Computer Networks 108 (2016) 184–198

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Dynamic embedding of workflow requests for bandwidth efficiency in data centers[☆]

Tram Truong-Huu[a,*], Mohan Gurusamy[a], Vishal Girisagar[b]

[a] *Department of Electrical and Computer Engineering, National University of Singapore, 4 Engineering Drive 3, Singapore 117583, Singapore*
[b] *Continental Automotive Singapore Pte Ltd, 80 Boon Keng Rd, Continental Building, Singapore 339780, Singapore*

ARTICLE INFO

Article history:
Received 11 March 2016
Revised 15 July 2016
Accepted 18 August 2016
Available online 20 August 2016

Keywords:
Workflows
Workflow embedding
Cloud data centers
Bandwidth minimization
Critical path
WDM-based optical networks

ABSTRACT

In this paper, we address the problem of embedding dynamically-arriving workflow requests in data centers. Workflows pose challenges due to data precedence and time disjointness among tasks, thus driving the need for intelligent methods to embed workflows in data centers while improving the bandwidth efficiency as well as guaranteeing the application performance. We first formulate an integer programming optimization model for the embedding problem that minimizes the amount of bandwidth required for workflow execution. We then develop two algorithms namely Critical Path Workflow Embedding (CPWE) and Edge Priority Workflow Embedding (EPWE) to solve this problem. We consider two data center network architectures: packet switching electrical networks and circuit switching optical wavelength division multiplexed (WDM) networks. While WDM-based optical networks have much larger bandwidth capacity to meet the ever-growing traffic demand in data centers, they pose challenges due to wavelength continuity constraint and the nature of circuit switching. We thus additionally propose methods for selecting appropriate Top-of-the-Rack (ToR) switches and wavelengths during the embedding process so as to increase the chance of accommodating many requests that span over multiple ToRs. We evaluate CPWE and EPWE through comprehensive simulations. The results show that CPWE and EPWE significantly reduce the bandwidth required for a workflow request by up to 66% for random workflows and 80% for realistic-application workflows compared to baseline algorithms. The results also show that the proposed methods for ToR selection and wavelength selection in optical data centers outperform other methods by reducing the rejection ratio by up to 47% with dynamic reconfiguration of lightpaths and 40% with incremental configuration of lightpaths.

1. Introduction

Cloud data centers have become an attractive candidate to meet resource demands of individual users and institutions. Instead of setting up a local infrastructure that costs a lot for device purchases and maintenance services, users nowadays are exploiting public clouds, which provide immense computing capacity, quasi-unlimited storage space and broad access network connections. Many large-scale and data-intensive applications have been migrated to the cloud to be able to handle a big amount of data as well as heavy computations. These applications generate a huge traffic demands mainly remaining within data centers as reported in [2]. This huge traffic demands force data centers look for the means for not only increasing the network capacity but also improving network resource utilization. While electrical packet-switched networks still keep their important role, using WDM-based optical networks in data centers is becoming a trend to meet the network capacity challenge. In comparison with electrical packet-switched networks, WDM-based optical networks have much larger bandwidth capacity with low power consumption and cabling complexity. Dynamic reconfiguration of lightpaths also brings in the flexibility for network management and traffic engineering. However, in addition to the limitations of circuit-switching, the wavelength continuity constraint due to the dynamic arrival of resource requests and high degree of the optical switch, which interconnects multiple ToR switches, has high impact on resource usage efficiency, making the problem of request embedding more challenging. Thus, intelligent methods for ToR and wavelength selection are needed to embed resource requests so as to use the servers under ToRs and bandwidth (or wavelength) resources between ToRs efficiently.

[☆] This article is an extended version of the paper published in ICDCN 2016 [1] that was carried out when Vishal Girisagar was affiliated with NUS.
[*] Corresponding author.
E-mail addresses: eletht@nus.edu.sg (T. Truong-Huu), elegm@nus.edu.sg (M. Gurusamy), vishalgirisagar801@gmail.com (V. Girisagar).

http://dx.doi.org/10.1016/j.comnet.2016.08.018
1389-1286/© 2016 Elsevier B.V. All rights reserved.

Considering both electrical and optical network data centers, in this paper, we address the problem of embedding workflow requests, which usually require large amount of computing resources and guaranteed bandwidth for applications such as detecting gravitational-waves [3], weather forecasting [4], predicting earthquakes [5]. A workflow resource request is represented by a set of computing tasks and a set of edges that represent the data dependencies among the tasks. Each computing task requires a number of virtual machines (VMs) to run the application for a certain duration and each edge requires a specific network bandwidth for data transmission between the VMs allocated to dependant tasks. Unlike the traditional virtual network requests [6], in which all requested resources including bandwidth and virtual machines are needed during the entire lifetime of the request, workflow requests have specific resource requirements at different times. Due to the task dependencies, some workflow tasks can execute concurrently while others have to run sequentially. VMs and bandwidth requested for a task may not be utilized before its preceding tasks finish their execution. Furthermore, due to large resource demands, workflow tasks are likely to be embedded across multiple ToRs of data centers, making the impact of the wavelength continuity constraint more pronounced in optical data centers. Thus, embedding of workflow requests in optical data centers needs to consider not only the lifetime of the tasks but also the ToR and wavelength selection to ensure high connectivity between ToRs with wavelength-continuous paths.

Given a workflow resource request, cloud providers need to embed this request in physical servers in data centers, considering the capacity of physical servers and network links as well as the network topology. The embedding process consists of allocating VMs on physical servers under ToR switches and reserving bandwidth on the links of the data center to guarantee the performance of workflow applications. While the amount of computing resources cannot be optimized, i.e., the number of VMs requested needs to be always satisfied, the bandwidth consumption for workflow execution is affected by the embedding solution. If a physical server has sufficient VMs to host all the requested VMs of two adjacent tasks in the workflow, then the execution of these two tasks will not consume link bandwidth. Otherwise, a certain amount of bandwidth of the link connecting two physical servers hosting the VMs of the two tasks needs to be reserved. Since the network bandwidth is limited, minimizing the bandwidth consumption during the execution of a workflow allows providers to accept more workflow resource requests, thereby increasing the revenue from the users who pay for resource usage. An intelligent embedding technique is therefore needed to help providers optimize the resource utilization in data centers.

Since workflow requests are different from the traditional virtual network requests due to data precedence and time disjointness among workflow tasks, existing virtual network embedding algorithms such as those presented in [7,8] are no longer applicable for embedding workflow requests in data centers. In this paper, we present a novel model for embedding workflow resource requests in data centers, considering the above challenges. We first formulate an integer programming optimization problem, which aims at minimizing the amount of bandwidth required for the execution of a workflow while guaranteeing its computing and network resource demands. Solving such an optimization problem and its variations has been shown to be computationally hard due to their $\mathcal{NP}$-complete nature [9]. We therefore develop two heuristic algorithms namely Critical Path Workflow Embedding (CPWE) and Edge Priority Workflow Embedding (EPWE) to solve the embedding problem efficiently.

Algorithm CPWE tries to embed all the VMs requested by the tasks on the *critical path* of the workflow on to the same physical server. We define the critical path as the path from the entry task to the exit task of the workflow for which the total bandwidth consumed on the links is the highest. Algorithm CPWE therefore needs an initial phase that determines the critical path before doing the embedding. Algorithm EPWE is computationally simpler than CPWE. It first sorts the edges in the workflow in the descending order of bandwidth requirement, and then starts embedding the tasks following the specified order. In both cases, it may not be successful to embed the entire critical path or the edge with the highest bandwidth requirement on the same physical server. Algorithms CPWE and EPWE then try to embed the requests in different servers so as to reduce the bandwidth consumption. To address the specific challenges of optical networks, we propose a method for ToR selection based on a connectivity-index function, which defines how well a ToR is connected with other ToRs with free wavelength-continuous paths so as to increase the chance of accommodating many future requests that span over multiple ToRs. We also propose a function that computes the goodness value for each wavelength between a pair of ToRs to choose the best wavelength for a lightpath. These two methods are integrated into CPWE and EPWE when realizing workflow request embedding in optical data centers. We evaluate the performance of the proposed algorithms through comprehensive simulations and compare their performance against baseline algorithms to demonstrate their effectiveness.

The rest of the paper is organized as follows. We discuss the related works in Section 2. We present the system model and mathematical formulations in Section 3. We present the proposed algorithms in Section 4. We present the methods for ToR and wavelength selection in Section 5. We carry out performance study and analyze the simulation results in Section 6 before concluding the paper in Section 7.

2. Related work

2.1. Workflow execution in clouds

Recently, research on workflows has received significant attention specially for the complex applications, which require large amount of data and computationally complex resources. Various systems such as MOTEUR [10] and Kepler [11] are used to interpret the workflow applications and submit the workflow tasks to a computing infrastructure for execution. These systems request only the computing resources for the workflows but they do not consider the bandwidth requirements between the dependent tasks in a workflow. In [12], the authors focused on cost minimization for embedding workflow application on data centers but they also did not consider bandwidth requirements. In [13], the authors aimed at minimizing the amount of computing resources, i.e., the number of VMs needed to execute workflows while ensuring their execution deadline. Depending on the number of remaining tasks and the available time before the deadline, the algorithm dynamically scales the number of VMs: increasing the number of VMs to meet the deadline or decreasing the number of VMs to reduce the usage cost. This work assumed that the required bandwidth is available on the links. This may not be practical in data centers since multiple tenant requests arrive at the same time and share the residual bandwidth on the links.

2.2. Workflow embedding and scheduling

Workflow embedding and scheduling has received significant attention from research communities [12,14–17]. In [16], the authors presented the *Myopic* algorithm that maps each individual task of the workflow on a first suitable computing resource. This simple algorithm did not consider the dependencies between workflow tasks, leading to high bandwidth consumption. Three heuristic algorithms have also been proposed in the literature [14]. These algorithms schedule workflow tasks based on their priority, e.g., execution deadline. The high priority task will then be mapped first on available computing resources. Yet, these algorithms did not consider the bandwidth consumption for data transmission between tasks. A genetic algorithm has been proposed in [18] to solve the problem of workflow scheduling within the deadline and execution cost constraints. A cost fitness function and a time fitness function have been designed respectively to compute the performance of the scheduling solution. While the cost fitness function considers only the cost of computing resources, the time fitness function considers only the execution time of workflow tasks on computing resources. The data transmission time and cost of bandwidth are ignored in this work, which is the key feature of our proposal.

In our recent work [1], we carried out initial study on the problem of workflow embedding in electrical data centers to minimize bandwidth consumption. We extend this work further to study the embedding problem in both electrical and optical data centers. Embedding workflow requests to optical data centers needs to deal with specific characteristic of optical networks such as the wavelength continuity constraint and wavelength selection.

2.3. Virtual network embedding (VNE)

Embedding of VM groups with bandwidth requests and virtual network requests in data centers has been studied in the literature [19–21]. Both works presented in [19] and [20] studied the problem of VM placement in data centers to reduce cross network traffic by embedding communicating VMs in the same server or rack. In [21], the problem of VM placement and bandwidth reduction for an application graph of VM clusters has been studied. In [8], the authors addressed the problem of allocation of computing and network resources in multi-tenant data centers with the assumption of knowledge of VM to VM bandwidth requirements, represented as a traffic matrix. These works considered fixed amount of bandwidth requirement between VMs for the entire lifetime of the request whereas workflows have different bandwidth requirements at different times. To the best of our knowledge, the problem of embedding workflow requests considering both VM placement and guaranteed bandwidth has not been addressed before. We try to minimize the bandwidth allocated for the requests while guaranteeing the availability of VMs and bandwidth for the application performance.

