Inter-Datacenter Job Routing and Scheduling with Variable Costs and Deadlines

Carlee Joe-Wong, *Student Member, IEEE,* Ioannis Kamitsos, Sangtae Ha, *Senior Member, IEEE*

***Abstract*—To reduce their operational costs, datacenter operators can schedule large jobs at datacenters in different geographical locations with time- and location-varying electricity and bandwidth prices. We introduce a framework and algorithms to do so that minimize electricity and bandwidth cost subject to job indivisibility, deadlines, priorities, and datacenter resource constraints. In doing so, we provide a way for datacenter operators to predict their operational costs for different datacenter placements and capacities, and thus make informed decisions about how to expand their datacenter network. Our distributed algorithm uses estimated job arrivals and day-ahead electricity prices to optimize over sliding time windows. We demonstrate its effectiveness on a Google datacenter trace and investigate the effects of different cost and performance criteria. The algorithm leverages heterogeneous job resource requirements and routing and scheduling flexibility: even deadline and indivisibility constraints yield little cost increase, though they significantly improve job completion times and localization at only one datacenter respectively. We show that our algorithm reduces the cost much more than optimizing only electricity, only bandwidth, or a combination of resource costs and job completion times.**

I. Introduction

THE expansion of the Internet has fueled a rise in cloud computing, with computational "jobs" sent to and run at remotely located datacenters [1]. The popularity of cloud services is such that Google and Microsoft spent $3.4 billion to upgrade their datacenter infrastructure in the second quarter of 2013 [2]. As this infrastructure expands, operators need to manage jobs arriving at many large server clusters in different geographical locations [3]. Reducing datacenter operational expenses promises significant savings [1], but equally importantly, operators must decide how to provision resources at different locations in the datacenter network.

While much research on this topic has focused on reducing electricity spending, recent studies have found comparable electricity and bandwidth costs [1]. Thus, we consider both electricity costs and ingress and egress bandwidth costs from a job's location to a given datacenter. We exploit variations in electricity and bandwidth prices at different datacenter locations and times by routing jobs to cheaper datacenters and scheduling them at inexpensive times, subject to constraints on job performance. Figure 1 illustrates this idea of routing and scheduling jobs. We provide a framework and algorithms to study the tradeoffs between electricity and bandwidth cost reduction as well as performance criteria.

C. Joe-Wong and I. Kamitsos: Princeton University, Princeton, NJ (cjoe@princeton.edu,kamitsos@alumni.princeton.edu).

S. Ha: University of Colorado, Boulder, CO (sangtae.ha@colorado.edu).

Manuscript received April 16, 2014, revised April 23, 2015, accepted June 6, 2015.

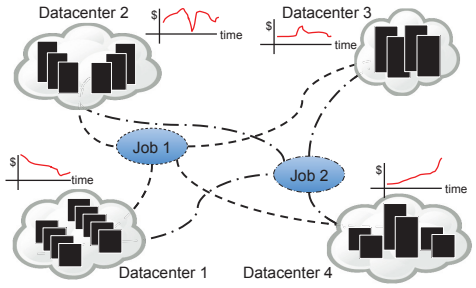


Fig. 1: Jobs from different locations can be routed to datacenters with different electricity and bandwidth cost profiles.

We demonstrate our algorithm performance on a Google datacenter trace, finding that the operator can exploit heterogeneous electricity and bandwidth requirements to optimize its costs, even when required to satisfy job deadlines or datacenter routing constraints. These insights can help operators determine where to add datacenters to its network. By computing a given network configuration's operational costs and job performance for a range of plausible future job patterns, the operator can determine the "best" configuration, or combination of datacenter locations and capacities, for its future needs. Our work therefore solves two problems for datacenter operators: minimizing day-to-day operational costs and optimally expanding their datacenter network.

Optimizing operational costs must be balanced with job performance requirements: for instance, access requests for web hosting, as considered in [4] and [5], require very fast response times and might be routed so as to minimize latency, even at the expense of large electricity costs. In this paper, we instead focus on larger jobs like video encoding or scientific computations. Since larger jobs are more expensive, they offer greater potential for cost savings than do smaller jobs. In this work we develop and numerically validate a novel framework that minimizes the costs of large jobs using these unique flexibilities. Our work is the first to account for and meet the following unique challenges for large datacenter jobs:[1]

- *Job dispatchability:* Some larger jobs may be divided into tasks that require intensive inter-task communication. These should then be scheduled at only one datacenter, to reduce the amount of inter-datacenter traffic and delays. Though this constraint introduces discrete elements into the cost minimization problem, we present an efficient algorithm to find a near-optimal solution.
- *Scheduling over long time periods:* Larger jobs with deadlines over one hour can occupy resources for a long period of time. Our algorithm incorporates the resulting

[1]We do not consider lower-level performance within a datacenter, e.g., fairness, as our focus is on *inter*-datacenter job allocation.

overlap between jobs arriving at different times, ensuring that the datacenter has sufficient capacity to handle all of the jobs assigned to it at a given time. To do so, we use a sliding window optimization that incorporates predictions of future job arrivals and resource requirements.

- *Job deadlines:* Larger jobs typically have runtimes over several minutes, and thus are more concerned with overall completion times, e.g., to satisfy service level agreements (SLAs), than with network latency, which is on the order of seconds. We incorporate completion time guarantees as constraints and study their implications in Section V.
- *Scalability:* The large number (on the order of several thousand or million) of jobs that must be scheduled at any given time necessitates efficient, scalable algorithms for computing the optimal routing and scheduling.

Our numerical evaluation shows that our proposed algorithm leverages heterogeneous job resource requirements to optimize costs, overcoming job dispatchability and deadline constraints: relaxing these constraints significantly affects job performance, but has little effect on cost. Comparing our results with optimizing only bandwidth or only electricity cost shows that we extract most of each resource's potential savings. Comparing the results with a heuristic that optimizes job completion times and resource costs shows that we fully leverage jobs' time flexibility to reduce their total cost.

We next discuss related works in Section II. We construct our analytical framework in Section III and develop algorithms to minimize operators' costs in Section IV. We evaluate the effects of different cost and performance criteria on a Google datacenter trace in Section V. Section VI concludes the paper. All proofs are included in Appendix A.

II. Related Work

Much recent research has considered minimizing electricity usage in datacenters. Many approaches do not explicitly consider prices [6], e.g., traffic engineering and routing [7], [8], speed scaling techniques [9], process migration and network virtualization [10], exploiting low power sleeping states [11], [12], and dynamic resource provisioning [13]. Another tradeoff often examined is that between minimizing electricity usage and provisioning multiple servers and devices to accommodate peak traffic loads within a given datacenter [14]. Other works explicitly model the energy used in data communication [15].

Some research has considered reducing datacenters' environmental impact instead of minimizing electricity costs. For instance, [4] considers the three-way tradeoff between environmental, latency and electricity costs, while [16] uses intelligent routing to increase the use of renewable energy while accounting for electricity prices at different datacenters. A similar objective is considered in [17], which attempts to optimize the fraction of energy that comes from renewable sources, and in [18], which includes carbon costs in electricity costs. Most datacenter operators, however, are likely motivated more by cost reduction than sustainability concerns.

Other works attempt to minimize electricity cost. For example, [19], [20] minimize a cloud service provider's total electricity cost under quality of service constraints, while [21] provides a framework to decide where to build data centers. The authors of [22] apply energy-aware routing techniques, such as routing jobs to the cheapest data centers within a distance threshold or to the closest data centers with electricity prices below a given threshold. Other works consider tradeoffs between long-term average [23] and real-time [5] electricity and bandwidth costs for short-term job requests. We consider similar tradeoffs, but focus on larger jobs, which may require scheduling over multiple timeslots during which electricity prices and overall job demands may change subject to deadline constraints. We also show how our algorithm can be used for longer-term resource provisioning and network planning.

III. Framework and Problem Formulation

Suppose that an operator has K datacenters. We consider a series of discrete timeslots $t = 1, 2, \ldots$, e.g., each lasting one hour or ten minutes. Jobs are submitted at the beginning of each timeslot, and we associate each job $i = 1, 2, \ldots, N$ with an arrival time s_i and deadline d_i: the job must finish processing by time $s_i + d_i$. We assume throughout the paper that resources in a datacenter are provisioned for a job according to its specific requirements, e.g., by dividing resources into fine-granular virtual machines, so that each job takes up only the resources that it requires. For instance, some jobs may be more CPU-intensive than others.

For simplicity, we suppose that all jobs can be continuously divided into tasks: for each job i, we use $x_{ik}(t) \in [0, 1]$ to denote the fraction of job i routed to datacenter k at time t, and $\mathbf{x}$ to denote the concatenated vector of the $x_{ik}(t)$. Infinite divisibility is an idealized assumption, but many jobs can be divided into several smaller tasks: in our evaluation dataset (Section V), jobs are divided into up to 5000 tasks.

