively. r_i denotes the resource type of node N_i. R_{N_i} denotes the required amount of a resource of node N_i, and R_{L_i} denotes the required amount of data transfer between both end nodes of edge L_i. In this paper, we assume all capacities of resources are normalized with regard to the base capacity and we can then express the required or available amount of resources by rational numbers, multiples of the base capacity. ST is the start time of a workflow, which has to be taken into consideration for in-advance scheduling. $Deadline$ is an optional parameter. If it is given, we may set the optimization objective to minimizing network resource consumption. Otherwise, we may set the optimization objective to minimizing the makespan of the workflow. Figure 2 shows an example of a task graph. Resource requirement and resource type are associated with each node, and only network resource requirement property is associated with each edge.

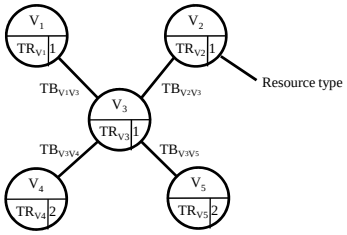

Figure 1. An example of a network resource graph

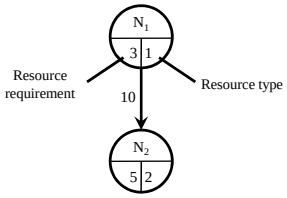

Figure 2. An example of a task graph

B. Construction of an Auxiliary Graph

In this paper, we translate the workflow scheduling problem into a network flow problem. The multicommodity flow problem, which optimizes the cost of multiple commodities with different source and destination nodes flowing through the network, is a well-known network flow problem. To formulate the workflow scheduling problem as a multicommodity flow problem, we first have to construct an auxiliary graph from the given network resource graph and task graph. The workflow scheduling problem is comprised of two mapping problems, a node mapping problem and an edge mapping problem onto a network resource graph. The goal of constructing an auxiliary graph is to convert a node mapping problem into an edge mapping problem since the multicommodity flow problem can deal with only an edge mapping problem.

An illustrative example of the auxiliary graph corresponding to Figures 1 and 2 is shown in Figure 3. An auxiliary graph $G_A = (V_A, E_A, TB_A)$ is constructed as follows. First, we expand the network resource graph by duplicating each node and connecting from the original one to the duplicated one. For convenience, let's call the original one a *frontend* node, and the duplicated one a *backend* node. For example,

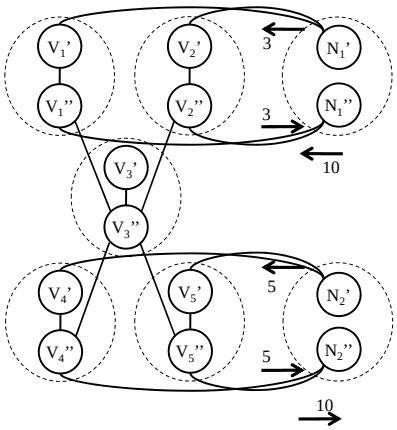

Figure 3. An example of an auxiliary graph

in Figure 3, the node V_1 is expanded into two nodes, V_1' and V_1'', and a new edge connecting these two nodes is inserted with the associated TB array corresponding to V_1's TR array. In this case, V_1' is a *frontend* node and V_1'' is a *backend* node. Obviously, this expansion is to convert a resource allocation problem into a network flow problem. The original topology of the network resource graph remains unchanged among the backend nodes of the expanded graph as in Figure 3. Note that some nodes of no chance of being selected may not need to be expanded. Second, we expand the task graph in the same way as we did the network resource graph. But we do not create any edge connecting nodes in the expanded task graph. Lastly, we interconnect the expanded network resource graph and the expanded task graph. The space complexity of an auxiliary graph is summarized as follows; $|V_A| = 2(|V| + |N|), |E_A| = |E| + |V| + 2|N||V|$.