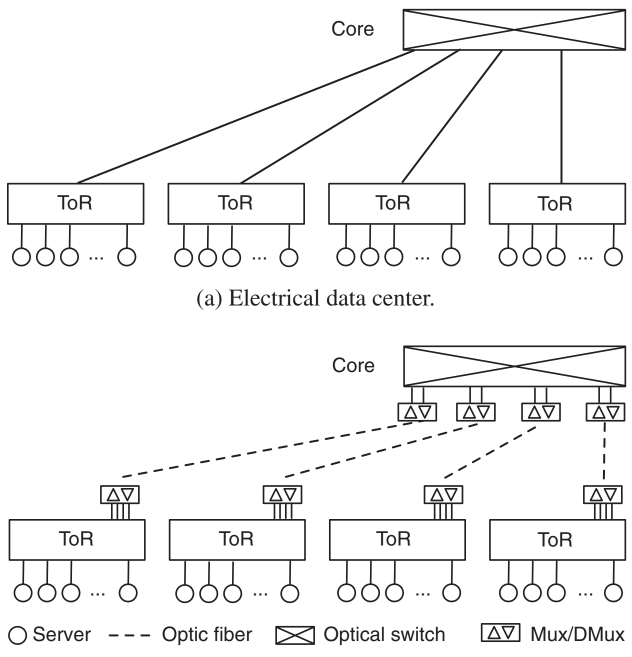
Virtual network embedding in optical networks has been recently studied [6,22–24]. While [22] and [23] studied the VNE problem in all-optical networks but not data centers, [6] and [24] considered this problem in hybrid optical-electrical data centers. The work presented in [24] additionally considered the capability of dynamic wavelength grouping of optical network architecture. These works however do not consider the complex resource requests like workflow resource requests, which is the focus of our work. Our work also differs from the works presented in [25,26], which focused more on the architecture design of optical networks rather than on the VNE problem.

3. System model

We now present the system model of our work. We first introduce the architecture of the data centers where our model can be applied. We then describe the workflow resource request model before we present the optimization problem to minimize the amount of bandwidth required for workflow execution and to improve the bandwidth efficiency in data centers. All the mathematical notations used throughout the paper are summarized in Table 1.

Table 1
Mathematical notations.

Notation	Description
W	Number of wavelengths carried in a fiber
M	Number of ToR switches in the data center
N	Number of physical servers connected to a ToR switch
$\mathcal{C}^{vm}$	Number of VMs can be hosted in a server
$\mathcal{C}^{eband}$	Capacity of server-ToR links
$\mathcal{C}^{cband}$	Capacity of a lightpath in the optical data center or ToR-core links in the electrical data center
$\mathcal{C}^{vm}_{i,j,t}$	Number of VMs available at time slot t on server j connected to ToR switch i
$\mathcal{C}^{cband}_{i,t}$	Residual bandwidth of the ToR-core link connecting ToR switch i to the core switch at time slot t in the electrical data center
$\mathcal{C}^{cband}_{i,i',t}$	Residual bandwidth of the lightpath connecting ToR switch i and ToR switch i' at time slot t in the optical data center
$\mathcal{C}^{eband}_{i,j,t}$	Residual bandwidth of the server-ToR link connecting ToR switch i and server j at time slot t
R^{vm}_{k}	Number of VMs requested by task k
D_k	Execution time of task k defined as multiple time slots
$R^{band}_{k,k'}$	Required bandwidth for the edge between task k and k'
$\mathcal{D}(K \times T)$	Resource usage duration of workflow tasks
$\mathcal{M}$	Embedding solution defined as a matrix $\mathcal{M}(K \times M \times N)$
$\mathcal{Q}_0(\mathcal{M})$	Total bandwidth consumed on server-ToR links for a workflow request
$\mathcal{Q}_1(\mathcal{M})$	Total bandwidth consumed on ToR-core links or lightpaths for a workflow request
$\mathcal{Q}(\mathcal{M})$	Total bandwidth consumed for a workflow



(a) Electrical data center.

(b) Optical data center.

Fig. 1. Two-tier data center architecture.

3.1. Data center architecture and resources

In this paper, we consider a two-tier data center architecture as shown in Fig. 1 where Fig. 1a presents the architecture of a packet-switched electrical data center and Fig. 1b presents that of a circuit-switched optical data center. In both architectures, the bottom most level contains physical servers that host VMs. The cost for communication between the VMs in the same physical server is zero as they do not need link bandwidth. Multiple physical servers are connected to a ToR switch by server-ToR links, forming a Performance Optimized Data center. The difference between the electrical data center and optical data center is the connection

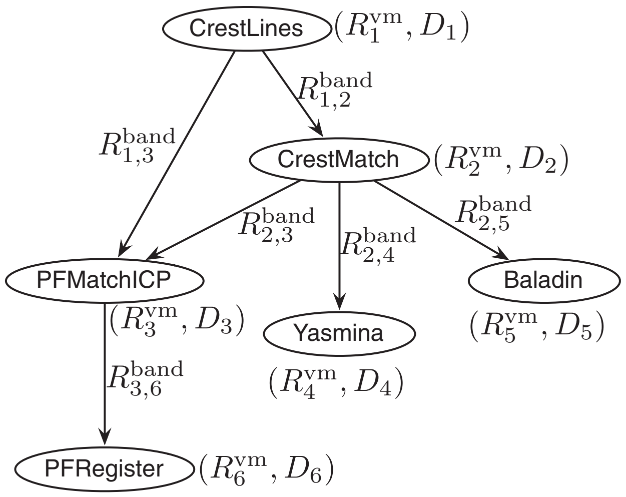


Fig. 2. Bronze Standard workflow and its resource request model.

between ToRs and the core switch. In the electrical data center, the ToRs are connected to the core packet switch by the ToR-core links while in the optical data center, the ToRs are connected to the core (optical) switch in the next level by multi-wavelength links. Each fiber can carry up to W wavelengths, meaning that a ToR switch can simultaneously reach up to W other ToR switches through optical paths or lightpaths. In the absence of wavelength converters, the lightpath between two ToR switches traversing through the optical switch needs to be wavelength-continuous. If VMs, which are hosted in different physical servers in the same rack, communicate with each other, the communication cost will incur only on server-ToR links. If VMs, which are hosted on different racks, communicate with each other, the communication cost will incur on both server-ToR links, and ToR-core links or lightpaths.

Let M denote the number of ToR switches in a data center. Each ToR switch can connect a maximum of N servers. For each server, up to $\mathcal{C}^{\rm vm}$ VMs can be deployed to satisfy the user's requirement. Thus, the total computing capacity of the data center is $M \times N \times \mathcal{C}^{\rm vm}$ VMs. We assume that all servers are homogeneous and users request for the same type of VMs for their applications. On the network resources, we also assume that all the links in the same level have the same capacity. The capacity of server-ToR links is $\mathcal{C}^{\rm eband}$ units and that of ToR-core links or lightpaths is $\mathcal{C}^{\rm cband}$ units. The maximum capacity of a fiber in the optical data center is then $W \times \mathcal{C}^{\rm cband}$.

We discretize the time horizon of the resource request into identically sized slots. At time slot t, let $\mathcal{C}_{i,j,t}^{\rm vm}$ denote the number of VMs available on server j connected to ToR switch i. For the electrical data center, we denote $\mathcal{C}_{i,t}^{\rm cband}$ as the residual bandwidth of the ToR-core link connecting ToR switch i to the core switch at time slot t. For the optical data center, we denote $\mathcal{C}_{i,i',t}^{\rm cband}$ as the residual bandwidth of the lightpath connecting ToR switch i and ToR switch i' at time slot t. Finally, the residual bandwidth on the server-ToR link connecting ToR switch i and server j at time slot t is denoted as $\mathcal{C}_{i,j,t}^{\rm eband}$. Initially, at time slot $t = 0$, $\mathcal{C}_{i,j,0}^{\rm vm} = \mathcal{C}^{\rm vm}$, $\mathcal{C}_{i,0}^{\rm cband} = \mathcal{C}^{\rm cband}$ and $\mathcal{C}_{i,j,0}^{\rm eband} = \mathcal{C}^{\rm eband}$ with $i = 1 \ldots M$ and $j = 1 \ldots N$.

3.2. Workflow resource request model

A workflow application is defined through a workflow graph, which is usually a Directed Acyclic Graph (DAG), featuring the application services to be executed (workflow tasks) and the data dependencies between these services (edges). In Fig. 2, we present an example of a workflow that is a medical image processing application, namely Bronze Standard [27]. It includes 6 image processing algorithms with names as shown in the figure. Based on the amount of input data, e.g., the number of images to be processed for the Bronze Standard application, users need to reserve cloud resources. For task k, let $R_k^{\rm vm}$ denote the number of VMs requested for its execution for a duration denoted as D_k. We assume that the execution time of workflow tasks is specified as multiple time slots. If there exists a data dependency between task k and task k', i.e., output of task k is input of task k', let $R_{k,k'}^{\rm band}$ denote the amount of bandwidth required between each VM pair of task k and task k'. Based on the resource requests and the dependencies between tasks of the workflow, we construct matrix $\mathcal{D}(K \times T)$ representing the resource usage during the entire execution of the workflow where K is the number of tasks in the workflow and T is the workflow *makespan*, which is defined as the total number of time slots needed to complete its execution. The value of $\mathcal{D}_{k,t}$ is set to 1 if task k needs resources in time slot t and $\mathcal{D}_{k,t}$ is set to 0 otherwise.

The construction of matrix $\mathcal{D}(K \times T)$ is based only on the application logic and the execution time of workflow tasks. Due to the data precedence among workflow tasks, the application logic needs to be known in advance to determine which tasks require resources first to execute and produce data for other tasks. The execution time of workflow tasks allows the users to determine the resource usage duration to request from cloud data centers. A conventional construction guarantees that there will be no delay in the execution of workflow tasks. A task will be scheduled immediately one after the other as soon as its input data is available, i.e., its preceding tasks have finished. We refer this construction to as a fixed construction.

Practically, a user may have his own time constraint, i.e., the processing deadline, workflow tasks thus do not have to be scheduled immediately one after the other as long as the time constraint is still preserved. In such a case, the construction of matrix $\mathcal{D}(K \times T)$ additionally considers the time constraint by which a task can be deferred by one or more time slots. Thus, we can have multiple variants of matrix $\mathcal{D}(K \times T)$ to represent the resource usage times of all tasks. We refer this construction to as a deferrable construction. We will study the design of deferrable construction algorithm for matrix $\mathcal{D}(K \times T)$ and the problem of selecting the best variant in our future work. In this paper, we assume that matrix $\mathcal{D}$ has already been formed and given as an input parameter for the problem.

It is noted that the discretization of tasks over time for resource allocation might lead to idle times of resources. If the duration of each time slot is too long, some workflow tasks may already finish the execution but the VMs for succeeding tasks are not ready for execution. The expertises on the application logic and execution time of tasks are therefore important for resource utilization in general and resource usage cost if running in commercial clouds. We also assume that the time slot is sufficient small and the tasks are sufficient longer to reduce the waste due to idle time. It is worth mentioning that this characteristic of workflows allows users to refine resource requirements for their execution. The workflow model is attractive because it allows the users to reserve different amount of resources at different execution time of the workflow instead of reserving fixed amount of resources based on the worst case requirement for the workflow makespan.