We consider two different types of jobs, dispatchable and non-dispatchable. *Dispatchable* jobs can be routed to different datacenters at different times without increasing the need for inter-datacenter communication, e.g., computations broken into several parallelizable tasks. *Non-dispatchable* jobs are defined as those that must run at a single datacenter, e.g., due to intensive inter-task communication, as in computational jobs with several sequential steps that can be started and stopped if run in order. Thus, non-dispatchable jobs may still be scheduled in different timeslots. We use I_d to denote the set of dispatchable and I_n the set of non-dispatchable jobs. We show in Section V that our algorithm schedules most jobs, even dispatchable ones, to complete on just one datacenter.

We take $t = 0$ to be the present timeslot and consider $t = 1, 2, \ldots, T$, e.g., $T = 24$ for a day of one-hour timeslots. The consequences of optimizing over only this finite time window are discussed in greater depth in Section IV-C. The operator's objective is to minimize its electricity and bandwidth costs, while ensuring that each job finishes before its deadline and that each datacenter has sufficient resources to handle all of its jobs in any given timeslot. Optional priority factors are used to prioritize completion of especially important jobs.

A. Optimization Constraints

We first examine the limits of possible routing and scheduling decisions by making explicit the optimization constraints.

1) Resource Constraints: Computational jobs require many non-substitutable resources, e.g., bandwidth, CPU, and memory. We consider two: electricity, which includes both CPU power and associated costs like cooling and is approximately proportional to CPU time [22]; and bandwidth, i.e., limited capacity on the datacenter ingress/egress link [5].[2] Bandwidth can be a capacity bottleneck in practice [24], while electricity overloads have caused many datacenter outages [25]. We use $P_k(t)$ and $B_k(t)$ to denote the finite electricity and bandwidth capacities for datacenter k at time t.

Suppose that each job has fixed requirements e_i for electricity and b_i for bandwidth resources. These requirements scale linearly with the fraction of job processed; thus, job i requires $e_i x_{ik}(t)$ electricity and $b_i x_{ik}(t)$ bandwidth from datacenter k at time t.[3] The bandwidth costs model a job's data coming into and results coming out of the datacenter where it is processed. Processing a dispatchable job at multiple datacenters is assumed, as in the definition of dispatchable jobs, not to introduce additional bandwidth requirements. We then have the resource constraints

$$\sum_{i=1}^{N} e_i x_{ik}(t) \le P_k(t),\ \sum_{i=1}^{N} b_i x_{ik}(t) \le B_k(t). \tag{1}$$

2) Job Deadlines: Since each job i must be completed before its deadline $s_i + d_i$, we introduce the constraints

$$\sum_{k=1}^{K} \sum_{t=s_i}^{s_i+d_i} x_{ik}(t) = 1 \tag{2}$$

for all $i \in I_d \cup I_n$ into the operator's scheduling and routing problem. Essentially, each job must be scheduled completely, with no more resources allocated than it can use.

3) Non-Dispatchable Jobs: Since a non-dispatchable job must be scheduled only at one datacenter, we must have $x_{ik}(t) = 0$ for all $k \neq k_i$ for some datacenter k_i if $i \in I_n$. Thus, we constrain $\sum_{t=1}^{T} x_{ik}(t) \in \{0, 1\}$ for all datacenters k; since $\sum_{t,k} x_{ik}(t) = 1$, these constraints ensure that job i runs only on one (unspecified) datacenter. Equivalently, we define variables $z_{ik} = \sum_{t=1}^{T} x_{ik}(t)$ for job $i \in I_n$ and datacenter k to obtain the constraints

$$\sum_{t=1}^{T} x_{ik}(t) = z_{ik},\ z_{ik} \in \{0, 1\}\ \forall i \in I_n;\ 1 \le k \le K. \tag{3}$$

B. Operator Objective

We now consider the operator's objective, which includes electricity cost, bandwidth cost, and job prioritization.

1) Electricity Costs: We assume, as is often the case, that each datacenter's regional utility company offers day-ahead electricity prices [26], and denote datacenter k's cost at time t by $c_k(t)$.[4] The operator then has a total electricity cost of

$$\sum_{k=1}^{K} \sum_{t=1}^{T} c_k(t) \sum_{i=1}^{N} e_i x_{ik}(t). \tag{4}$$

2) Bandwidth Costs: Different jobs may originate at different customer locations, introducing location-varying bandwidth costs for transporting the job's data and results from physically distant datacenters. Letting p_{ik} denote job i's unit bandwidth cost for datacenter k, the bandwidth cost is

$$\sum_{i=1}^{N} \sum_{k=1}^{K} p_{ik} \sum_{t=1}^{T} b_i x_{ik}(t). \tag{5}$$

We use job- and datacenter-specific transport costs to reflect the location-varying bandwidth costs and assume a single bottleneck access link for each datacenter [5]. In reality, this link may encompass multiple equal-cost paths.

3) Job Priorities: While deadline constraints ensure satisfaction of completion time requirements, the operator may wish to additionally prioritize some jobs, e.g., to reward loyal customers if possible given other jobs' deadline constraints, which may be guaranteed by SLA contracts. Thus, we introduce factors $\gamma_i(t)$ to model the priority of job i at time t. Generally, we take $\gamma_i(t_1) \ge \gamma_i(t_2)$ if $t_1 \le t_2$ in order to encourage each job i to complete as early as possible; similarly, if $\gamma_{i_1}(t) > \gamma_{i_2}(t)$, then the operator has more incentive to complete job i_1 than job i_2 at time t. This priority term is then $\sum_{i=1}^{N} \sum_{t=1}^{T} \gamma_i(t) \sum_{k=1}^{K} x_{ik}(t)$. We note that shifting all of the $\gamma_i(t)$ by a constant value does not affect the optimization due to the constraint (2). In particular, taking all the $\gamma_i(t)$ equal to a constant is equivalent to minimizing the electricity and bandwidth costs. If multiple cost-minimizing solutions exist, including job priorities can bias the algorithm to a solution with lower job completion times.

4) The Optimization Problem: The operator wishes to maximize the sum of the priority factors, less the electricity and bandwidth costs. Adding (4–5) to find this objective and incorporating the constraints (1-3), the operator solves

$$\max_{x_{ik}(t), z_{ik}} \sum_{i=1}^{N} \sum_{t=1}^{T} \gamma_i(t) \sum_{k=1}^{K} x_{ik}(t) - \sum_{k,t=1}^{K,T} c_k(t) \sum_{i=1}^{N} e_i x_{ik}(t) - \sum_{i,k=1}^{N,K} p_{ik} \sum_{t=1}^{T} b_i x_{ik}(t) \tag{6}$$

$$\text{s.t.} \sum_{i=1}^{N} e_i x_{ik}(t) \le P_k(t), \sum_{i=1}^{N} b_i x_{ik}(t) \le B_k(t) \forall\ k, t \tag{7}$$

$$\sum_{k=1}^{K} \sum_{t=s_i}^{s_i+d_i} x_{ik}(t) = 1\ \forall i \in I_n \cup I_d \tag{8}$$

$$\sum_{t=1}^{T} x_{ik}(t) - z_{ik} = 0,\ z_{ik} \in \{0, 1\}\ \forall\ k;\ i \in I_n \tag{9}$$

$$x_{ik}(t) \in [0, 1];\ x_{ik}(t) = 0 \text{ if } t < s_i, t > s_i + d_i \tag{10}$$

[2]We emphasize that these constraints could also represent other resources, and that additional resource constraints do not conceptually change the model. We do not explicitly minimize data transfer latency or propagation delay as they are negligible compared to overall job completion times.

[3]Electricity costs can vary with CPU speed; however, we assume for simplicity that CPU speeds are uniform. Varying speeds for different datacenters can be easily incorporated by making the marginal costs e_i depend on the datacenter k, which does not affect our solution algorithms.

[4]If the utility company instead uses real-time spot prices, then the prices over the next day can be predicted using historical price data [27].

The constraints (10) ensure that jobs are scheduled between their arrival and completion deadlines.

The problem (6–10) is a mixed-integer linear optimization, with the $x_{ik}(t)$ continuous and z_{ik} binary variables. Most known algorithms for such problems will not scale well to the several thousand or million $x_{ik}(t)$ variables. We turn to these challenges and our proposed solutions next.

IV. Routing and Scheduling Algorithms

In this section, we derive algorithms to exactly solve (6–10) for dispatchable jobs (Section IV-A) and extend the solution to non-dispatchable jobs (Section IV-B). Section IV-C develops an online, sliding window version of the algorithms.