III. MILP Formulation

The workflow scheduling problem can be formulated as a mixed integer linear programming (MILP) problem, which is a variant of a multicommodity flow problem. The objective of the MILP problem can be minimizing the finish time or minimizing the total network resource consumption depending on whether the deadlines for workflows are given or not. The constraints of the MILP problem are composed of four parts: 1) multi-commodity flow constraints, 2) task assignment constraints, 3) precedence constraints, and 4) deadline constraints. Since we have transformed the workflow scheduling problem into a multi-commodity problem, the typical multi-commodity flow constraints remain valid. Additional multi-commodity flow constraints are added to account for malleable resource allocation. Secondly, the task assignment constraints are integer constraints to enforce that one task is mapped to only one resource node in the network topology graph. Thirdly, the precedence constraints ensure that precedence constraints of a workflow are obeyed. Finally, the deadline constraints are just for the case when deadlines for workflows are given. The notation for the MILP formulation is listed in Table I.

Notation	Description
$pred(v)$	Returns the set of predecessors of node v
$\mathbf{J}$	$\{(s_j, d_j, F_j) \mid s_j, d_j \in V_A, 0 \leq j < \|\mathbf{N}\| + \|\mathbf{L}\|\}$ where s_j and d_j are source and destination nodes of job j and F_j is the required amount of flow (resource) for job j, a set of jobs. One job corresponds to either a node or an edge of a workflow.
$\mathbf{J_c}$	A set of communication jobs, $\mathbf{J_c} \subset \mathbf{J}$.
$\mathbf{J_{nc}}$	A set of non-communication jobs, $\mathbf{J_{nc}} \subset \mathbf{J}, \mathbf{J} = \mathbf{J_c} \cup \mathbf{J_{nc}}$.
$\mathbf{P_j}$	A set of allowed paths for job $j \in \mathbf{J}$ (from s_j to d_j).
G_A	An auxiliary graph, $(\mathbf{V_A}, \mathbf{E_A}, \mathbf{TB_{E_A}})$.
$\mathbf{E_{inf}}$	A set of edges with infinite available bandwidth $\subset \mathbf{E_A}$.
$b_{lk}(m)$	The available bandwidth on edge (l, k) during the time slice m in an auxiliary graph G_A.
T_m	The start time of time slice m.
T_{m+1}	The end time of time slice m.
M	The number of time slices to be considered.
N	The number of workflows in case of multiple workflow scheduling.
$\mathbf{Inc_v}$	The set of edges incident on node v.
D	Deadline of a workflow/workflow n.
WST	Start time of a workflow/workflow n.
Q	Very large number.
T_f	Finish time of a workflow/workflow n.
$f_{lk}^j(m)$	The amount of flow transferred (resource allocated) for job $j \in \mathbf{J}$ on link (l, k) during the time slice m for a workflow/workflow n. This variable is defined in case that the type of a job is same as that of the edge.
$f_p^j(m)$	The amount of flow transferred (resource allocated) for job $j \in \mathbf{J}$ on path $p \in \mathbf{P_j}$ during the time slice m for a workflow/workflow n.
x_{lk}^j	0 or 1. 1 if edge $(l, k) \in Inc_{s_j}$ is selected for job $j \in \mathbf{J}$ or 0 otherwise for a workflow/workflow n.
y_{lk}^j	0 or 1. 1 if edge $(l, k) \in Inc_{d_j}$ is selected for job $j \in \mathbf{J}$ or 0 otherwise for a workflow/workflow n.
z_m^j	0 or 1. 1 if job $j \in \mathbf{J}$ is allocated in time slice m or 0 otherwise for a workflow/workflow n.
ST_j	Start time of a job $j \in \mathbf{J}$ in a workflow/workflow n.
END_j	End time of a job $j \in \mathbf{J}$ in a workflow/workflow n.