3.3. Workflow embedding for bandwidth minimization

We now formulate the optimization problem to generate the embedding solution that minimizes the bandwidth consumption for workflow execution. We first present the objective function and then the constraints applied in case of electrical data centers or optical data centers.

3.3.1. Objective function

Let $\mathcal{M}(K \times M \times N)$ denote the embedding matrix where each element $\mathcal{M}_{k,i,j}$ is the number of VMs allocated for task k embedded in ToR i hosting server j with $k = 1 \dots K,\ i = 1 \dots M$ and $j = 1 \dots N$. Given an embedding matrix, the amount of bandwidth consumed on ToR-core links of the electrical data center or lightpaths of the optical data center for workflow execution, denoted as $\mathcal{Q}_1(\mathcal{M})$, is defined as follows:

$$\mathcal{Q}_1(\mathcal{M}) = \sum_{t=1}^{T}\sum_{k=1}^{K}\sum_{k'=1}^{K}\sum_{i=1}^{M}\sum_{i'=1, i' \neq i}^{M}\sum_{j=1}^{N}\sum_{j'=1}^{N} \mathcal{D}_{k,t} R_{k,k'}^{\text{band}} \mathcal{M}_{k,i,j}\mathcal{M}_{k',i',j'}. \quad (1)$$

This equation means that bandwidth will be consumed for every adjacent task pair k and k' if their requested VMs are embedded in two servers j and j', which are connected to different ToR switches i and i', respectively. Similarly, let $\mathcal{Q}_0(\mathcal{M})$ denote the total amount of bandwidth consumed on server-ToR links, which is computed as follows:

$$\mathcal{Q}_0(\mathcal{M}) = \sum_{t=1}^{T}\sum_{k=1}^{K}\sum_{k'=1}^{K}\sum_{i=1}^{M}\sum_{i'=1}^{M}\sum_{j=1}^{N}\sum_{j'=1}^{N} \mathcal{D}_{k,t} R_{k,k'}^{\text{band}} \mathcal{M}_{k,i,j}\mathcal{M}_{k',i',j'} - \sum_{t=1}^{T}\sum_{k=1}^{K}\sum_{k'=1}^{K}\sum_{i=1}^{M}\sum_{j=1}^{N} \mathcal{D}_{k,t} R_{k,k'}^{\text{band}} \mathcal{M}_{k,i,j}\mathcal{M}_{k',i,j} \quad (2)$$

where the first term is the total amount of bandwidth on server-ToR links consumed for all the adjacent task pairs. Since there may exist several adjacent task pairs whose VMs are allocated on the same server. The communication cost will be zero. Thus, the second term in Eq. (2) computes the bandwidth that will be deducted. The total bandwidth required for workflow execution, which we aim to minimize, for an embedding solution is as follows:

$$\mathcal{Q}(\mathcal{M}) = \mathcal{Q}_0(\mathcal{M}) + \mathcal{Q}_1(\mathcal{M}). \quad (3)$$

It is to be noted that depending on the architecture of the data center network, the total bandwidth consumption is calculated differently given an embedding solution, i.e., the number of elements in Eq. (3) may be different. For instance, in this paper, we consider a two-tier data center so that the bandwidth consumption may incur on the server-ToR and ToR-core links. If there exists the aggregation level, then the bandwidth consumption may also incur on the links that connect the aggregate switches to the core switch. It is also worth mentioning that considering a two-tier data center with single links among network entities, i.e., between servers and ToR switches, and between ToR switches and the core switch, simplifies the problem but still reflects realistic scenarios. We believe that generalizing the problem to a different data center network is straightforward but it may make the problem more complex in some cases. For instance, considering a full data center network with redundant links requires solving the routing problem, i.e., computing the optimal paths for data transmission among VMs or ToR switches. In this paper, we focus on the embedding problem, workflow task dependencies and two-tier data center architecture. This can be extended to three-tier data center architecture with a routing algorithm.

3.3.2. Computing resource constraints

Given an embedding solution, $\mathcal{M}$, it is feasible if and only if it satisfies the two following computing resource constraints. First, it must ensure that, for every workflow task, the number of VMs allocated in the data center is equal to the requested amount:

$$\sum_{i=1}^{M}\sum_{j=1}^{N} \mathcal{M}_{k,i,j} = R_k^{\text{vm}}, k = 1 \dots K. \quad (4)$$

Second, at a given time slot t, the total number of VMs allocated to the workflow at server j connected to ToR i should be less than the number of VMs available in this server. This constraint is represented as follows:

$$\sum_{k=1}^{K} \mathcal{D}_{k,t}\mathcal{M}_{k,i,j} \leqslant \mathcal{C}_{i,j,t}^{\text{vm}}, i = 1 \dots M, j = 1 \dots N. \quad (5)$$

3.3.3. Network resource constraints

The embedding solution also needs to satisfy the constraints of network resources on server-ToR links, and ToR-core links or lightpaths depending on type of data center network. At a given time slot t, the total bandwidth consumed by the workflow on every server-ToR link should also be less than its residual capacity. This constraint is mathematically represented as follows:

$$\sum_{k=1}^{K}\sum_{k'=1}^{K}\sum_{i'=1}^{M}\sum_{j'=1}^{N} \mathcal{D}_{k,t} R_{k,k'}^{\text{band}} \mathcal{M}_{k,i,j}\mathcal{M}_{k',i',j'} - \sum_{k=1}^{K}\sum_{k'=1}^{K} \mathcal{D}_{k,t} R_{k,k'}^{\text{band}} \mathcal{M}_{k,i,j}\mathcal{M}_{k',i,j} \leqslant \mathcal{C}_{i,j,t}^{\text{eband}},$$
$$i = 1 \dots M, j = 1 \dots N. \quad (6)$$

For the bandwidth constraint on the links connecting the ToRs via the core switch, we separately describe this constraint for electrical data centers and optical data centers. While this contraint is simple in electrical data centers, it is more tricky in optical data centers since lightpaths can be dynamically created during the embedding process.

ToR-core Link Constraint in Electrical Networks. At a given time slot t, the total bandwidth consumed by the workflow on every ToR-core link should be less than its residual capacity. This constraint is represented as follows:

$$\sum_{k=1}^{K}\sum_{k'=1}^{K}\sum_{i'=1, i' \neq i}^{M}\sum_{j=1}^{N}\sum_{j'=1}^{N} \mathcal{D}_{k,t} \mathcal{R}_{k,k'}^{\text{band}} \mathcal{M}_{k,i,j}\mathcal{M}_{k',i',j'} \leqslant \mathcal{C}_{i,t}^{\text{cband}},$$
$$i = 1 \dots M. \quad (7)$$

Wavelength Continuity Constraint in Optical Networks. At a given time slot t, the total bandwidth consumed by the workflow on the lightpaths that connect ToR switches i and i' is computed as follows:

$$R_{i,i',t}^{\text{cband}} = \sum_{k=1}^{K}\sum_{k'=1}^{K}\sum_{j=1}^{N}\sum_{j'=1}^{N} \mathcal{D}_{k,t} \mathcal{R}_{k,k'}^{\text{band}} \mathcal{M}_{k,i,j}\mathcal{M}_{k',i',j'},$$
$$i = 1 \dots M, i' = 1 \dots M, i \neq i'. \quad (8)$$

The embedding solution is feasible if the total residual bandwidth of the established lightpaths between ToR i and ToR i' is sufficient, i.e., $R_{i,i',t}^{\text{cband}} \leqslant \mathcal{C}_{i,i',t}^{\text{cband}}$. However, it is not similar to electrical networks where requests will be rejected if the residual bandwidth is not sufficient, the capacity of the link between two ToR switches in optical networks can be elastically added by establishing new lightpaths with different wavelengths if they are available on the fibers connecting those two ToRs to the core switch.

For ToR i, let $A_{i,\,l}$ be a binary variable indicating that wavelength l has been used to connect ToR i to one of other ToRs, i.e., $A_{i,l} = 1$ if wavelength l has been used and $A_{i,l} = 0$, otherwise. Given the absence of wavelength converters, a new lightpath can be established between two ToRs if there exists a common wavelength that has not been used by both ToRs. The maximum bandwidth amount that can be added to the link between two ToRs i and i' is computed as follows:

$$\sum_{l=1}^{W} (1 - A_{i,l})(1 - A_{i',l})\mathcal{C}^{\text{cband}}. \quad (9)$$

The bandwidth constraint of lightpaths is represented as follows for $i = 1 \ldots M$ and $i' = 1 \ldots M$:

$$R^{\text{cband}}_{i,i',t} \leqslant \sum_{l=1}^{W} (1 - A_{i,l})(1 - A_{i',l})\mathcal{C}^{\text{cband}} + \mathcal{C}^{\text{cband}}_{i,i',t}. \tag{10}$$

It is worth mentioning that when new lightpath(s) need to be created to provide required bandwidth, it is not necessary that all available common wavelengths between the two ToRs will be used. Instead, sufficient number of lightpaths will be created and reserves the remaining wavelengths for future usage. Selecting the best wavelengths for lightpath creation therefore has an impact on the connectivity of the network and the utilization of wavelengths. We present our wavelength selection method in Section 5.

3.3.4. Problem statement and formulation

We now give the formal problem statement and its integer programming formulation of the workflow resource request embedding as follows. *"Given a workflow request and the available resource capacity of a data center, find an embedding matrix $\mathcal{M}$ such that the total bandwidth consumed for the workflow execution on the links in the data center is minimized."*

$$\text{Minimize:} \quad \mathcal{Q}(\mathcal{M}) = \mathcal{Q}_0(\mathcal{M}) + \mathcal{Q}_1(\mathcal{M}) \tag{11}$$

$$\text{subject to:} \quad (4), (5), (6) \text{ and } (8) \text{ or } (10), \tag{12}$$

where constraints (8) and (10) are used exclusively depending on the type of the network of the data center.

3.3.5. Problem complexity

Solving such a non-linear optimization problem is computationally prohibitive since simpler virtual network embedding problems have been shown to be $\mathcal{NP}$-complete [9]. Indeed, while existing works claim their $\mathcal{NP}$-completeness nature [6,22,24], they have made several assumptions making the problem much more simpler such as (i) they do not consider capacity physical servers and assume that the ToR switches have sufficient VMs to embed a virtual node, and (ii) a virtual node can be embedded in only one ToR instead of being divided into multiple groups as done in our work. Such problems can be considered as special cases of the workflow request embedding problem since we do not make such assumption and the workflow resource requests are more complex.

Given this consideration, obtaining the optimal solution of our problem even for a reasonable small size is still unaffordable because of the non-linearity nature of the problem and large number of decision variables involved. For instance, if we consider a workflow with 6 tasks and a data center with 6 ToR switches, each connects 4 physical servers, we will deal with a problem with $6 \times 6 \times 4 = 144$ decision variables. It is to be noted that making assumptions simplifies the problem but it makes the problem no longer reflect the realistic scenarios as we aim in our work. It is worth mentioning again that the workflow requests have more complexity in resource requirements due to the data dependencies and time disjointness between workflow tasks. These specific requirements complicate the problem formulation and make it hard to obtain the optimal solution.