A. Distributed Optimization

We use the subgradient method [28] to distributedly solve (6-10) when all jobs are dispatchable: such an approach is computationally scalable to millions of jobs and requires little information to be passed between jobs. We define Lagrange multipliers $\lambda_k(t)$ and $\mu_k(t)$ for the constraints (7) to find the Lagrangian $L(\mathbf{x},\lambda,\mu) = F(\mathbf{x}) + \sum_{k=1}^{K}\sum_{t=1}^{T}\big(\lambda_k(t)P_k(t) + \mu_k(t)B_k(t) - \sum_{i=1}^{N}\left(\lambda_k(t)e_i + \mu_k(t)b_i\right)x_{ik}(t)\big)$, where F denotes the value of (6). Following standard duality theory, we then define the optimal multipliers as those minimizing $G(\lambda,\mu) = \max_{\mathbf{x}\in X} L(\mathbf{x},\lambda,\mu)$, where X denotes the set of $\mathbf{x}$ that satisfy (8–10). We then use G's subgradient $\left[\mathbf{g}_\lambda^{(0)}, \mathbf{g}_\mu^{(0)}\right]^+$ at each k to evolve λ and μ towards their optimal values.

The main difficulty in implementing this algorithm is to find the subgradients $\mathbf{g}^{(j)}$ at each iteration j by evaluating $\partial L/\partial(\lambda,\mu)$ at $\mathbf{x}^{(j)} = \text{argmax}_{\mathbf{x}\in X} L(\mathbf{x},\lambda,\mu)$. Since $\mathbf{x}$ is a vector with NKT variables, this represents a large-scale optimization problem. In our case, however, we can decompose the optimization into N subproblems, each with KT variables.

Since the Lagrangian is linear in $\mathbf{x}$, finding its maximum with respect to $\mathbf{x}$ is equivalent to solving

$$\max_{\mathbf{x}_{ik}(t)\in X} \sum_{k=1}^{K}\sum_{t=1}^{T}\beta_{ik}(t)x_{ik}(t) \tag{11}$$

where $\beta_{ik}(t) = \gamma_i(t) - e_i c_k(t) - b_i p_{ik} - \lambda_k(t)e_i - \mu_k(t)b_i$ for each user i. Taking $\mathbf{x}^{(j)}$ to be the concatenation of the solutions to (11), we can find the solutions easily and scalably:

Lemma 1: Let $\mathbf{x}_{ik}^{(j)}(t)$ solve (11). Then $\mathbf{x}_{i\kappa}^{(j)}(\tau) = 1$ for a unique $(\kappa,\tau) = \text{argmax}_{k,t}\beta_{ik}(t)$ and 0 otherwise. Moreover, finding the subgradients $\mathbf{g}^{(j)}$ at a given iteration j has a complexity of $NKT\log(KT)$.

Since solving (11) can be done independently for each job i, the complexity of each iteration for a given job is then $KT\log(KT)$. This quantity will be fairly low as we can expect that $K \leq 100$ with at most a few hundred timeslots.

While subgradient algorithms are fully distributed, classical subgradient algorithms do not solve for optimal primal solutions. Recently, however, it was shown that the running average of the solution candidates used in the subgradient method yields near-optimal primal solutions [28]. More precisely, at each iteration j of the subgradient method, we solve for

$$\begin{aligned}\mathbf{y}_{ik}^{(j+1)}(t) &= \frac{1}{\sum_{m=0}^{j}\alpha_m}\sum_{l=0}^{j}\alpha_l\mathbf{x}_{ik}^{(l)}(t)\\ &= \frac{\sum_{m=0}^{j-1}\alpha_m}{\sum_{m=0}^{j}\alpha_m}\mathbf{y}_{ik}^{(j)}(t) + \frac{\alpha_j\mathbf{x}_{ik}^{(j)}(t)}{\sum_{m=0}^{j}\alpha_m}\end{aligned} \tag{12}$$

where each α_j is the subgradient step size at iteration j.

Algorithm 1: Modified subgradient algorithm.

$\lambda_k^{(0)}(t) \leftarrow 0$ and $\mu_k^{(0)}(t) \leftarrow 0$ for all k, t;
Use Prop. 1 to find the desired number of iterations J;
for $j = 0$ *to* J **do**
  for $i = 1$ *to* N **do**
    Find the $x_{ik}^{(j)*}(t)$ that solve (11) using Lemma 1;
  $\mathbf{g}^{(j)} \leftarrow \partial L/\partial(\lambda,\mu)|_{\mathbf{x}^{(j)*}}$;
  $\mathbf{y}^{(j)} \leftarrow \frac{\sum_{m=0}^{j-1}\alpha_m}{\sum_{m=0}^{j}\alpha_m}\mathbf{y}^{(j)} + \frac{\alpha_j\mathbf{x}^{(j)}}{\sum_{m=0}^{j}\alpha_m}$;
  $[\lambda_k(t),\mu_k(t)]^{(j+1)} \leftarrow \left[[\lambda_k(t),\mu_k(t)]^{(j)} - \alpha_j\left[\mathbf{g}_\lambda,\mathbf{g}_\mu\right]^{(j)}\right]^+$;

Proposition 1 ([28]): Suppose the Slater condition holds for (6–10) with $I_n = \emptyset$. Let q^* denote the optimal value of (6), and consider $\mathbf{y}_{ik}^{(j)}(t)$ as given by (12) for constant stepsize α. Denote violations of the constraints (7) by $f_{kt}(\mathbf{x}) = P_k(t) - \sum_{i=1}^{N} e_i x_{ik}(t)$, $b_{kt}(\mathbf{x}) = B_k(t) - \sum_{i=1}^{N} b_i x_{ik}(t)$. If λ,μ are initialized to zero, then $V_r^{(j)} \equiv \max\left\{\left|f_{kt}(\mathbf{y}^{(j)})\right|, \left|b_{kt}(\mathbf{y}^{(j)})\right|\right\} \leq \phi/(\alpha j)$ and $q^* + (q^* - F(\overline{x}))\,V_r^{(j)}/\gamma \geq F(\mathbf{y}^{(j)}) \geq q^* - \alpha L^2/2$, where $\phi = 3(q^* - F(\overline{\mathbf{x}}))/\gamma + \alpha L^2/(2\gamma) + \alpha L$, L bounds the subgradients $\mathbf{g}^{(j)}$ and $\gamma = \min_{k,t}\{f_{kt}(\overline{\mathbf{x}}), b_{kt}(\overline{\mathbf{x}})\}$ for some $\overline{\mathbf{x}} \in X$ satisfying (7) with strict inequality.

Thus, as $j \to \infty$, the averaged solution (12) satisfies the constraints, and we can come arbitrarily close to the optimum value of (6) by choosing a sufficiently small stepsize α. To bound the constraint violations by a constant ϵ, each datacenter can compute the maximum number of iterations needed to derive a solution within a threshold ϵ/K. We can then solve (6-10) using the modified subgradient method in Algorithm 1.

B. Extension to Non-Dispatchable Jobs

If we use Algorithm 1 to solve (6-10) with non-dispatchable jobs, then some of these jobs will be routed to multiple datacenters. While our Lagrange multiplier approach may be combined with branch-and-bound techniques and linear relaxations to find an optimal solution with non-dispatchability [29], in the interests of scalability we turn to an approximation algorithm instead of searching for an optimal solution.

We note that from Lemma 1, the solutions $\mathbf{x}^{(j)}$ generated by minimizing $G(\lambda,\nu)$ are in fact integer solutions. Under some conditions, this is in fact an optimal solution:

Proposition 2: Suppose that $\mathbf{x}^{(j)}$ or $\mathbf{y}^{(j)}$ generated by Algorithm 1 is feasible, i.e., it satisfies (7-10). Then if $\lambda_k(t)\left(P_k(t) - \sum_{i=1}^{N} e_i x_{ik}^{(j)}(t)\right) = 0 = \mu_k(t)\big(B_k(t) - \sum_{i=1}^{N} b_i x_{ik}^{(j)}(t)\big)$ for all datacenters k and times t (similarly for $\mathbf{y}^{(j)}$), $\mathbf{x}^{(j)}$ $\left(\mathbf{y}^{(j)}\right)$ is an optimal solution to (6-10).

These conditions may hold for $\mathbf{y}^{(j)}$ if $\text{argmax}_{k,t}\beta_{ik}(t)$ yields the same datacenter k at each iteration for all jobs

Algorithm 2: Incorporating non-dispatchable jobs.