Table I
Notation for problem formulation

The tasks and data transfers among them are all mapped into jobs in a multicommodity flow problem. A job in the formulation is described as a three-tuple (s, d, F), where s and d denote the source and destination nodes of the job, F denotes the required amount of flow (resource). ST and END, the start and end times of the job, are determined by workflow scheduling algorithms. The resource type does not have to be included in this tuple since a flow is forced to take a link of the same resource type due to the carefully chosen connect edges between a network resource graph and a task graph. Three kinds of binary decision variables are introduced; x, y and z. The discrete nature of the problem is due to the fact that a task cannot be split and we have discrete time intervals to accommodate jobs. binary decision variables, x_{lk}^j and y_{lk}^j, determine which resource is to be allocated to a non-split task. Regarding a job j corresponding to a task in a task graph, the flow of the job can take only one outgoing edge from the frontend node of a task and only one incoming edge into the backend node of a task in

Objective

$$\text{minimize } T_f \text{ or } \sum_{j\in J}\sum_{(l,k)\in \mathbf{E_A}}\sum_{m=0}^{M-1} f_{lk}^{j}(m) \quad (1)$$

Multi-commodity flow constraints

$$\sum_{k:(l,k)\in \mathbf{E_A}} f_{lk}^{j}(m) - \sum_{k:(k,l)\in \mathbf{E_A}} f_{kl}^{j}(m) = 0,$$
$$\forall j \in \mathbf{J}, \forall l \in \mathbf{V_A}, 0 \le m < M, l \ne s_j, l \ne d_j \quad (2)$$

$$\sum_{m=0}^{M-1}\Big(\sum_{k:(l,k)\in \mathbf{E_A}} f_{lk}^{j}(m) - \sum_{k:(k,l)\in \mathbf{E_A}} f_{kl}^{j}(m)\Big) = \begin{cases} F_j, & \text{if } l = s_j \\ -F_j, & \text{if } l = d_j \end{cases},$$
$$\forall l \in \mathbf{V_A}, \forall j \in \mathbf{J} \quad (3)$$

$$\sum_{j\in \mathbf{J}} f_{lk}^{j}(m) \le b_{lk}(m) \times (T_{m+1} - T_m),\ \forall (l,k) \in \mathbf{E_A}, 0 \le m < M \quad (4)$$

$$0 \le f_{lk}^{j}(m) \le b_{lk}(m) \times (T_{m+1} - T_m) \times z_m^j,$$
$$\forall l \in \mathbf{V_A}, \forall j \in \mathbf{J}, \forall (l,k) \in \mathbf{E_A}\backslash\mathbf{E_{inf}}, 0 \le m < M \quad (5)$$

$$ST_j \le T_m \times z_m^j + (1 - z_m^j)Q,\ \forall j \in \mathbf{J}, 0 \le m < M \quad (6)$$

$$END_j \ge T_{m+1} \times z_m^j,\ \forall j \in \mathbf{J}, 0 \le m < M \quad (7)$$

Task assignment constraints

$$\sum_{(l,k)\in \mathbf{Inc_{s_j}}, l = s_j} x_{lk}^{j} = 1, \forall j \in \mathbf{J} \quad (8)$$

$$\sum_{(l,k)\in \mathbf{Inc_{d_j}}, k = d_j} y_{lk}^{j} = 1, \forall j \in \mathbf{J} \quad (9)$$

$$\frac{\sum_{m=0}^{M-1} f_{lk}^{j}(m)}{Q} \le \begin{cases} x_{lk}^{j}, & \text{if } (l,k) \in \mathbf{Inc_{s_j}} \\ y_{lk}^{j}, & \text{if } (l,k) \in \mathbf{Inc_{d_j}} \end{cases}, \forall j \in \mathbf{J} \quad (10)$$