Nevertheless, the non-linear integer programming formulated above gives us better description of the embedding problem and better understanding of the problem complexity. It shows that it is very essential for us to devise efficient heuristic algorithms to solve the problem. However, developing efficient heuristic algorithms is a difficult task. Consider the example shown in Fig. 3 where there exist many heuristic embedding solutions. Given the workflow request (Fig. 3a) with three tasks: 1, 2 and 3, each requires 1 VM for 1 time slot. The required bandwidth between task 1 and task 2 is 10 Mbps and that between task 1 and task 3 is 5 Mbps. The data

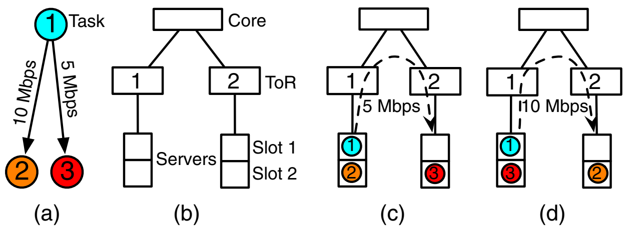


Fig. 3. Embedding example: (a) Three-task workflow request. (b) Data center with 2 servers. (c) One of the best embedding solutions. (d) Another possible way of embedding.

center has 2 ToR switches, each has 1 server. Each server has 1 VM, which is available for 2 consecutive time slots as shown in Fig. 3b. The embedding solution shown in Fig. 3c, which embeds task 1 and task 2 in the same server under ToR 1 and then embeds task 3 in the server under ToR 2, requires only 5 Mbps on the links traversed. The embedding solution shown in Fig. 3d, which embeds task 1 and task 3 in the same server under ToR 1 and then embeds task 2 in the server under ToR 2, is an inferior solution that requires 10 Mbps on the links traversed. In the next section, we develop two heuristic algorithms, namely Critical Path Workflow Embedding (CPWE) and Edge Priority Workflow Embedding (EPWE), to determine a feasible mapping for workflow resource requests while minimizing the bandwidth used for workflow in data centers.

4. Workflow embedding algorithms

Generally, data transmission happens when two workflow tasks are dependent. Since these two tasks execute sequentially, the VMs requested by the preceding task can be reused for the succeeding task. Thus, these VMs can be embedded in the same physical server, thereby nullifying the bandwidth required on the links. The critical path based on bandwidth is a path from the entry task of the workflow to the exit task of the workflow with the highest bandwidth. Algorithm CPWE tries to embed the entire critical path in the same physical server, thereby minimizing the bandwidth needed. On the other hand, algorithm EPWE is computationally simpler since it is based only on a single edge. It sorts all the edges of the workflow in the descending order of consumed bandwidth and then repeatedly embeds each edge until all the edges are embedded. It may so happen that both algorithms CPWE and EPWE cannot embed the critical path or an edge in the same physical server, they then look for the way to embed the request in different servers. We now present the detailed description of these algorithms.

4.1. Critical Path Workflow Embedding

The pseudo code of algorithm CPWE is presented in Algorithm 1. The input parameter of algorithm CPWE is the availability status of data center resources in the time horizon, i.e., time slots $t, t+1, \ldots$ and a workflow request that arrives at time slot t. The output of algorithm CPWE is the embedding matrix in the case the workflow has been admitted or a rejection status, otherwise.

Algorithm CPWE starts by adding two dummy tasks: an ENTRY task and an EXIT task, which are not accounted for in the execution and data transmission. The ENTRY task connects all tasks that do not have any preceding task, i.e., their input data is the workflow input. These tasks will execute immediately when the workflow is launched. The EXIT task connects all the tasks that do not have succeeding tasks, i.e., their output is the final results

Algorithm 1 Critical Path Workflow Embedding

```
Input: Workflow resource request and resource availability status
    of the data center;
Output: Embedding matrix if admitted or rejection;
1: Add ENTRY and EXIT as dummy tasks to workflow;
2: while Exist unmapped task do
3:     Determine the critical path denoted as $\mathcal{CP}$;
4:     SUCCESS $\leftarrow 0$;
5:     if Exist a server that can embed $\mathcal{CP}$ entirely then
6:         Embed $\mathcal{CP}$ on the selected server;
7:         Update $\mathcal{M}$ with embedding solution of $\mathcal{CP}$;
8:         if $\mathcal{M}$ satisfies (4), (5), (6), (8) or (10) then
9:             SUCCESS $\leftarrow 1$;
10:        else
11:            Cancel the embedding solution of $\mathcal{CP}$;
12:        end if
13:    end if
14:    if SUCCESS $= 0$ then
15:        $\mathcal{M}^{CP} \leftarrow$ Run Algorithm 3 with $\mathcal{CP}$;
16:        if $\mathcal{M}^{CP} \neq$ NUL then
17:            Update $\mathcal{M}$ with embedding solution $\mathcal{M}^{CP}$;
18:            if $\mathcal{M}$ satisfies (4), (5), (6), (8) or (10) then
19:                SUCCESS $\leftarrow 1$;
20:            else
21:                Cancel the embedding solution of $\mathcal{CP}$;
22:            end if
23:        end if
24:    end if
25:    if SUCCESS = 0 then
26:        return Rejected;
27:    else
28:        Mark all tasks on $\mathcal{CP}$ as embedded tasks;
29:        Set bandwidth required by $\mathcal{CP}$ to 0;
30:    end if
31: end while
32: return $\mathcal{M}$;
```

of the workflow. Adding these two tasks eases the phase of determining the critical path, which traverses a number of tasks from the ENTRY task to the EXIT task.

The main part of algorithm CPWE is the *while* loop, which will repeat until all tasks in the workflow are embedded or a rejection status is obtained. For each iteration, algorithm CPWE determines the critical path of the workflow, denoted as $\mathcal{CP}$ as shown in line 3. Since there have been many algorithms in the literature to compute the critical path in a workflow, we omit the description of such algorithm in this paper. We refer the readers to [28] and [29] for the detailed descriptions of critical path selection algorithms. Given the critical path, $\mathcal{CP}$, algorithm CPWE verifies if there exists a server that can embed $\mathcal{CP}$ entirely, i.e., all VMs requested by all the tasks on the critical path will be embedded on that server. If such a server does not exist or the embedding solution of $\mathcal{CP}$ does not satisfy all the constraints described in Eq. (12), CPWE then uses the procedure in Algorithm 3 to divide the set of VMs requested by the tasks on the critical path into multiple partitions, each will be embedded on a different server. The result of this step is stored in $\mathcal{M}^{CP}$ as shown in line 15. We describe Algorithm 3 in detail in Section 4.3. In the worst case, if the critical path is not successfully embedded even it is divided, the request is then rejected. Otherwise, all the tasks on the critical path will be marked as embedded and all the edges on the critical path are then updated to request a zero bandwidth (see lines 28 and 29). By setting the required bandwidth of the edges connecting the embedded tasks to zero, these edges will not affect the critical path selection in the next iteration of the *while* loop.

To verify whether there exists a server that can embed the critical path entirely or not, a practical implementation needs to consider the availability status of the server in the time horizon of all the tasks on the critical path. A possible solution is to use a three-dimensional matrix S where $S_{i,j,t}$ indicates the number of VMs available at time slot t on server j connected to ToR switch i, for $i = 1 \ldots M$, $j = 1 \ldots N$ and $t = 1, 2, \ldots$ Given that task k requests for R_k^{vm} VMs in the time horizon represented by vector $\mathcal{D}_{k,t}$, which was set to 1 if task k requires resources at time slot t, we can align S with $\mathcal{D}$ and R_k^{vm} for the verification purpose.

4.2. Edge Priority Workflow Embedding

Algorithm EPWE is computationally simpler than algorithm CPWE since it does not need to compute the critical path. It embeds the workflow resource request using a greedy manner that gives the priority for the two tasks whose edge consumes higher bandwidth. As shown in Algorithm 2, EPWE sorts all the edges of the workflow in the descending order based on the bandwidth requirement. Similar to algorithm CPWE, for each edge in the sorted list, algorithm EPWE tries to embed the VMs requested by the two tasks in the same server. If it is not successful to do so, EPWE will run Algorithm 3 to divide the requested VMs into multiple partitions, each will be embedded in a different server. Obviously, there will be the case that an edge could not be embedded even though its requested VMs have been divided into multiple partitions. A rejection status is then announced. The input and output of algorithm EPWE are similar to that of algorithm CPWE.

Algorithm 2 Edge Priority Workflow Embedding

```
Input: Workflow resource request and resource availability status
    of the data center;
Output: Embedding matrix if admitted or rejection;
1: Sort all edges in the descending order of the bandwidth re-
    quirement. Let $\mathbb{E}$ denote the sorted list of edges;
2: for each $e \in \mathbb{E}$ do
3:     SUCCESS $\leftarrow 0$;
4:     if Exist a server that can embed $e$ entirely then
5:         Embed $e$ on the selected server;
6:         Update $\mathcal{M}$ with embedding solution of $e$;
7:         if $\mathcal{M}$ satisfies (4), (5), (6), (8) or (10) then
8:             SUCCESS $\leftarrow 1$;
9:         else
10:            Cancel the embedding solution of $e$;
11:        end if
12:    end if
13:    if SUCCESS $= 0$ then
14:        $\mathcal{M}_e \leftarrow$ Run Algorithm 3 with $e$;
15:        if $\mathcal{M}_e \neq$ NUL then
16:            Update $\mathcal{M}$ with embedding solution $\mathcal{M}_e$;
17:            if $\mathcal{M}$ satisfies (4), (5), (6), (8) or (10) then
18:                SUCCESS $\leftarrow 1$;
19:            else
20:                Cancel the embedding solution of $e$;
21:            end if
22:        end if
23:    end if
24:    if SUCCESS = 0 then
25:        return Rejected;
26:    end if
27: end for
28: return $\mathcal{M}$;
```

Algorithm 3 Task Embedding On Different Servers

```
Input: Path $P$ and resource availability status of the data center;
Output: Embedding solution $\mathcal{M}_P$ or rejection status;
 1: $R^{\max} = \max\{R_k^{\text{vm}}, k \in P\}$;
 2: $G_2 \leftarrow 1$;
 3: while $G_2 \leqslant R^{\max} - 1$ do
 4:     $G_1 = R^{\max} - G_2$;
 5:     for $k \in P$ do
 6:         $R_{k,1}^{\text{vm}} = \min\{R_k^{\text{vm}}, G_1\}$;
 7:         $R_{k,2}^{\text{vm}} = R_k^{\text{vm}} - R_{k,1}^{\text{vm}}$;
 8:     end for
 9:     if Exist 2 servers that can embed $R_{k,1}^{\text{vm}}, R_{k,2}^{\text{vm}}$ then
10:         Embed $R_{k,1}^{\text{vm}}$ and $R_{k,2}^{\text{vm}}$ on the selected servers;
11:         $\mathcal{M}_P \leftarrow$ The solution of $R_{k,1}^{\text{vm}}$ and $R_{k,2}^{\text{vm}}$;
12:         return $\mathcal{M}_P$;
13:     end if
14:     $G_2 \leftarrow G_2 + 1$;
15: end while
16: $\mathcal{M}_P \leftarrow$ NUL;
17: return $\mathcal{M}_P$;
```

It is to be noted that while algorithm EPWE also uses the procedure presented in Algorithm 3 for dividing a request into multiple partitions to embed in different servers, the partitioning involves only the VMs requested by the two tasks connected by the edge. This partitioning is computationally less intensive compared to the partitioning in algorithm CPWE, which may involve more than one edge since the number of edges on the critical path depends on the number of execution stages of the workflow. A stage contains the tasks that can execute concurrently. This makes algorithm EPWE further simpler than algorithm CPWE.