Solve (6-8) without the constraint (9) with Algorithm 1;
$k_i \leftarrow 0$ for all $i \in I_n$;
while $\exists\, i \in I_n$ *with* $k_i = 0$ **do**
  forall the $i \in I_n$ **do**
    $k_i \leftarrow \text{argmax} \sum_{t=1}^{T} y_{ik}(t)$;
  for $k = 1$ *to* K **do**
    $s_k \leftarrow 0$ // Indicator for feasibility of (13)
    while $s_k < 1$ **do**
      Solve (13) at each datacenter k using Algorithm 1;
      if *(13) is infeasible* **then**
        $k_i \leftarrow 0$ for $i = \text{argmin}_{k_i=k} \sum_{t=1}^{T} y_{ik}(t)$;
      else
        $s_k \leftarrow 1$;
  $s \leftarrow 0$ // Indicator for feasibility of (6-8)
  while $s < 1$ **do**
    Solve (6-8) with $x_{ik}(t) = 0$ for all $k \neq k_i$ if $i \in I_n$, $k_i > 0$;
    $x_{ik}(t) = 0$ for all t if $k_i < 0$;
    if *(6-8) is infeasible* **then**
      $(\kappa, \tau) \leftarrow \text{argmax}_{k,t}\, (\lambda_k(t), \mu_k(t))$;
      $\iota \leftarrow \text{argmax}_i e_i x_{i\kappa}(\tau)$ // Or $\text{argmax}_i b_i x_{i\kappa}(\tau)$
      $k_\iota \leftarrow -1$ if $\iota \in I_n$; remove job i otherwise;
    else
      $s \leftarrow 1$;

$i \in I_n$. By Lemma 1, in that case job i runs only at datacenter k. However, if $\text{argmax}_{k,t} \beta_{ik}(t)$ changes, then non-dispatchability is not satisfied. We thus propose Algorithm 2 to adjust $\mathbf{y}^{(j)}$ "as little as possible" to find a feasible solution.

The key insight of our algorithm is that once a datacenter k_i is chosen for each job $i \in I_n$, we can decompose (6-10) into separate optimization problems for non-dispatchable jobs

$$\max_{x_{ik}(t) \in X} \sum_{t, i:k_i=k} \left(\gamma_i(t) - e_i c_k(t) - b_i p_{ik}\right) x_{ik}(t) \tag{13}$$
$$\text{s.t.} \sum_{i:k_i=k} e_i x_{ik}(t) \leq P_k(t), \sum_{i:k_i=k} b_i x_{ik}(t) \leq B_k(t)$$

at each datacenter k. Thus, in the first step of our algorithm each job $i \in I_n$ assigns itself to the datacenter k_i maximizing $\sum_t y_{ik}(t)$. Each datacenter then schedules all of its assigned non-dispatchable jobs; if this assignment is not feasible, jobs are routed to datacenters with spare resources in decreasing order of $\sum_t y_{ik}(t)$. If no feasible datacenter exists, the job is dropped. Once each datacenter has a feasible set of jobs, we optimize over *all* jobs given these assignments, using Algorithm 1 with $x_{ik}(t) = 0$ for all $k \neq k_i$, $i \in I_n$. In practice, jobs are only dropped in rare cases of infeasibility. In Section V's evaluation, no jobs were dropped.

This algorithm is scalable: the dominant complexity at each iteration is sorting non-dispatchable jobs at each datacenter (on average $|I_n|/K$ jobs) to decide which jobs to drop or reassign. In the best case, sorting is unnecessary and we again have the complexity of $KT \log(KT)$ from Algorithm 1 (Lemma 1).

C. Online Algorithm

In reality, the job allocation problem is dynamic: at each time t, new jobs arrive and new day-ahead electricity prices are released, which cannot be predicted with perfect accuracy. Thus, at each time $t > 1$, the operator must correct the solution computed at time $t-1$ and adjust the routing accordingly.

Similar multi-stage optimization problems under uncertainty arise in resource provisioning problems for many industries, where stochastic linear programming and dynamic programming approaches have been proposed. However, such approaches often lead to scalability challenges, especially for multi-stage (i.e., multi-timeslot) problems like those considered here [30]. Thus, we instead optimize over a sliding time window.[5] We compute the optimal solution for a given time window of length T, route jobs in the next timeslot accordingly, and re-compute the optimal solution for the next window of length T. Algorithm 3 formalizes this idea. We note that at each stage τ, the computed routing for $t > \tau$ is re-optimized in later stages of the algorithm.

Algorithm 3: Sliding time window algorithm.

Initialize predicted electricity prices $c_k(t)$ and estimated job arrivals s_i and deadlines d_i for $t = 1, 2, \ldots, T$;
$\tau \leftarrow 1$ // τ tracks the current time.
while $\tau > 0$ **do**
  Solve (14) for $t = \tau, \tau+1, \ldots, T+\tau-1$ using Algorithm 2;
  Update the predicted job arrivals s_i and deadlines d_i for $t = \tau+1, \ldots, T+\tau$;
  Update the resource capacities $\overline{P}_k(t)$, $\overline{B}_k(t)$ for $t = \tau+1, \ldots, T+\tau$;
  $\tau \leftarrow \tau + 1$;

This approach introduces a challenge near the end of the time interval $t = T$. New jobs will arrive during this timeslot, but many of them can be expected to have deadlines $s_i + d_i > T$. Thus, constraint (8) must be modified to allow some of these jobs to complete after time T, while ensuring that they can start before time T. In the best case, we have periodic job arrivals, which allows us to solve (6-10) as follows:

Proposition 3: Suppose that for each set of timeslots $(j-1)T+1$ to jT, $j = 1, 2, \ldots$, the operator observes the same pattern of job arrivals with the same e_i and b_i. Further, suppose that all resource capacities and electricity prices are periodic, i.e., $P_k(t) = P_k(t+T)$, $B_k(t) = B_k(t+T)$, and $c_k(t) = c_k(t+T)$ for any timeslot t. Then if at each time t, s_i and $s_i + d_i$ in (6–10) is interpreted as modulo T and $d_i \leq T$ (i.e., each job must finish within T timeslots), a solution to (6–10) maximizes the long-term time-average of (6).

In this case, periodicity allows the operator to know exactly how much of each job to reserve for timeslots after time T. We do not observe perfect periodicity in practice, but strong daily patterns do occur in electricity prices [27] and datacenter workloads [24]. We thus propose to choose a one-day sliding window T; from Prop. 3, our solution will then achieve near-optimality.[6] We use historical information to estimate the fraction of jobs to reserve for times $t > T$.

For each time $\tau = T+1, \ldots 2T$, we use $\overline{P}_k(\tau)$ to denote the expected electricity and $\overline{B}_k(\tau)$ the expected bandwidth capacity of datacenter k at time τ, less the resources used by jobs arriving after T. In the periodic case, $\overline{P}_k(\tau) = P_k(\tau) - \sum_{i:s_i<\tau-T} e_i x_{ik}(\tau - T)$, i.e., the capacity at time τ, less the (known) electricity used by jobs arriving between times $T+1$ and τ. We use historical smoothed averages to calculate $\overline{P}_k(\tau)$ and $\overline{B}_k(\tau)$. Jobs currently being scheduled (the $x_{ik}(t)$), are included, as they are the most recent arrivals and likely the

[5]The sliding window avoids an infinite horizon optimization, which requires predicting electricity prices and job arrivals indefinitely far into the future.

[6]The operator can adjust T if the dominant periodicities change.

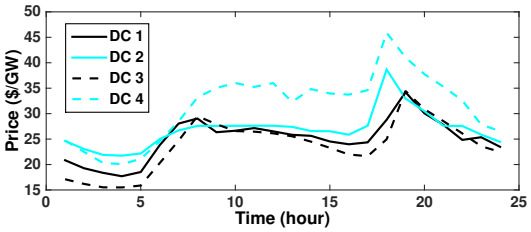


Fig. 2: Datacenter electricity prices.

best predictors. Mathematically, we compute the averages

$$a_0\left(P_k(\tau)-\sum_{s_i<\tau-T} e_i x_{ik}(\tau-T)\right)+\sum_{j=2}^{\lfloor\tau/T\rfloor} a_{j-1}\overline{P}_k(\tau-jT)$$

where the weights a_j sum to 1 and are determined by the averaging method, e.g., $a_j = 1/\lfloor\tau/T\rfloor$ for uniform averaging. Taking $a_0 = 1$, $a_j = 0$ for $j > 1$, assumes perfect periodicity. We analogously define $\overline{B}_k(\tau)$. We can now modify (6–10) to form an online optimization problem:

Proposition 4: Suppose that a given number of jobs arrives in each timeslot $1 \le t \le T$. Then given the available capacities $\overline{P}_k(\tau)$ and $\overline{B}_k(\tau)$ for $\tau > T$, the cost-minimizing $x_{ik}(t)$ solve the following optimization problem:

$$\max_{x_{ik}(t),z_{ik}} \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{t=1}^{2T}\left(\gamma_i(t)-c_k(t)e_i-p_{ik}b_i\right)x_{ik}(t) \quad (14)$$

$$\text{s.t.} \sum_{i=1}^{N} e_i x_{ik}(t) \le \left\{P_k(t), t\le T; \overline{P}_k(t), t>T\right\} \quad (15)$$

$$\sum_{i=1}^{N} b_i x_{ik}(t) \le \left\{B_k(t), t\le T; \overline{B}_k(t), t>T\right\} \quad (16)$$

$$\sum_{t=1}^{2T} x_{ik}(t) - z_{ik} = 0,\ z_{ik}\in\{0,1\}\ \forall k; i\in I_n \quad (17)$$

with the additional constraints (8) and (10). This optimization problem can be solved exactly as (6–10).