Precedence constraints

$$ST_j = WST, if\ pred(j) = \emptyset, \forall j \in \mathbf{J} \quad (11)$$
$$ST_j \le END_j, \forall j \in \mathbf{J} \quad (12)$$
$$END_p \le ST_j, \text{if } p \in pred(j),\ p, j \in \mathbf{J} \quad (13)$$
$$END_j \le T_f, \forall j \in \mathbf{J} \quad (14)$$
$$y_{lk}^{i} = x_{lk}^{j}, \text{if } i \in pred(j),\ i \in \mathbf{J_{nc}},\ j \in \mathbf{J_c} \quad (15)$$
$$y_{lk}^{i} = y_{lk}^{j}, \text{if } i \in pred(j),\ i \in \mathbf{J_c},\ j \in \mathbf{J_{nc}} \quad (16)$$

Deadline constraints (optional)

$$T_f \le D \quad (17)$$

Figure 4. Workflow scheduling problem formulation via network flow model

the auxiliary graph. These constraints reflect the non-split property of a task. z_m^j indicates whether time slice m is used for the job j or not. These binary decision variables can be easily extended to the multiple workflow scheduling problem by using separate variables for each workflow.

The complete formulation is presented in Figure 4. First of all, the problem can be optimized for either minimum finish time, T_f, or minimum network resource consumption, $\sum_{j\in J}\sum_{(l,k)\in \mathbf{E_A}}\sum_{m=0}^{M-1} f_{lk}^{j}(m)$, as in Expression 1. Minimizing network resource consumption can be helpful for saving more resources for future arriving requests so that more requests can be accepted in the long term.

IV. LP Relaxation

As you will see in the experimental results, the running time of MILP for the workflow scheduling increases exponentially as the number of nodes of a workflow grows. The general workaround to solve the MILP problem fast enough to be useful in practical is the linear programming relaxation by transforming binary variables into real variables ranging between 0 and 1. We can turn the solution to the linear programming relaxation of the MILP problem into the approximate solution to the MILP problem via techniques such as rounding. In this paper, we propose a LP relaxation (LPR) algorithm, consisting of two steps, for the workflow scheduling problem.

- We determine which resources are selected for the tasks (nodes) of a task graph.
- The next step is to iteratively determine the start and end times of jobs along with network resource allocations for data transfer jobs.

With the solution of the first-step algorithm, we can determine the start and end times of jobs by solving small MILP problems iteratively, regarding unscheduled jobs. The basic idea is that finding a solution to the MILP problem with determined x/y binary variables and undetermined z binary variables for a small number of jobs, e.g., 3, takes little time. Thus, we can divide the problem into many small problems and solve them sequentially. To pick appropriate jobs, we also use same bottom level priority scheme as the heuristic. However, in our case, the node mapping is already determined.

A. Enhancing time complexity

There exist two kinds of LP formulations for the multicommodity flow problem, node-arc form and edge-path form. The MILP formulation in Figure 4 takes the node-arc form, which assigns a separate decision variable for a certain job on a certain link. In contrast, the edge-path form assigns a separate decision variable for a certain job on a certain path in a set of paths, P, which the job can take. Accordingly, if we limit the number of paths in the set P, we can reduce the number of decision variables, which leads to better performance in terms of time complexity by sacrificing the accuracy of the solution. We will refer to this edge-path based LP relaxation as LPREdge for the rest of the paper.

V. Experimental Evaluation

A. Experiment Setup

We compare the performance of four algorithms, the optimal MILP algorithm, the LP relaxation algorithm, the edge-path form LP relaxation algorithm and the list scheduling heuristic in terms of the makespan, i.e., the schedule length of workflows and the computational time of algorithms. In the following, we refer to the MILP algorithm, the LP relaxation algorithm, the edge-path form LP relaxation algorithm, and the general extended list scheduling algorithm as MILP, LPR, LPREdge and LS, respectively.

We first compare the performance of all four algorithms with regard to workflows with a small number of nodes, 3. This experiment is for comparison of non-optimal algorithms against the optimal algorithm. We then compare two algorithms, LPREdge and LS, with regard to workflows with

a larger number of nodes ranging from 10 to 50 with an increment of 10. Note that the size of e-Science networks is small and accordingly the size of workflows is usually small. The second experiment is to verify that our algorithm performs better than the heuristic algorithm in terms of makespan.