4.3. Task embedding on different servers

Given a critical path determined in algorithm CPWE or an edge determined in algorithm EPWE, there may not exist a server, which can host all the VMs requested by the tasks. Since all the tasks on the critical path or the edge execute sequentially, the cause of the embedding failure comes from the fact that there is at least one task, which requires a larger number of VMs than the available VMs of the server at a specific time. For such a task, a possible solution is to divide its requested VMs into multiple partitions, each will be embedded in a different server that has sufficient capacity. However, it is not straightforward to do so. Many issues need to be considered to obtain an optimal partitioning such as the number of partitions, the number of VMs in each partition and the bandwidth consumed for each partitioning. Exhaustively, we can repeatedly divide the original request into two partitions. We adjust the number of VMs in the first partition until it can be embedded in a certain server. For the second partition, if there does not exist a server, which can embed it, we will repeat again the procedure to divide it into two partitions. The number of partitions is then not pre-determined, the partitioning procedure repeats until all requested VMs are embedded or the data center cannot accommodate the request. It is obvious that this method is computationally intensive. In the worst case, for task k that requests for R_k^{vm} VMs, the complexity of the procedure is $\mathcal{O}(\frac{R_k^{\text{vm}}(R_k^{\text{vm}}+1)}{2}MN)$ where M is the number of ToR switches, N is the number of servers connected to each ToR switch. It is to be noted that while the number of VMs required by workflow tasks may be in the order of several dozen of VMs, the number of ToR switches and physical servers is much larger, e.g. the number of ToR switches connected to a core switch can be up to 48 and similar for the number of physical servers connected to a ToR switch. This makes it impossible to iterate through all possible partitions and determine the best.

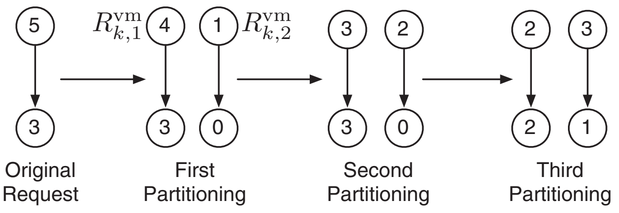


Fig. 4. 2-partition example with a critical path of 2 tasks. The number shown in the tasks is the number of VMs requested.

In addition to the complexity of the partitioning procedure, the amount of bandwidth consumed for the request might dominate. As shown in [8], dividing a request into more than 2 partitions may admit more requests but more link bandwidth may be needed to satisfy the inter-partition communications, leading to the rejection of the future arriving requests. Therefore, we limit the number of partitions to 2 and develop an algorithm to perform the partitioning. The pseudo code of the partitioning procedure is presented in Algorithm 3.

Let P be the input of Algorithm 3 where P can be critical path $\mathcal{CP}$ from CPWE or edge e from EPWE. Task $k \in P$ requests for R_k^{vm} VMs. As shown in line 1, the algorithm starts by determining the task in P, which requires the highest number of VMs, denoted as $R^{\max}$, as this task can be the cause of failure when trying to embed in the same server. While the VMs of this task are reused for its succeeding tasks once embedded sufficiently, it requires more VMs apart from the VMs released by its preceding task.

Given $R^{\max}$, the algorithm tries all combinations of the two possible partitions of VMs, where the number of VMs in each partition is denoted and G_1 and G_2, respectively. For instance, the first trial considers the first partition has $G_1 = R^{\max} - 1$ VMs and the second partition has $G_2 = 1$ VM. For every task k in P, we then have the new resource requirement represented by two values, $R_{k,1}^{\text{vm}} = \min\{R_k^{\text{vm}}, G_1\}$ and $R_{k,2}^{\text{vm}} = R_k^{\text{vm}} - R_{k,1}^{\text{vm}}$, respectively. The algorithm looks for two servers, which can entirely embed these two partitions, respectively. In case, one of the partitions could not be embedded, the algorithm tries the next possible partitioning. These steps are presented in the *while* loop (lines 3–15). In Fig. 4, we present an example of the partitioning for the first three iterations. If the algorithm has tried all combinations without success, the request is rejected.

It is worth mentioning that integrating the proposed algorithms into realistic systems is feasible without complicating the current system architecture of cloud data centers. It will not require any additional computing resources since every cloud infrastructure has a cloud resource manager that is responsible for receiving users' requests and allocating resources to admitted requests. The proposed algorithms should be implemented as a component of the cloud resource manager to perform the embedding of workflow resource requests. Since the proposed algorithms are heuristic and run in polynomial time, the overhead added to the cloud manager is negligible. Furthermore, the information exchanged between the manager and computing servers is only the embedding solutions. Therefore, the bandwidth used for such communication can be ignored.

4.4. Computational complexity of algorithms

In this section, we analyze the computational complexity of the proposed algorithms. As mentioned in Section 4.3, the complexity of Algorithm 3 in the worst scenario is $\mathcal{O}(\frac{R_k^{\text{vm}}(R_k^{\text{vm}}+1)}{2}MN)$ where M is the number of ToR switches, N is the number of servers connected to each ToR switch. Algorithm 3 is used in Algorithms 1 and 2, which have to iterate until all workflow tasks have been embedded. Given that there is K tasks in a workflow request, for every iteration, Algorithm 1 needs to compute the critical path whose the algorithm for finding the critical path in a workflow has a complexity of $\mathcal{O}(|E| + |K|\log|K|)$ where $|E|$ is the number of edges in the workflow[1]. The computational complexity of Algorithm 1 in the worst scenario is therefore $\mathcal{O}(\frac{R^{\text{vm}}(R^{\text{vm}}+1)}{2}(|E| + |K|\log|K|)KMN)$ whereas the complexity of Algorithm 2 is $\mathcal{O}(\frac{R^{\text{vm}}(R^{\text{vm}}+1)}{2}KMN)$ where R^{vm} is the maximum number of VMs required by a workflow task. As mentioned earlier, the most important components that influence the algorithm complexity are the number of ToR switches in the data center and the number of physical servers connected to a ToR switch. The values of these two components are much larger than the number of VMs required by each workflow task and the number of tasks in each workflow.

It is to be noted that the task dependencies of workflow requests do not affect the size of the optimization problem, i.e., the non-linear integer programming problem defined in the previous section since the number of decision variables depends only on the number of tasks of the workflow, the number of ToR switches and the number of physical servers under each ToR switch. However, it actually affects the complexity of the proposed algorithms. Indeed, the Critical Path Workflow Embedding needs to compute the critical path of the workflow based on bandwidth consumption of the edges between tasks. The more complex the task dependencies, the more complex the algorithm. Similarly, the Edge Priority Workflow Embedding needs more iterations for embedding an edge of the workflow. Furthermore, with more edges in a workflow, the algorithms needs more verifications for the bandwidth constraints on both levels of links in data centers since two dependent tasks are highly embedded on two different physical servers or ToRs.

5. Dealing with optical network characteristics

While our optimization formulation presented in Section 3 considers how efficiently bandwidth of the links in data centers is used, dealing with lightpath creation in optical data centers needs new methods to: (i) select the ToR that has the feasibility to create new lightpath when communication is required, and (ii) select the wavelength to improve the wavelength utilization, thus increasing the bandwidth efficiency. In this section, we present our proposed methods to solve these issues.

5.1. Top-of-the-Rack selection

In electrical packet-switched data centers, a request will be rejected if required bandwidth is not available on the links connecting ToRs. In optical data centers, for such cases, we need to check whether it is possible to create a new lightpath between the ToRs before rejecting the request. A ToR with "good" connectivity with others will likely reduce the rejection ratio of the requests. Thus, selecting a ToR with the highest connectivity, i.e., the one with the highest possibility of reaching out to other ToRs, is a good choice among the ToRs. We define a function to compute the connectivity index of a ToR depending on the number of available wavelengths and its reachability to other ToRs in the data center. For ToR i, the connectivity index function is defined as follows:

$$F(i) = \frac{\sum_{j=1, j\neq i}^{M} \mathcal{G}(i,j)}{M-1}, \tag{13}$$

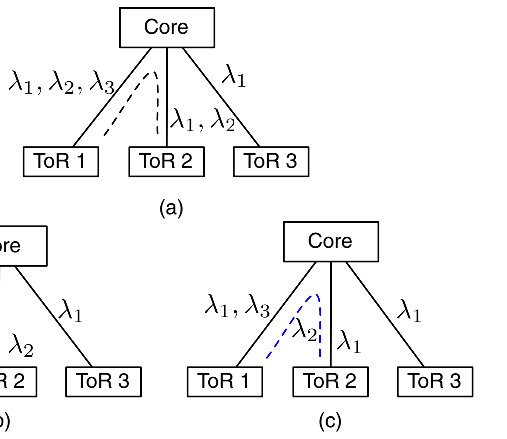


Fig. 5. An example of wavelength selection: (a) Wavelength availability and lightpath to be created. (b) Use of λ_1 disconnects ToR 1 and ToR 3. (c) Use of λ_2 ensures good connectivity among ToRs.

where $F(i)$ is the connectivity index of ToR i, $\mathcal{G}(i, j)$ is the number of lightpaths that can be created between ToR switches i and j, i.e., the number of available wavelengths that are common between the two ToR switches, and M is the total number of ToRs in the data center. By choosing the ToR with the highest connectivity index, we can ensure high reachability among ToRs upon creating a new lightpath from this ToR. Furthermore, this function also ensures a good balancing of wavelengths with many wavelength-continuous paths among ToRs. The rationale for using this function is to keep high degree of connectivity among ToR switches and avoid that ToR switches are disconnected even though wavelengths are available.

5.2. Wavelength selection for lightpath creation

When a lightpath needs to be created between two ToRs, selection of a wavelength is an important problem to maximize wavelength utilization. Within a data center, the dynamic arrival of resource requests and high degree of the optical switch make the impact of the continuity constraint more pronounced. Consider an example shown in Fig. 5 where a lightpath between ToR 1 and ToR 2 needs to be created. Since they have two common wavelengths, i.e., λ_1 and λ_2, one of them can be used for the new lightpath. However, if λ_1 is used, ToR 1 cannot reach ToR 3 in future if communication is required between ToR 1 and ToR 3 (see Fig. 5b). Thus, it is better to use λ_2 to connect ToR 1 and ToR 2 and reserve λ_1 for the future as shown in Fig. 5c. To solve this problem, we propose a function that computes the goodness value for each wavelength between a given pair of ToRs to choose the wavelength with the highest goodness value for a lightpath. The goodness function for a common wavelength λ between ToR i and ToR j is defined as follows:

$$F(\lambda) = \frac{1}{R(\lambda)} \tag{14}$$

where $R(\lambda)$ is the number of fibers on which wavelength λ is available. This is equivalent to choosing a wavelength that is used on most of the fibers, ensuring the highest connectivity of ToRs.