V. Evaluation

In this section, we use Section IV's algorithms on a Google datacenter trace. We find that job dispatchability, deadlines, and resource prices can significantly affect job performance, but dispatchability and deadlines have little effect on the overall cost. We use these factors to find the cost benefits of different datacenter placements in an operator's network.

A. Our Dataset

We consider 4 data centers, located in California, Massachusetts, Minnesota, and New York and referred to respectively as DCs 1, 2, 3, and 4. We use day-ahead electricity prices for the four locations from February 1, 2012 [26]. Bandwidth costs were taken from Amazon's EC2 costs for datacenter egress [3] and are assumed not to vary with time. Figure 2 shows the resulting electricity and Table I the bandwidth prices.

DCs 2 and 4 have larger electricity prices than DCs 1 and 3, and DC 2 has the highest bandwidth price.

TABLE I: Bandwidth prices.

	DC 1	DC 2	DC 3	DC 4
Price($/Gbps)	19	25	20	12

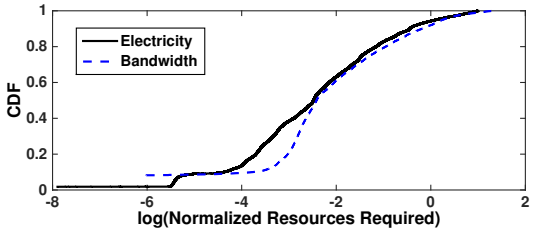


Fig. 3: Job resource requirements.

We run our algorithms on a trace of 22,383 jobs from a Google datacenter cluster in May 2011 [31].[7] Figure 3 shows the cumulative density function (CDF) of the electricity and bandwidth required for each job.[8] Datacenter capacities were calculated by randomly partitioning cluster machines among the four datacenters. The resource requirements and capacities have units of GW and Gbps. We infer jobs' non-dispatchability from each job's division into homogeneous subtasks. Most (85.6%) are non-dispatchable, though dispatchable jobs make up 61.4% of the electricity and 44.1% of the bandwidth requirements. We take a job's deadline as the time between its submission and completion. Unless otherwise noted, all jobs are assumed to have equal priorities $\gamma_i(t)$. We use Algorithm 3 with $\alpha_0 = 1$ to compute the routing and scheduling.

We first run a baseline scenario optimizing bandwidth and electricity cost subject to deadline and non-dispatchability constraints. Table II shows the comparative changes in cost for four other scenarios: no job deadlines, all jobs dispatchable or non-dispatchable, and linear priority weights on half the jobs. No jobs were dropped in any scenario. Appendix B compares our algorithm to three alternatives: optimizing only electricity or only bandwidth cost and a heuristic that includes job completion times in the objective function. Optimizing both costs extracts most of the potential electricity and bandwidth savings, and including completion times in the objective can significantly increase the total cost.

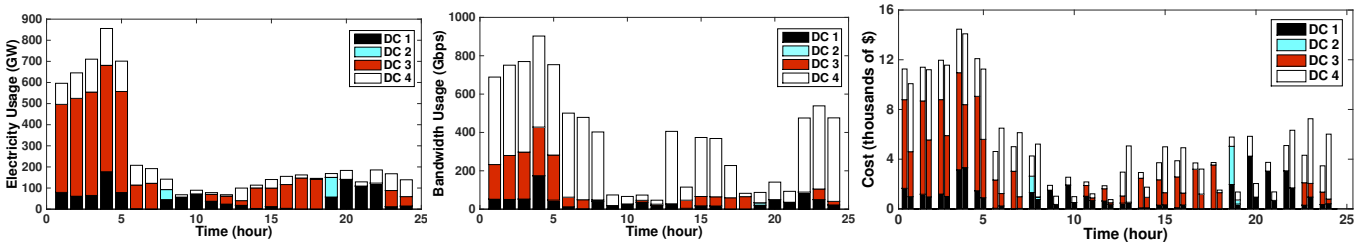
B. Job Heterogeneity, Deadlines, and Non-Dispatchability

We first illustrate the effects of job heterogeneity by examining datacenter usage and cost in the baseline scenario. Figures 4a and 4b respectively show the electricity and bandwidth used at each datacenter and timeslot. DC 2 is barely used, likely due to its high cost of bandwidth and electricity. DC 4's bandwidth usage is much higher than its electricity usage and vice versa for DC 3, likely due to DC 4's low bandwidth and DC 3's low electricity prices. Thus, *the algorithm leverages heterogeneous job resource requirements: those with higher bandwidth or electricity requirements are respectively routed to datacenters and times with cheaper bandwidth and electricity.*

Figure 4c shows the cost of electricity and bandwidth at each hour. Electricity and bandwidth costs are generally

[7]Since our electricity prices are from different dates and regions, we stress that these results are illustrative and do not exactly represent real scenarios.

[8]We use CPU and memory respectively as our electricity and bandwidth requirements, since CPU and electricity usage are approximately proportional [22] and memory roughly measures the size of a job's data. A few jobs have negligible requirements for one resource, which we round to zero.



(a) Electricity usage. (b) Bandwidth usage. (c) Electricity and bandwidth cost.

Fig. 4: Electricity and bandwidth usage and cost. In (c), the left bar at each time is the electricity and the right bandwidth cost.

TABLE II: Change in cost (%) relative to the baseline.

Scenario	Electricity Cost	B/w Cost	Overall Cost
All jobs dispatchable	-0.46	0.40	-0.043
All jobs non-dispatchable	-0.39	0.93	0.43
No deadlines	-9.00	-3.37	-6.26
Linear job priorities	0.53	-0.64	-0.041

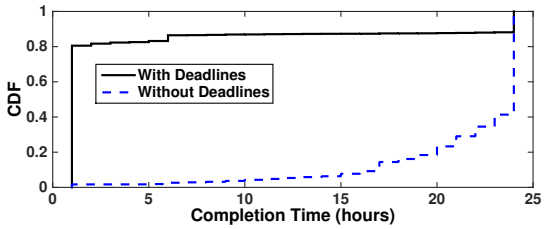


Fig. 5: Job completion times with and without deadlines.

comparable, since jobs are routed to datacenters and timeslots with cheaper prices for their resource requirements. Both types of costs are higher towards the beginning of the day, likely due to cheaper electricity prices and a large number of job arrivals at those times: 19.7% of jobs arrive at time $t = 1$.

Next, we investigate the tradeoff between worse performance and lower costs as the deadline and non-dispatchability constraints are relaxed. Though relaxing these constraints respectively increases our scheduling and routing flexibility, it does not greatly affect the cost, indicating that heterogeneity in resource requirements alone allows significant cost savings.

When all deadlines are relaxed, most jobs' completion times drastically increase: 90% of jobs complete after 16 hours, while 80% complete in an hour with deadlines. Without deadlines, jobs wait, possibly for many hours, for lower electricity prices. Figure 5 shows the CDF of these completion times. However, relaxing the deadlines only reduces the cost by 6.26% due to capacity constraints at times of low prices. *Including deadlines significantly decreases job completion times, but yields little increase in overall cost as the algorithm can still leverage datacenter routing flexibility.*

Non-dispatchability constraints have an even smaller effect on cost. From Table II, the decrease in cost from making all jobs dispatchable is almost negligible at -0.043%; even if all jobs are taken as non-dispatchable, the cost increases by 0.43%. In fact, many dispatchable jobs complete at only one datacenter. Figure 6 shows the CDF of the largest fraction $\max_k \sum_t x_{ik}(t)$ of each job i that completes at one datacenter when all jobs are dispatchable. About half–52%–of the jobs run on only one datacenter, and in the baseline scenario 93% of jobs, including over half the dispatchable jobs, complete at one datacenter. This behavior, as noted in Section IV-B, reflects a job's assignment to only one datacenter unless resource

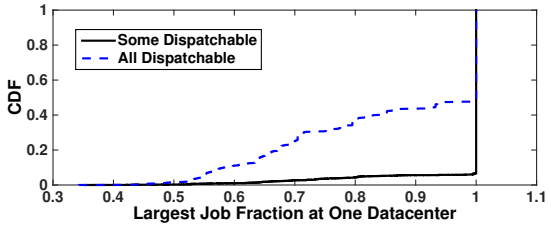
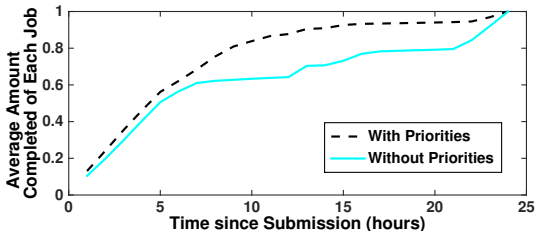
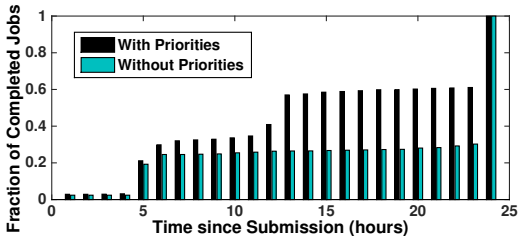


Fig. 6: Largest fraction of a job completed at one datacenter with all jobs dispatchable and only some jobs dispatchable.