As a network resource graph, we use the Abilene network [23] (see Figure 5), which is deployed in practice. The resource capacities of nodes of the network resource graph as well as the bandwidth capacities of edges are randomly selected from a uniform distribution between 10 to 1024. In this paper, we use a random workflow generation method that depends on three parameters: the number of nodes, the average degree of nodes and communication-to-computation ratio (CCR). The number of nodes is varied according to the aforementioned experiments. The average degree of nodes is related to the level of parallelism of workflows and fixed to 2. The different CCRs of 0.1, 1, and 10 are used to assess the impact of the communication factor on the performance of the algorithms. A larger CCR means a workflow is more data-intensive. The weights of nodes of a workflow are randomly selected from a uniform distribution between 10 to 1024 as the resource capacities of the Abilene network are determined. Subsequently, the weights of edges of a workflow are set to the CCR times the uniform distribution between 10 to 1024. One hundred trials were for every combination of workflow parameters, the number of nodes, CCR and the chosen algorithm. We then averaged the results and plotted charts for performance evaluation.

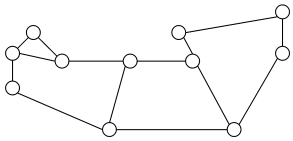

Figure 5. The Abilene network

B. Results

We evaluate the performance of workflow scheduling algorithms with regard to two metrics: schedule length of workflows, i.e., makespan, and computational (running) time. The detailed results with explanation are presented in this subsection.

1) Schedule Length of Workflows:

Comparison against optimal scheduling results: Since the optimal schedules for randomly generated workflows on the given network resource graph, the Abilene network, are not known ahead of time, the only way of evaluating the makespans of non-optimal algorithms is to compare makespans of those algorithms against the optimal algorithm.

In Figure 6, we can see that the performance of non-optimal algorithms, i.e., LPR, LPREdge, and LS, is comparable to the optimal algorithm, MILP, when CCR = 0.1 and 1.0. However, as CCR grows up to 10, the makespan of LS becomes roughly 2 times of the makespan of MILP. In contrast, the makespan of LPR and LPREdge is at most 20% more than the optimal makespan.

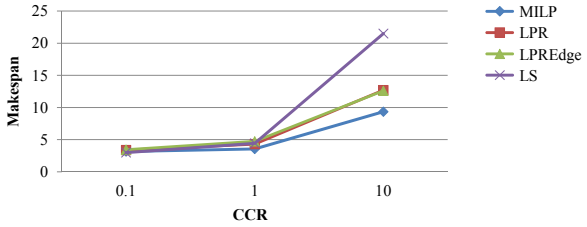

Figure 6. Makespan vs. CCR for all algorithm when the number of nodes in a workflow is 3 in the Abilene network

Comparison between LPREdge and LS: As the general workflow scheduling problem is a NP-hard, our corresponding formulation, MILP, requires exponential computational time as the size of workflows increases. For large workflows, it is impractical to determine the optimal makespan using the MILP algorithm. For this reason, we compare the makespans of only our non-optimal algorithms, LPREdge and LS, in Figure 7. We can see that LPREdge is much better than LS. It achieves half the makespan of LS in some cases.

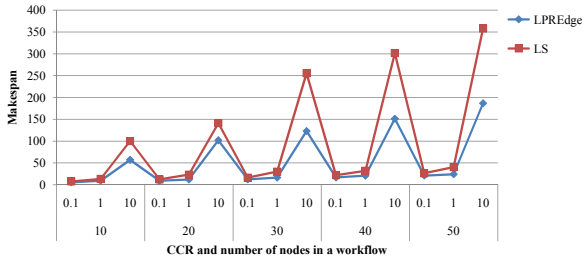

Figure 7. Makespan vs. CCR and the number of nodes in a workflow for LPREdge and LS in the Abilene network

2) Computational Time:

Comparison against optimal scheduling results: The running time of the optimal algorithm grows exponentially as shown in Figure 8. This algorithm takes approximately 14 seconds when there are 3 nodes and CCR = 0.1. With 3 nodes, the run time becomes approximately 47 seconds when CCR = 10. When the number of nodes is increased to 10 and CCR = 0.1, MILP takes more than 1,500 seconds. By contrast, LPREdge takes less than 5 seconds when there are 3 nodes and less than 150 seconds when the number of nodes is less than 50 (Figure 9).