5.3. Integration into embedding algorithms

The ToR selection method and the wavelength selection method need to be integrated into the algorithms described in Section 4 to realize the embedding in optical data centers. The ToR selection method will be invoked by line 5 in Algorithm 1, line 4 in Algorithm 2 and line 9 in Algorithm 3 where the algorithms check whether there exist a candidate server for embedding a critical

[1] Finding the critical path in a workflow is the inverse problem of finding the shortest path in a graph. Thus, to compute the critical path, we adopt the Dijkstra algorithm that has a complexity of $\mathcal{O}(|E| + |V|\log|V|)$ where $|E|$ is the number of edges and $|V|$ is the number of nodes in the graph.

path, an edge or a partition of the critical path or edge. If there exist multiple candidate servers, the ToR selection method is used to determine the best ToR that hosts the best server for embedding. It is to be noted that ToR selection in packet-switched electrical data centers is much simpler since it needs to verify only the availability of computing resources (VMs) and the bandwidth of the links. In optical data centers, as discussed earlier, selection of a ToR has an impact on the future connectivity of the network due to the wavelength continuity constraint. Thus, in addition to the verification of the availability of computing and network resources, the algorithm also needs to verify the future connectivity of the network caused by the selection of ToR.

Similar to the ToR selection method, the wavelength selection method is integrated into line 18 of Algorithm 1 and line 17 of Algorithm 2 during the verification of the constraint presented in Eq. (10). As presented in Section 3, there may be multiple wavelengths available for lightpath creation, we need to select the best wavelength(s) to provide sufficient bandwidth for the request.

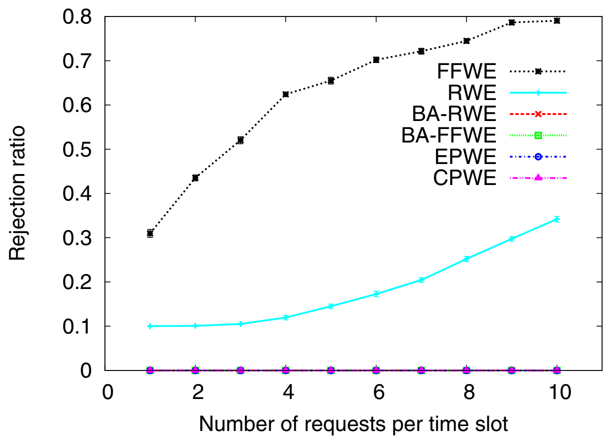


Fig. 6. Rejection ratio for random workflows.

6. Performance study

In this section, we evaluate the performance of the proposed algorithms. We first describe the settings of simulation parameters and the baseline algorithms for comparison purpose then we analyze the obtained results.

6.1. Parameter settings

We consider the electrical data center and optical data center with the architecture as shown in Fig. 1. The core switch connects 16 ToR switches. The number of servers connected to each ToR is set to 16. The number of VMs, which can be hosted in each server, is set to 64. The maximum number of VMs available in the data center is 16384. We set the capacity of server-ToR links to 1 Gbps. In the electrical data center, the capacity of ToR-core links is set to 10 Gbps. In the optical data center, each fiber can carry up to 10 wavelengths with wavelength capacity set to 1 Gbps. Thus, in both electrical and optical data centers, the over-subscription ratio is 1: 1.6.

Workflow requests are generated randomly by a DAG generator. The total number of tasks in a workflow is chosen randomly from [3, 25]. The number of VMs required by a task is chosen randomly from [10, 20]. The bandwidth requirement between two dependant tasks is assumed to be in the range of [600, 800] Kbps.

6.2. Performance of CPWE and EPWE

In this section, we evaluate the performance of embedding algorithms: CPWE and EPWE. We consider the electrical data center to eliminate the impact of the wavelength continuity constraint on the performance.

6.2.1. Performance metrics and Comparison

We use the two following performance metrics to evaluate the proposed algorithms.

- Average bandwidth required per accepted request: We compute the total bandwidth consumption then divide for the number of accepted requests;
- Average rejection ratio: The ratio between the number of rejections and the number of requests arrived.

We compare the performance of the proposed algorithms to that of the four following baseline algorithms.

- First Fit Workflow Embedding (FFWE) looks for the first fit server that has sufficient VMs to embed a certain workflow task. It repeats the same process until all tasks have been embedded. If no server is available for a certain task, the request is rejected.
- Random Workflow Embedding (RWE) randomly selects a server that has sufficient VMs for a workflow task. If the selected server cannot accommodate, RWE removes this server from the list then selects another server. In the worst case, if no server is available for the task, the request is rejected.
- Bandwidth-aware First Fit Workflow Embedding (BA-FFWE) is similar to FFWE when selecting a server. It additionally verifies whether the links connecting the servers can accommodate the bandwidth requirement if all the successors of the task are spread over different servers, i.e., the maximum bandwidth required by the task in the worst embedding, termed as the *upper bound* bandwidth requirement.
- Bandwidth-aware Random Workflow Embedding (BA-RWE) is similar to RWE in the way to select a server but it additionally considers the upper bound bandwidth requirement as in BA-FFWE.

It is to be noted that algorithm FFWE is the simplest algorithm similar to *Myopic* algorithm [16]. Algorithm RWE is also considered as a well-known baseline algorithm for performance comparison in many previous works [30,31]. While algorithms FFWE and RWE embed workflow tasks naively without considering the bandwidth consumption for data transmission between tasks, we improve these two algorithms by adding the upper bound bandwidth requirement, resulting in algorithms BA-FFWE and BA-RWE for comparison purpose.

6.2.2. Analysis of results

Performance with Random Workflows. In Fig. 6, we present the rejection ratio of all the algorithms with respect to the arrival rate, i.e., the number of workflow resource requests that arrive per time slot. It is observed that algorithms FFWE and RWE reject many requests even at a low arrival rate while other algorithms, which consider bandwidth requirement, perform better and do not reject any request. This is explained by the fact that algorithms FFWE and RWE do not consider the bandwidth requirement of the edges between a preceeding task and its succeeding tasks while determining the physical server for the preceeding tasks. Thus, even though the physical server selected for the preceeding task has sufficient VMs, the link connecting the physical server to the ToR switch as well as the link connecting the corresponding ToR switch to the core switch may not have sufficient bandwidth if later on the succeeding tasks are embedded on different physical servers or different ToRs. It is to be noted that due to the complexity of the

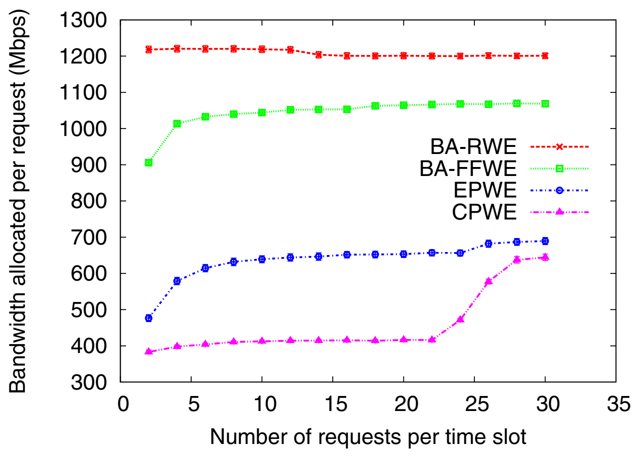


Fig. 7. Bandwidth required per accepted request for random workflows.

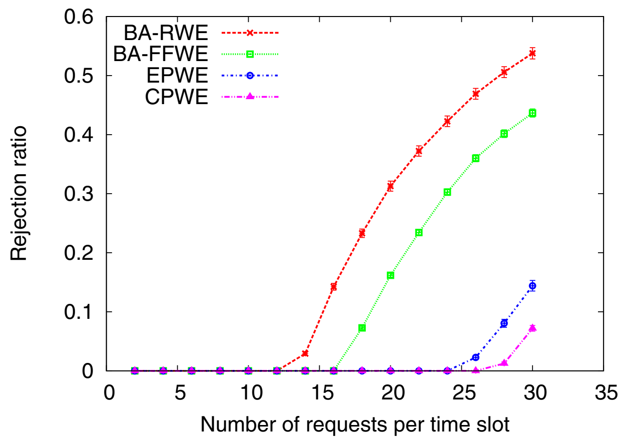


Fig. 8. Rejection ratio for random workflows.

embedding problem, the algorithms do not perform the iteration to verify all possible embedding solutions. The request will be rejected if the first trial fails. Since it is obvious that algorithms FFWE and RWE always have the worst performance, we omit their presentations hereafter.

Fig. 7 presents the average bandwidth required per accepted request with respect to the arrival rate. The results show that CPWE and EPWE outperform the baseline algorithms. While CPWE has a global view about the bandwidth consumption by analyzing the critical path, EPWE gives the priority for the edge with the highest required bandwidth. Embedding the critical path or such edge in the same server reduces the bandwidth consumption significantly. As shown in Fig. 7, EPWE reduces the bandwidth required per accepted request by about 40% compared to BA-FFWE and 50% compared to BA-RWE. Algorithm CPWE performs even better reducing the bandwidth required per accepted workflow request by up to 55% compared to BA-FFWE and 66% compared to BA-RWE. Comparing CPWE and EPWE, we observe that, when the arrival rate is low, entire critical path is well accommodated in the same server. Thus, CPWE outperforms EPWE. However, at high arrival rate, embedding the entire critical path in the same server is more difficult, the critical path is then spread over different servers, making the bandwidth consumption increases drastically. Thus, EPWE performs better than CPWE.

Since CPWE and EPWE use less bandwidth for each workflow while guaranteeing the bandwidth requirement, they can accommodate more requests compared to the baseline algorithms. Consequently, CPWE and EPWE start rejecting resource requests much later than the baseline algorithms as shown in Fig. 8. While the baseline algorithms start rejecting the request at the arrival rate of 16, EPWE and CPWE start rejecting the request at the arrival rate of 24 and 26, respectively.

Performance with Real Application Workflows. We also evaluate the proposed algorithms using realistic-application workflows: a medical image analysis [15], an Alzheimer's disease case study [32] and a protein structure annotation workflow [13]. For each time slot, we generate a number of requests for these three workflows. We obtain the same behavior as we obtained in the case of random workflows. Fig. 9 presents the average bandwidth required per accepted request. It is observed that algorithm CPWE saves up to 70% and 80% of bandwidth required for a workflow compared to BA-FFWE and BA-RWE, respectively. EPWE has similar performance by reducing about 65% and 78% of bandwidth allocated to a workflow compared to BA-FFWE and BA-RWE, respectively. We also observe that large variation of bandwidth requirement on the edges of

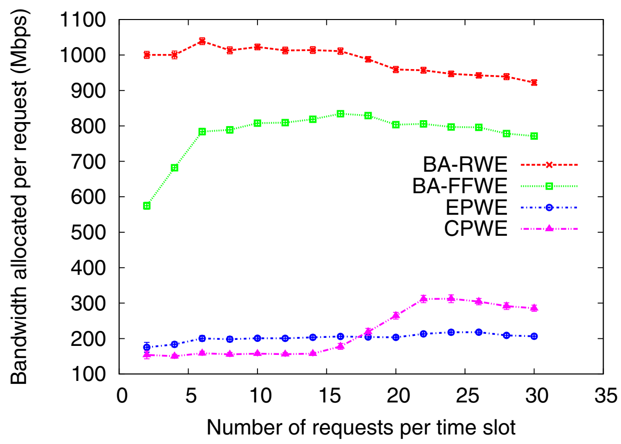


Fig. 9. Average bandwidth allocated to a realistic workflow.