(a) Average job completion.



(b) Fraction of jobs completed.

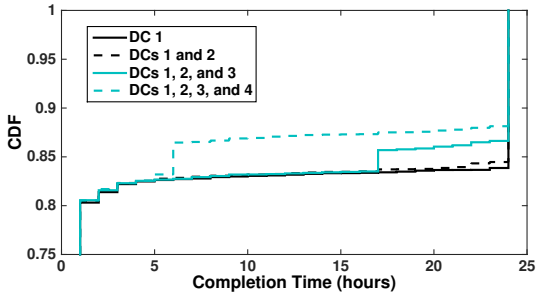
Fig. 7: Average completed fraction of each job and fraction of total jobs completed over time for jobs with and without linear priority weights that penalize longer completion times.

constraints force a change in Lemma 1's optimal $x_{ik}(t)$. Thus, *the selection of a single datacenter for each non-dispatchable job is nearly sufficient to minimize the operational cost.*

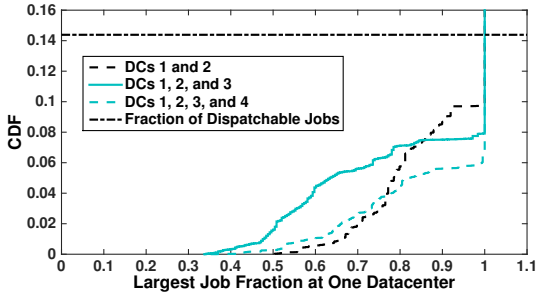
We consider the effect of priority weights by linearly penalizing longer completion times for a randomly selected 11194 jobs (i.e., the $\gamma_i(t)$ increase linearly with each hour after the job is submitted). Prioritized jobs then complete faster: Figure 7 shows the completion rates for jobs with no specified deadline with and without priorities. Both the average fraction completed of each job and the fraction of jobs completed (out of the total number of jobs with and without priorities) are larger when priorities are introduced. These shorter completion times barely change the operational cost (Table II).

C. Datacenter Provisioning

We finally show how our framework can be used to recommend the number and location of datacenters required



(a) Job completion times.



(b) Largest job fraction completed at one DC.

Fig. 8: Completion times and maximum fraction of each job completed at one datacenter with different datacenter combinations.

by an operator. We first show the cost and performance impact of adding more datacenters, and then consider different placements of a fourth datacenter.

Adding more datacenters generally decreases job completion times, as we would expect since it increases the total resources available. Figure 8a shows the job completion times as datacenters are added. While 80% of jobs complete within one hour with only DC 1, adding more datacenters can significantly reduce some jobs' completion times. Most dramatically, adding DC 4 decreases the completion time for 3% of jobs to 6 from more than 17 hours. It also yields a 15.7% increase in the fraction of jobs completed only at one datacenter (Figure 8b), though some jobs are divided among more datacenters than before. *Adding more datacenters allows some jobs to be spread among multiple datacenters, but increases the probability that a job will find and localize at a low-cost datacenter. These changes have a large impact on a small fraction of jobs.*

Since the addition of DC 4 has the most dramatic impact on job deadlines, we now consider the best location of this fourth datacenter by comparing the per-job costs of adding DC 4 in New York to adding another datacenter in Minnesota (i.e., DC 3). Figure 9a shows the CDF of the ratio of each job's cost with DCs 1–4 to that with only DCs 1–3. We see that 60% of jobs decrease their total cost by up to 40% when DC 4 is added, with no job increasing its cost. Some jobs' electricity cost increases, but the bandwidth cost always decreases: DC 4 has the lowest bandwidth, but highest electricity costs (Table I and Figure 2). DC 3, however, has higher bandwidth but lower electricity costs than DCs 1 and 2. Figure 9b shows the ratio of each job's cost when the DC 3 copy is added to DCs 1–3. No job reduces its cost by more than 28%, though 23.8% of jobs reduce their total and 31.1% their electricity costs.

Adding DC 4 saves more money than the DC 3 copy: adding DC 4 to DCs 1–3 reduces the cost by 14.1%, but a copy of DC 3 reduces the cost by only 2.2%. However, the DC 3 copy has lower job completion times. Figure 9c shows the distribution of completion times for DCs 1–3, DCs 1–4, and DCs 1–3 + a copy of DC 3. Adding DC 3 reduces the completion time by 8 hours for 4.3% of jobs relative to adding DC 4, and adding DC 4 reduces the job completion time for about 4% of jobs by 11 hours relative to DCs 1–3. Thus, *different datacenter configurations can lead to a tradeoff between shorter job completion times and lower costs, as shown by the scenarios of adding DCs 3 and 4 respectively.*

VI. Conclusion

We consider the problem of routing and scheduling large jobs to datacenters in different geographical locations at different times, taking into account time- and location-varying electricity and bandwidth prices. Our distributed algorithm routes and schedules jobs so as to minimize electricity and bandwidth costs, subject to non-dispatchability and deadline constraints. The algorithm is of low complexity, yielding efficient and scalable solutions to this large-scale problem. We incorporate future prices and job arrival predictions by optimizing over a sliding time window.

We demonstrate our algorithm on a trace of 22,383 jobs from a Google datacenter. The algorithm adapts to job non-dispatchability and deadline constraints by respectively leveraging time and routing flexibility, yielding small changes in overall cost. Including these constraints, however, improves job completion times and localization at only one datacenter. Prioritizing lower completion times for some jobs improves their completion times with little increase in cost. We also compare operational cost and job completion times with different datacenter combinations, finding a tradeoff between reducing costs and completion times. Our work thus provides a framework and algorithms for operators to predict the operational costs and job performance when routing and scheduling large jobs, allowing them to make informed decisions about how to expand their datacenter network.

References

[1] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.

[2] D. Harris, "Google and Microsoft spent a combined $3.4b on infrastructure last quarter," GigaOm, 2013, http://gigaom.com/2013/07/18/google-and-microsoft-spent-a-combined-3-4b-on-infrastructure-last-quarter/.

[3] Amazon, "EC2 pricing," 2013, http://aws.amazon.com/ec2/pricing/.

[4] P. X. Gao, A. R. Curtis, B. Wong, and S. Keshav, "It's not easy being green," in *Proc. of SIGCOMM*. ACM, 2012, pp. 211–222.

[5] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *Proc. of INFOCOM*. IEEE, 2013.

[6] C. Ge, Z. Sun, and N. Wang, "A survey of power-saving techniques on data centers and content delivery networks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1334–1354, 2013.

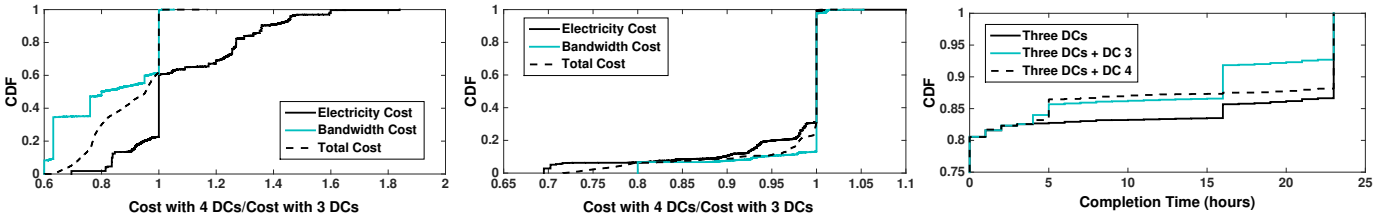
[7] L. Chiaraviglio, M. Mellia, and F. Neri, "Reducing power consumption in backbone networks," in *Proc. of ICC*. IEEE, 2009, pp. 1–6.