Comparison between LPREdge and LS: The running time of the heuristic is a few seconds whereas the running time of LPREdge is linearly increasing up to 150 seconds when the number of nodes is 50.

If requests for workflow scheduling from users are on-demand and should be handled real-time, the computational time of the fast greedy algorithm shown in the experiments is not positive. However, when the requests are in-advance, there is enough time between request arrival time and request

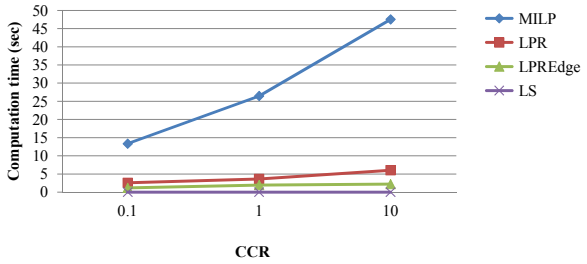

Figure 8. Computational time vs. CCR and the number of nodes in a workflow for all algorithms in the Abilene network

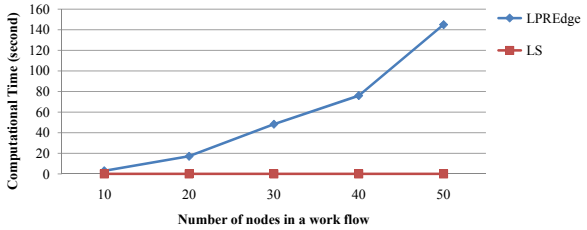

Figure 9. Computational time vs. CCR and the number of nodes in a workflow for LPREdge and LS in the Abilene network

start time, and the centralized server is a more high-end machine, LPREdge is deployable in practice.

VI. Conclusions

We have formulated workflow scheduling problems in e-Science networks. The formulations are different from previous work in the sense that they allow dynamic multiple paths for data transfer between tasks and more flexible resource allocation that may vary over time. The computation time of the optimal formulation increases exponentially with regard to the size of a workflow. Accordingly, the LP relaxation algorithm, referred to as LPR, for deployment in practice has been developed based on the optimal algorithm through the common linear relaxation technique. We also propose the edge-path form LP relaxation algorithm, LPREdge, to enhance time complexity.

The experimental results show that the makespan of LPR and LPREdge is comparable, less than 20% longer, to that of the optimal algorithm regardless of CCR for small workflows. In contrast, the general list scheduling algorithm, LS, performs roughly similar to LPR and LPREdge when CCR = 0.1, but the performance gap of LPR/LPREdge and LS grows dramatically as CCR grows from 1 to 10. Data-intensive workflow scheduling, which is common in e-Science application, can benefit from dynamic multiple paths and malleable resource allocation.

To the best of our knowledge, the optimal algorithm, the MILP formulation, is the first algorithm that jointly schedules heterogeneous resources including network resources using dynamic multiple network paths and malleable resource allocation. The approximation based on the optimal algorithm achieves the reasonable performance compared with the optimal algorithm in terms of schedule length (makespan). The application of these results to optical networks will be future work.

References

[1] The U.K. Research Councils, http://www.research-councils.ac.uk/escience/.

[2] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[3] Internet2, http://www.internet2.edu.

[4] Energy Science Network (ESnet), http://www.es.net.

[5] National Lambda Rail, http://www.nlr.net.