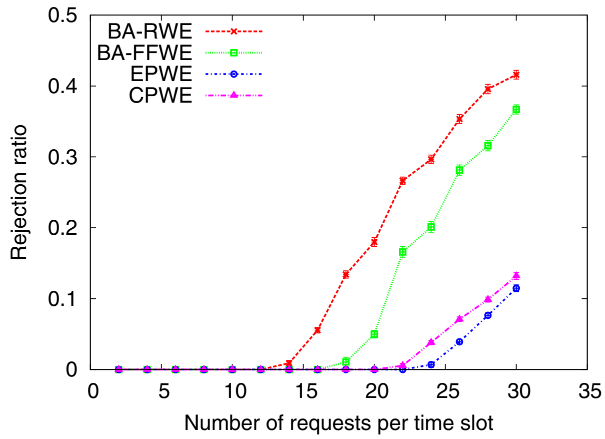


Fig. 10. Rejection ratio for realistic workflows.

realistic-application workflows makes EPWE performs better than CPWE at high arrival rate. Fig. 10 presents the rejection ratio of the proposed algorithms compared to the baseline algorithms. The results show that algorithms CPWE and EPWE can accommodate up to 22 concurrent requests while BA-RWM and BA-FFWM can afford up to 10 and 16 concurrent requests, respectively. It is to be noted that we increase the loads, i.e., the number of workflow resource requests, gradually to evaluate the performance of the algorithms from small loads to high loads. Since we used the realistic

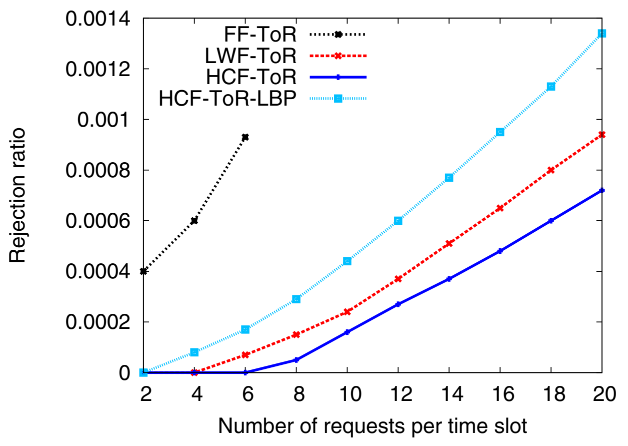


Fig. 11. Performance of proposed HCF-ToR (vs) other methods.

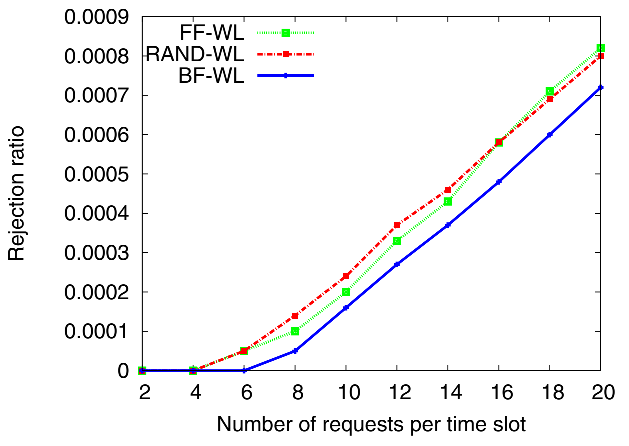


Fig. 12. Performance of wavelength selection methods.

workflow applications for the performance study, e.g., a medical image processing application [15,33], the number of input parameters, i.e., the number of images provided by the patients, creates the workflow instances submitted to the data center for execution. Such a scaling behavior represents the number of patients whose images are processed concurrently by the workflow applications in the data center.

6.3. Performance of algorithms in optical data centers

As shown in previous section, CPWE has the best performance for workflow embedding in data centers. Thus, we use CPWE to evaluate the performance of the proposed methods that deal with the challenging issues of optical data centers: ToR and wavelength selection methods.

6.3.1. Performance of ToR selection method

We compare the performance of the proposed ToR selection method that uses the highest connectivity ToR first denoted as HCF-ToR. We compare the performance of HCF-ToR with that of the following baseline methods:

- First-fit ToR (denoted as FF-ToR) selects the first ToR that has sufficient VMs to embed the request;
- Largest-wavelength First ToR (denoted as LWF-ToR) selects the ToR based on the number of available wavelengths. The ToR with the highest number of available wavelengths is selected to host VMs;
- Highest Connectivity ToR First with Lowest Bandwidth Path (HCF-ToR-LBP) uses the connectivity index function to determine the ToR and embed the path with the lowest bandwidth consumption.

The results in Fig. 11 show that the FF-ToR method has the worst performance. Our method, HCF-ToR, which uses the highest connectivity ToR first, reduces the rejection ratio significantly. When the arrival rate is 20 requests per time slot, HCF-ToR achieves a performance gain of 25% in terms of reduction of rejection ratio compared to LWF-ToR. The results also show that using the critical path based embedding achieves better performance. Compared to HCF-ToR-LBP, which gives priority to the path with the lowest bandwidth consumption, HCF-ToR achieves a performance gain of 47% in terms of reduction of rejection ratio.

6.3.2. Performance of wavelength selection method

In this simulation, we evaluate the performance of the wavelength selection method. We compare the performance when using the proposed goodness function for wavelength selection called

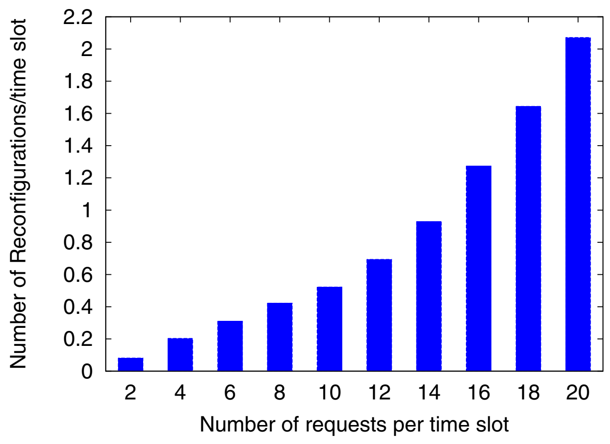


Fig. 13. Number of reconfigurations of lightpaths.

Best Fit Wavelength selection (BF-WL) with that of the following baseline methods. All use HCF for ToR selection and critical-path based embedding.

- First-fit wavelength method denoted as FF-WL;
- Random wavelength method denoted as RAND-WL.

Results in Fig. 12 show that using the goodness function has better performance than other methods by achieving a performance gain of up to 15% in terms of reduction of rejection ratio.

6.3.3. Dynamic reconfiguration of lightpaths

The flexibility of optical switch brings in the dynamic reconfiguration of lightpaths where a lightpath can be removed dynamically when no longer required. However, dynamic reconfiguration incurs additional overhead. We carry out reconfiguration at the end of each time slot and the average number of reconfigurations per time slot is shown in Fig. 13. It is observed that at the arrival rate of 20, only about 2 lightpaths are removed. With such a small number of reconfigurations, we achieve a significant performance improvement by reconfiguration as shown in Fig. 14. The rejection ratio in the data center without dynamic reconfiguration of lightpaths increases drastically. At the arrival rate of 20, the rejection ratio reaches 0.4% with no reconfiguration, while that of the data center with dynamic reconfiguration of lightpaths and the electrical packet-switched data center are only 0.072% and 0.036%, respectively.

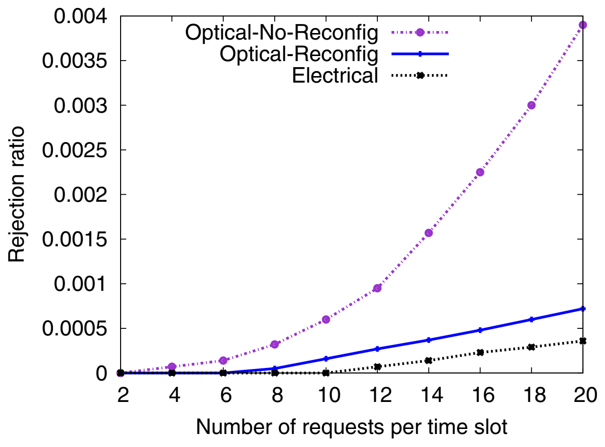


Fig. 14. Performance improvement of dynamic reconfiguration.

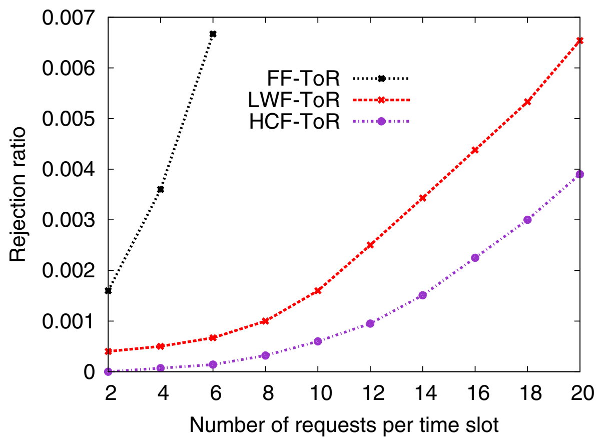


Fig. 15. Performance of proposed HCF-ToR without dynamic reconfiguration.

6.3.4. *Performance without dynamic reconfiguration*

We observe a similar behavior when evaluating the performance of the ToR selection method and the wavelength selection method in the case of no dynamic reconfiguration of lightpaths, i.e., lightpaths are incrementally configured. In Fig. 15, we present the performance of the ToR selection method. The FF-ToR method again has the worst performance by rejecting roughly 10% of requests when the arrival rate is 20 requests per time slot. The HCF-ToR method has the best performance by reducing the rejection ratio to 0.4% while that of the LWF-ToR method is 0.65%, corresponding to a performance of 40%.

6.4. *Optical vs electrical data centers*

We evaluate the impact of the wavelength continuity constraint by comparing the rejection ratio when embedding workflow resource requests in electrical and optical data centers. The capacity of the links, which connect ToR switches to the core switch in the electrical data center, is set to 10 Gbps. This is equivalent to the case of optical data center with 10 wavelengths per fiber, each has bandwidth capacity of 1 Gbps. In Fig. 16, we present the performance in terms of rejection ratio for optical and electrical data centers. It is observed that at the arrival rate of 20 requests per time slot, the rejection ratio of the optical data center reaches 0.072% while that of the electrical data center is only 0.036%. The performance of the optical data center is improved when wavelength conversion is used, where the rejection ratio decreases to 0.052% at the arrival rate of 20, corresponding to a reduction of 28%. This is due to the wavelength continuity constraint in the

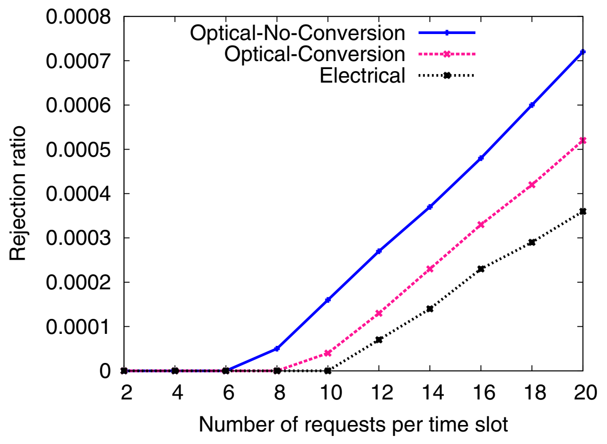


Fig. 16. Performance comparison of optical versus electrical data centers: impact of continuity constraint.

optical network. Indeed, while a ToR switch in the electrical network can reach any other ToR switch as long as there is available bandwidth in the links that connect the two ToR switches to the core switch, a ToR in the optical network can simultaneously reach maximum a number of ToRs depending the number of wavelengths carried by the fiber that connects the ToR switch to the core optical switch. This leads to the fact that two ToRs may still have sufficient VMs for allocating to two dependant workflow tasks but they cannot connect with each other since they have no more available (and common in case of no wavelength converter) wavelength to establish the lightpath even though the bandwidth provided by other wavelengths is not fully utilized. This additional constraint of optical networks makes them perform poorer than electrical networks.