[8] N. Vasić and D. Kostić, "Energy-aware traffic engineering," in *Proc. of the 1st International Conference on Energy-Efficient Computing and Networking*. ACM, 2010, pp. 169–178.

[9] A. Wierman, L. L. Andrew, and A. Tang, "Power-aware speed scaling in processor sharing systems," in *Proc. of INFOCOM*. IEEE, 2009, pp. 2007–2015.

[10] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *Computer Communication Review*, vol. 38, no. 2, pp. 17–29, 2008.

[11] I. Kamitsos, L. Andrew, H. Kim, S. Ha, and M. Chiang, "Better energy-delay tradeoff via server resource pooling," in *Proc. of ICNC*. IEEE, 2012, pp. 611–616.



(a) Change in job costs with DC 4. (b) Change in job costs with a copy of DC 3. (c) Job completion times.

Fig. 9: Change in costs and completion times for all jobs when DC 4 or a copy of DC 3 is added to DCs 1, 2, and 3.

[12] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: Eliminating server idle power," in *Proc. of ASPLOS*. ACM, 2009, pp. 205–216.

[13] V. Mathew, R. K. Sitaraman, and P. Shenoy, "Energy-aware load balancing in content delivery networks," in *Proc. of IEEE INFOCOM*. IEEE, 2012, pp. 954–962.

[14] D. Kliazovich, P. Bouvry, and S. U. Khan, "DENS: Data center energy-efficient network-aware scheduling," in *Proc. of GreenCom*. IEEE, 2010, pp. 69–75.

[15] N. Cordeschi, M. Shojafar, and E. Baccarelli, "Energy-saving self-configuring networked data centers," *Computer Networks*, vol. 57, no. 17, pp. 3479–3491, 2013.

[16] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. Andrew, "Greening geographical load balancing," in *Proc. of SIGMETRICS*. ACM, 2011, pp. 233–244.

[17] Y. Zhang, Y. Wang, and X. Wang, "Greenware: Greening cloud-scale data centers to maximize the use of renewable energy," in *Middleware 2011*. Springer, 2011, pp. 143–164.

[18] X. Zheng and Y. Cai, "Energy-aware load dispatching in geographically located internet data centers," *Sustainable Computing: Informatics and Systems*, vol. 1, no. 4, pp. 275–285, 2011.

[19] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment," in *Proc. of INFOCOM*. IEEE, 2010, pp. 1–9.

[20] V. Mathew, R. K. Sitaraman, and P. J. Shenoy, "Reducing energy costs in internet-scale distributed systems using load shifting." in *Proc. of IEEE COMSNETS*, 2014, pp. 1–8.

[21] Í. Goiri, K. Le, J. Guitart, J. Torres, and R. Bianchini, "Intelligent placement of datacenters for Internet services," in *Proc. of ICDCS*. IEEE, 2011, pp. 131–142.

[22] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for Internet-scale systems," in *Proc. of SIGCOMM*. ACM, 2009, pp. 123–134.

[23] Y. Li, H. Wang, J. Dong, J. Li, and S. Cheng, "Operating cost reduction for distributed internet data centers," in *Proc. of IEEE/ACM CCGrid*. IEEE, 2013, pp. 589–596.

[24] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with Netstitcher," in *Proc. of SIGCOMM*. ACM, 2011, pp. 74–85.

[25] P. McLaughlin, "Causes and costs of data center outages," Cabling: Installation and Maintenance, 2011, http://www.cablinginstall.com/articles/print/volume-19/issue-7/features/causes-and-costs-of-data-center-outages.html.

[26] Federal Energy Regulatory Commission, "Electric power markets: National overview," 2012, http://www.ferc.gov/market-oversight/mkt-electric/overview.asp.

[27] A.-H. Mohsenian-Rad and A. Leon-Garcia, "Optimal residential load control with price prediction in real-time electricity pricing environments," *IEEE Trans. on Smart Grid*, vol. 1, no. 2, pp. 120–133, 2010.

[28] A. Nedic and A. Ozdaglar, "Approximate primal solutions and rate analysis for dual subgradient methods," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1757–1780, 2009.

[29] J. T. Linderoth and M. W. Savelsbergh, "A computational study of search strategies for mixed integer programming," *INFORMS Journal on Computing*, vol. 11, no. 2, pp. 173–187, 1999.

[30] G. A. Godfrey and W. B. Powell, "An adaptive dynamic programming algorithm for dynamic fleet management, i: Single period travel times," *Transportation Science*, vol. 36, no. 1, pp. 21–39, 2002.

[31] Google, "Cluster data," May 2011, https://code.google.com/p/googleclusterdata/.

APPENDIX A
PROOFS

A. Lemma 1

Suppose that the optimal solution has $\mathbf{x}_{ik}^{(j)}(\tau) = a < 1$. Then the optimal value of (11) would be $a \max_{k,t} \alpha_{ik}(t) + (1-a) \max_{(k,t)\neq(\kappa,\tau)} \alpha_{ik}(t) \leq \max_{k,t} \alpha_{ik}(t)$, the optimal value with $\mathbf{x}_{ik}^{(j)}(t)$.

Finding the subgradient consists of finding, for each job i, the maximum of KT scalars. Standard selection algorithms for this search have complexity $O(KT\log(KT))$, so the overall complexity of finding the subgradients is $O(NKT\log(KT))$. Updating the multiplier values, which is done with one addition per multiplier, has complexity $3KT$, so the overall complexity is $O(NKT\log(KT))$.

B. Proposition 2

From duality theory, $G(\lambda^*, \mu^*)$ is an upper bound on the optimal value of (6). Thus, if the difference $\lambda_k(t)\left(P_k(t) - \sum_{i=1}^N e_i x_{ik}^{(j)}(t)\right) + \mu_k(t)\left(B_k(t) - \sum_{i=1}^N b_i x_{ik}^{(j)}(t)\right) = 0$ between $G(\lambda^*, \mu^*)$ and $F(\mathbf{x}^{(j)}$ is zero, then $\mathbf{x}^{(j)}$ is an optimal solution.

C. Proposition 3

The long-term average of (6) at the optimal solution $\mathbf{x}^*$ may be written (with slight abuse of the summation notation) as

$$\lim_{J\to\infty} \frac{1}{J} \sum_{j=1}^{J} \left(\sum_{i,k,t=1}^{N,K,T} \big(\gamma_i(t) - c_k(t) - f_{ik}\big) x_{ik}^*(t + (j-1)T) \right).$$

For $t = 1, 2, \ldots, T$ and $j = 1, 2, \ldots$, define $y_{ik}(t+(j-1)T) = \lim_{J\to\infty} \frac{1}{J} \sum_{j=1}^{J} x_{ik}^*(t + (j-1)T)$. The idea of the proof is to show that $\mathbf{y}$ is an optimal solution.

It is straightforward to see that the objective function value with $\mathbf{y}$ is the same as that with $\mathbf{x}^*$. Thus, we need only check that $\mathbf{y}$ satisfies all relevant constraints. We first note that (9) and (10) are clearly satisfied. Since $\mathbf{y}$ is periodic with respect to T, we need only check the resource constraints (7) for $t = 1, 2, \ldots, T$:

$$\sum_{i=1}^{N} f_i y_{ik}(t) = \lim_{J\to\infty} \frac{1}{J} \sum_{j=1}^{J} \sum_{i=1}^{N} f_i x_{ik}^*(t + (j-1)T) \leq F_k(t)$$

and similarly for the other resource constraints. Since job arrivals and deadlines are periodic, it also follows that (8) is satisfied and that all jobs complete by their deadline. Thus, there exists a periodic solution.

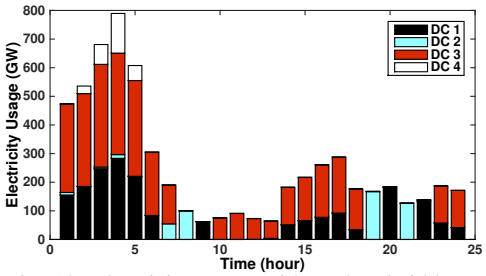


Fig. 10: Electricity usage with no bandwidth costs.

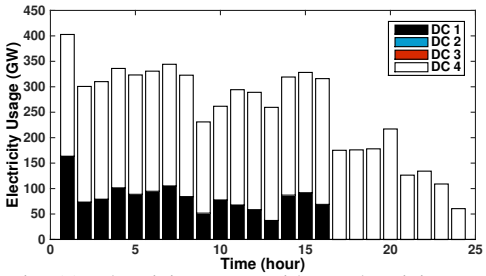


Fig. 11: Electricity usage with no electricity costs.