[6] O. Sinnen, *Task scheduling for parallel systems*. Hoboken N.J.: Wiley-Interscience, 2007.

[7] T. L. Adam, K. M. Chandy, and J. R. Dickson, "A comparison of list schedules for parallel processing systems," *ACM Commun.*, vol. 17, no. 12, pp. 685–690, 1974.

[8] E. G. Coffman and R. L. Graham, "Optimal scheduling for two-processor systems," *Acta Informatica*, vol. 1, no. 3, pp. 200–213, 1972.

[9] A. Khan, C. Mccreary, and M. Jones, "A comparison of multiprocessor scheduling heuristics," in *International Conference on Parallel Processing*, vol. 2, 1994, pp. 243–250.

[10] M. Palis, J. Liou, and D. Wei, "Task clustering and scheduling for distributed memory parallel architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, no. 1, pp. 46–55, 1996.

[11] L. Wang, H. J. Siegel, V. P. Roychowdhury, and A. A. Maciejewski, "Task matching and scheduling in heterogeneous computing environments using a Genetic-Algorithm-Based approach," *Journal of Parallel and Distributed Computing*, vol. 47, no. 1, pp. 8–22, Nov. 1997.

[12] Y. Kwok and I. Ahmad, "Benchmarking the task graph scheduling algorithms," in *IPPS/SPDP*, 1998, pp. 531–537.

[13] H. Topcuoglu, S. Hariri, and M. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

[14] O. Sinnen and L. Sousa, "List scheduling: extension for contention awareness and evaluation of node priorities for heterogeneous cluster architectures," *Parallel Computing*, vol. 30, no. 1, pp. 81–101, 2004.

[15] ——, "Communication contention in task scheduling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 6, pp. 503 – 515, june 2005.

[16] M. Wieczorek, A. Hoheisel, and R. Prodan, "Taxonomies of the Multi-Criteria grid workflow scheduling problem," in *Grid Middleware and Services*, 2008, pp. 237–264.

[17] Y. Wang, Y. Jin, W. Guo, W. Sun, W. Hu, and M. Wu, "Joint scheduling for optical grid applications," *Journal of Optical Networking*, vol. 6, no. 3, pp. 304–318, Mar. 2007.

[18] X. Liu, W. Wei, C. Qiao, T. Wang, W. Hu, W. Guo, and M. Wu, "Task scheduling and lightpath establishment in optical grids," in *INFOCOM*, 2008, pp. 1966–1974.

[19] X. Liu, "Application-Specific, agile and private (ASAP) platforms for federated computing services over WDM networks," Ph.D. dissertation, The State Universify of New York at Buffalo, Aug. 2009.

[20] X. Luo and B. Wang, "Integrated scheduling of grid applications in WDM optical Light-Trail networks," *Journal of Lightwave Technology*, vol. 27, no. 12, pp. 1785–1795, Jun. 2009.

[21] Z. Sun, W. Guo, Z. Wang, Y. Jin, W. Sun, W. Hu, and C. Qiao, "Scheduling algorithm for Workflow-Based applications in optical grid," *Journal of Lightwave Technology*, vol. 26, no. 17, pp. 3011–3020, 2008.

[22] K. Rajah, S. Ranka, and Y. Xia, "Scheduling bulk file transfers with start and end times," in *Sixth IEEE International Symposium on Network Computing and Applications*, 2007, pp. 295–298. [Online]. Available: 10.1109/NCA.2007.39

[23] "Abilene," http://abilene.internet2.edu/.

[24] A. Benoit, M. Hakem, and Y. Robert, "Contention awareness and fault-tolerant scheduling for precedence constrained tasks in heterogeneous systems," *Parallel Computing*, vol. 35, no. 2, pp. 83–108, Feb. 2009.

[25] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: task graphs for free," in *Proceedings of the 6th international workshop on Hardware/software codesign*. Seattle, Washington, United States: IEEE Computer Society, 1998, pp. 97–101.