We note that for electrical networks, we considered an ideal case where a single link with aggregate capacity of number of wavelengths multiplied by capacity per wavelength. This ideal case is not feasible when the number of wavelengths per fiber increases, i.e., it is impractical (even impossible) to fabricate a copper cable with capacity of 100 Gbps. In such a case, a complex network structure with more fibers and switches will be required in packet-switched networks.

7. Conclusion

In this paper, we addressed the problem of embedding workflow resource requests in packet-switched and circuit-switched optical data centers with the consideration of bandwidth requirements. We formulated the embedding problem as an optimization programming model that minimizes the bandwidth required for workflow execution. Since the optimization problem is computationally prohibitive, we developed two heuristic algorithms namely CPWE and EPWE to solve this problem. To deal with the specific characteristic of optical networks, we additionally proposed the methods for ToR selection and wavelength selection to improve the connectivity of optical networks and wavelength utilization. The validation was performed through simulations with random workflows as well as workflows of realistic applications. The simulation results demonstrate the effectiveness of the proposed algorithms by reducing up to 66% of bandwidth required for random workflows and 80% of bandwidth required for realistic-application workflows compared to the baseline algorithms. The results also show that the ToR selection method and the wavelength selection method applied in optical data centers outperform other methods by reducing the rejection ratio up to 47% with dynamic reconfiguration of lightpaths and 40% with incremental configuration of lightpaths.

Acknowledgment

This work was supported by Singapore Ministry of Education, Academic Research Fund Tier 2, Grant No. MOE2013-T2-2-135, NUS WBS No. R-263-000-B11-112.

References

[1] V. Girisagar, T. Truong-Huu, M. Gurusamy, Mapping workflow resource requests for bandwidth efficiency in data centers, in: 17th International Conference on Distributed Computing and Networking (ICDCN 2016), Singapore, 2016, pp. 1–10.
[2] Cisco Global Cloud Index: Forecast and Methodology, 2013–2018, 2015, (White Paper).
[3] D.A. Brown, P.R. Brady, A. Dietz, J. Cao, B. Johnson, J. McNabb, A case study on the use of workflow technologies for scientific analysis: gravitational wave data analysis, in: I.J. Taylor, E. Deelman, D.B. Gannon, M. Shields (Eds.), Workflows for e-Science, Springer, 2007, pp. 39–59.
[4] X. Li, B. Plale, N. Vijayakumar, R. Ramachandran, S. Graves, H. Conover, Real–time storm detection and weather forecast activation through data mining and events processing, Earth Sci. Inf. 1 (2) (2008) 49–57.
[5] E. Deelman, S. Callaghan, E. Field, H. Francoeur, R. Graves, N. Gupta, V. Gupta, T.H. Jordan, C. Kesselman, P. Maechling, J. Mehringer, G. Mehta, D. Okaya, K. Vahi, L. Zhao, Managing large-scale workflow execution from resource provisioning to provenance tracking: the cybershake example, in: e-Science 2006, Amsterdam, the Netherlands, 2006, pp. 1–14.
[6] S. Hegde, R. Srinivas, D.M. Divakaran, M. Gurusamy, Dynamic embedding of virtual networks in hybrid optical-electrical datacenters, in: IEEE ICCCN 2014, Shanghai, China, 2014, pp. 1–8.
[7] A. Dalvandi, M. Gurusamy, K.C. Chua, Time-aware VM-placement and routing with bandwidth guarantees in green cloud data centers, in: CloudCom 2013, Bristol, UK, 2013, pp. 212–217.
[8] D.M. Divakaran, N.T. Le, M. Gurusamy, An online integrated resource allocator for guaranteed performance in data centers, IEEE Trans. Parallel Distrib. Syst. 25 (6) (2014) 1382–1392.
[9] A. Fischer, J.F. Botero, M.T. Beck, H. De Meer, X. Hesselbach, Virtual network embedding: a survey, IEEE Commun. Surv. Tutor. 15 (4) (2013) 1888–1906.
[10] T. Glatard, J. Montagnat, D. Lingrand, X. Pennec, Flexible and efficient workflow deployement of data-intensive applications on grids with MOTEUR, IJHPCA 22 (3) (2008) 347–360.
[11] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E.A. Lee, J. Tao, Y. Zhao, Scientific workflow management and the kepler system: research articles, Concurr. Comput.. 18 (10) (2006) 1039–1065.
[12] A. Zareie, M.M. Pedram, M. Kelarestaghi, A. Kosari, An approach to papping scientific workflow in cloud computing data centers to minimize costs of workflow execution, Int. J. Comp. Tech. Appl. 2 (5) (2011) 1627–1640.
[13] Y. Ahn, Y. Kim, Auto-scaling of virtual resources for scientific workflows on hybrid clouds, in: ScienceCloud, Vancouver, Canada, 2014, pp. 47–52.
[14] M. Maheswaran, S. Ali, H.J. Siegel, D. Hensgen, R.F. Freund, Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems, in: HCW'99, San Juan, Puerto Rico, 1999, pp. 30–44.
[15] T. Truong-Huu, G. Koslovski, F. Anhalt, J. Montagnat, P. Vicat-Blanc Primet, Joint elastic cloud and virtual network framework for application performance-cost optimization, J Grid Comput. 9 (1) (2011) 27–47.
[16] M. Wieczorek, R. Prodan, T. Fahringer, Scheduling of scientific workflows in the ASKALON grid environment, ACM SIGMOD 34 (3) (2005) 56–62.
[17] J. Yu, R. Buyya, A taxonomy of workflow management systems for grid computing, J Grid Comput. 3 (3–4) (2005) 171–200.
[18] J. Yu, R. Buyya, Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms, Sci. Program. J. 14 (3–4) (2006) 217–230.
[19] X. Meng, V. Pappas, L. Zhang, Improving the scalability of data center networks with traffic-aware virtual machine placement, in: INFOCOM 2010, San Diego, California, USA, 2010, pp. 1154–1162.
[20] J. Jiang, T. Lan, S. Ha, M. Chen, M. Chiang, Joint VM placement and routing for data center traffic engineering, in: IEEE INFOCOM 2012, Orlando, FL, USA, 2012, pp. 2876–2880.
[21] J. Lee, M. Lee, L. Popa, Y. Turner, S. Banerjee, P. Sharma, B. Stephenson, CloudMirror: application-aware bandwidth reservations in the cloud, in: HotCloud 2013, San Jose, CA, USA, 2013, pp. 1–6.
[22] L. Gong, W. Zhao, Y. Wen, Z. Zhu, Dynamic transparent virtual network embedding over elastic optical infrastructures, in: IEEE ICC 2013, Budapest, Hungary, 2013, pp. 3466–3470.
[23] A. Pagès, J. Perelló, S. Spadaro, J. García-Espín, J. Riera, S. Figuerola, Optimal allocation of virtual optical networks for the future internet, in: ONDM 2012, Colchester, UK, 2012, pp. 1–6.
[24] R. Srinivas, S. Hegde, D.M. Divakaran, M. Gurusamy, Virtual network embedding in hybrid datacenters with dynamic wavelength grouping, in: CloudCom 2014, Singapore, 2014, pp. 905–910.
[25] N. Farrington, G. Porter, S. Radhakrishnan, H.H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, A. Vahdat, Helios : a hybrid electrical/optical switch architecture for modular data centers, in: ACM SIGCOMM 2010, New Delhi, India, 2010, pp. 339–350.
[26] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, Y. Chen, OSA : an optical switching architecture for data center networks with unprecedented flexibility, IEEE/ACM Trans. Netw. 22 (2) (2014) 498–511.
[27] T. Truong-Huu, J. Montagnat, Virtual resources allocation for workflow-based applications distribution on a cloud infrastructure, in: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid 2010), Melbourne, Australia, 2010, pp. 612–617.
[28] T. Ma, R. Buyya, Critical-path and priority based algorithms for scheduling workflows with parameter sweep tasks on global grids, in: IEEE SBAC-PAD 2005, Rio de Janeiro, Brazil, 2005, pp. 251–258.
[29] M. Rahman, S. Venugopal, R. Buyya, A dynamic critical path algorithm for scheduling scientific workflow applications on global grids, in: e-Science 2007, Bangalore, India, 2007, pp. 35–42.
[30] J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal, K. Kennedy, Task scheduling strategies for workflow-based applications in grids, in: CCGrid'05, Cardiff, UK, 2005, pp. 759–767.
[31] A. Mandal, K. Kennedy, C. Koelbel, G. Marin, J. Mellor-Crummey, B. Liu, L. Johnsson, Scheduling strategies for mapping application workflows onto the grid, in: HPDC-14, NC, USA, 2005, pp. 125–134.
[32] J. Rojas Balderrama, T. Truong-Huu, J. Montagnat, Scalable and resilient workflow executions on production distributed computing infrastructures, in: ISPDC 2012, Germany, 2012, pp. 119–126.
[33] P. Vicat-Blanc Primet, V. Roca, J. Montagnat, J.-P. Gelas, O. Mornard, L. Giraud, G. Koslovski, T. Truong-Huu, A scalable security model for enabling dynamic virtual private execution infrastructures on the internet, in: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2009), Shanghai, China, 2009, pp. 348–355.

Tram Truong-Huu (M12-SM15) received the MS degree from the Francophone Institute for Computer Science, Hanoi, Vietnam and the PhD degree in computer science from the University of Nice Sophia Antipolis, France in October 2007 and December 2010, respectively. He held a post-doctoral fellowship at the French National Center for Scientific Research (CNRS) from January 2011 to June 2012. He joined the National University of Singapore in July 2012, where he is currently a research fellow in the Department of Electrical and Computer Engineering. His research interests include scientific workflows, grid, cloud and mobile computing, and networks. He won the Best Presentation Recognition at IEEE/ACM UCC 2013. He has been a member of the IEEE since 2012 and senior member since 2015.

Mohan Gurusamy (M00-SM07) received the PhD degree in Computer Science and Engineering from the Indian Institute of Technology, Madras in 2000. He joined the National University of Singapore in June 2000, where he is currently an Associate Professor in the Department of Electrical and Computer Engineering. He is currently serving as an editor for IEEE Transactions on Cloud Computing, Elsevier Computer Networks journal and Springer Photonic Network Communications journal. He has served as the lead guest editor for two special issues of the IEEE Communications Magazine (OCS), August 2005 and November 2005, and as a co-guest editor for a special issue of the Elsevier Optical Switching and Networking journal, November 2008. He served as a TPC Co-Chair for several conferences including IEEE ICC 2008 (ONS). His research interests are in the areas of Cloud Data Center networks, Software Defined Networks, and Optical Networks. He has about 160 publications to his credit including two books and three book chapters in the area of optical networks. He has been a member of the IEEE since 2000 and senior member since 2007.

Vishal Girisagar received the MS degree in Communications Engineering from the National University of Singapore in 2015. He is now a software engineer developing applications for embedded systems at Continental Automotive Singapore Pte Ltd, Singapore. He has more than 4 years of work experience in embedded software and computer networking with leading telecom companies.