It remains to show that this periodic solution solves (6–10) if the times t are interpreted modulo T as in the proposition statement. It is easy to see that if $d_i \leq T$, the modulo T interpretation is equivalent to restricting the $x_{ik}(t)$ so that repeating this solution for $t > T$ is feasible. But this is simply the feasibility condition for maximizing the long-term average of (6). Since this long-term average objective is equal to (6) with a periodic solution, the proposition follows.

D. Proposition 4

Conceptually, the only difference between the static optimization problem (6–10) and the online one is that in the online optimization, we allow jobs that arrive at times $s_i \leq T$ to be scheduled after time T. Thus, we obtain (14) by adding $\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{t=T+1}^{2T}(\gamma_i(t) - c_k(t)e_i - p_{ik}b_i)\,x_{ik}(t)$ to (6).

We now consider the modifications to the constraints (7–9). First, we must ensure that at times $t = T+1,\ldots,2T$, each datacenter's bandwidth and memory capacity is not exceeded; thus, we obtain the constraints $\sum_{i=1}^{N} e_i x_{ik}(t) \leq \overline{P}_k(t)$ and $\sum_{i=1}^{N} b_i x_{ik}(t) \leq \overline{B}_k(t)$ for $t = T+1,\ldots,2T$. Combined with the capacity constraints (7), these modifications yield the two constraints (15) and (16) respectively. The constraint that each job be entirely scheduled between its arrival and deadline (8) is unchanged. Finally, the indivisibility constraint (9) is modified to (17) by summing from $t = 1$ to $t = 2T$ instead of $t = T$, reflecting the fact that some jobs can now be (partially) scheduled after time T.

APPENDIX B
ALGORITHM COMPARISONS

In this appendix, we compare our algorithm's results in Section V-B to those obtained with three other algorithms. We first consider variants of our algorithm when only electricity or only bandwidth cost is optimized, and then consider a heuristic that aims to optimize both the job completion times and resource costs. All three comparison algorithms yield a much higher cost. Throughout this section, for clarity of exposition we refer to our algorithm as the TCO (Two-Cost Optimizer).

A. Minimizing Individual Resource Costs

We first optimize only electricity costs, which yields the electricity usage in Figure 10. DC 3 is used the most due to its low electricity costs, as with the TCO (Figure 4a). DCs 1 and 2 are used slightly more than with the TCO, due to their high bandwidth costs (Table I).

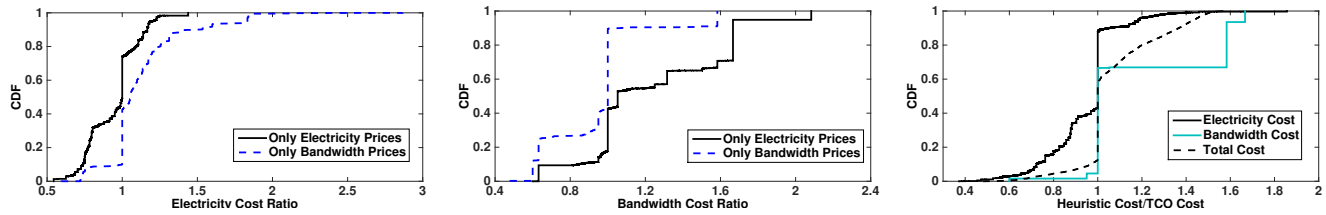
Figure 11 shows the electricity usage when only bandwidth costs are optimized. DC 4, which has the lowest bandwidth cost, is used the most, while DCs 2 and 3, which have the highest bandwidth costs, are not used at all. Usage is spread fairly evenly over time, since unlike electricity cost, bandwidth costs do not vary over time. Similar results are obtained for bandwidth usage.

Figure 12a compares the electricity cost per job when only electricity, only bandwidth, or both costs are optimized. Each CDF represents the ratio of electricity cost when only one resource is optimized to that with the TCO. As expected, optimizing electricity costs yields smaller ratios than optimizing bandwidth costs. However, only 53% of jobs reduce and 26% of jobs increase their electricity cost relative to the TCO when only electricity costs are optimized, while 10.3% of jobs lower their electricity cost when only bandwidth cost is optimized. Thus, the TCO extracts most of the potential electricity savings: *despite scheduling jobs to different timeslots and datacenters, optimizing only electricity saves very little (1.67%) on electricity cost, but optimizing only bandwidth increases electricity costs by 36.54%.*

We next show that the TCO can extract much of the potential saving in bandwidth costs. Figure 12b shows the bandwidth cost ratio when only one resource cost is optimized relative to that with the TCO. As with electricity costs, the CDF when only bandwidth is optimized is consistently to the left of that when only electricity is optimized. A surprisingly small 43.5% of jobs improve their bandwidth costs when only bandwidth is optimized, but only 10.5% of jobs increase their cost relative to the TCO. When only electricity is optimized, 17.8% of jobs reduce and 57.3% of jobs increase their bandwidth costs. As with electricity cost, *optimizing only electricity cost significantly increases the bandwidth cost by 37.5% relative to the TCO, while optimizing bandwidth reduces the bandwidth costs by 9.93%.* Overall, optimizing only one resource cost increases the total cost by 14 to 17% relative to the TCO.

B. Optimizing Job Completion Times

Finally, we compare the TCO to a heuristic aiming to minimize both job completion times and electricity and bandwidth costs. In this heuristic, all arriving jobs are initially scheduled to the cheapest datacenter in the current timeslot, which helps to minimize resource costs. If these jobs' required resources exceed any datacenter's capacity, some jobs are delayed to the next timeslot on the same datacenter. We begin by delaying jobs with the latest completion deadline, and continue delaying jobs in reverse order of their deadlines until each datacenter has sufficient capacity for its scheduled jobs. In the next timeslot, the delayed jobs are joined by newly arrived jobs,



(a) Electricity costs per job. (b) Bandwidth costs per job. (c) Per-job costs with adjusted job arrivals.

Fig. 12: Cumulative distribution functions for the ratios of (a) electricity costs per job with each resource cost optimized, (b) bandwidth costs per job with each resource cost optimized, and (c) electricity, bandwidth, and total costs per job with each resource cost optimized and the heuristic, relative to the same costs with the TCO.

and jobs from both sets are delayed in reverse order of their deadlines until the capacity constraints are met. Since we do not delay jobs to reduce costs, we expect that this heuristic will yield higher costs than the TCO.

We find that the cost improvement depends on job arrival patterns. If many jobs arrive when electricity prices are low, e.g., the 19.7% of jobs arriving in the first hour of the baseline scenario, the heuristic performs quite well, with only a 0.63% increase in cost over the TCO. However, when half of these jobs arrive in hour 18, which has higher electricity prices, the heuristic solution's cost increases by 8.46%. In contrast, the TCO shows only a 0.01% increase in cost with the new job arrival pattern. Figure 12c shows the ratio of each job's heuristic and TCO costs with the new job arrival pattern. Bandwidth costs in particular increase, as do the overall costs for 42.51% of jobs. If we compare these costs with the new job arrival pattern to those with full time flexibility (no deadlines), the TCO yields a 7.39% higher cost, and the heuristic a 16.43% higher cost. Thus, *even though resource costs are considered in the heuristic, fully leveraging time flexibility is crucial to the TCO's ability to reduce a datacenter operator's costs.*

Ioannis Kamitsos received his M.A. and Ph.D. from Princeton University, NJ, USA in 2009 and 2012 respectively and his Diploma in Electrical and Computer Engineering from the National Technical University of Athens, Greece in 2006. In the summer of 2010 he interned at the Standards and RF Lab, Samsung Telecommunications America, Richardson, TX. His research interests include optimization theory, optimal control and their applications on energy efficient computing, communications, networks and Smart Grid.

Carlee Joe-Wong (S'11) is a Ph.D. candidate at Princeton University's Program in Applied and Computational Mathematics. She received her A.B. in mathematics in 2011 and her M.A. in applied mathematics in 2013, both from Princeton University. In 2013, she was the Director of Advanced Research at DataMi, a startup she co-founded in 2012 that commercializes new ways of charging for mobile data. She received the INFORMS ISS Design Science Award in 2014 and the Best Paper Award at IEEE INFOCOM 2012. In 2011, she received the National Defense Science and Engineering Graduate Fellowship (NDSEG).

Sangtae Ha (S'07, M'09, SM'12) is an Assistant Professor in the Department of Computer Science at the University of Colorado at Boulder. He received his Ph.D. in Computer Science from North Carolina State University. His research focuses on building and deploying practical systems. He is a co-founder and the founding CTO/VP Engineering of DataMi, a startup company on mobile networks, and is a technical consultant to a few startups. He is an IEEE Senior Member and serves as an Associate Editor for IEEE Internet of Things (IoT) Journal. He received the INFORMS ISS Design Science Award in 2014.