Coordinated Energy-Aware Job Scheduling and Flexibility Provisioning in Distributed Data Centers

Aditya Shankar Kar
Department of Electrical and Computer Engineering
Iowa State University, IA 50010, USA
adityask@iastate.edu

Hongbo Sun, Arvind Raghunathan, Jianlin Guo
Mitsubishi Electric Research Labs
Cambridge, MA 02139, USA
{hongbosun,raghunathan,guo}@merl.com

***Abstract*—To reduce the data center's (DC) operational costs and provide grid services, large computational jobs need to be scheduled or transferred to the intended DCs with lower energy prices. Usually, the transfer or scheduling of large computation jobs are managed by job scheduling software, such as a portable batch system. The paper proposes a novel rolling window mixed integer linear programming-based (MILP) approach that can be leveraged by the job scheduling system to determine a job's location and start time to minimize cost and provide grid services. To address higher number of jobs in a stipulated time frame, the constraints arising from the job's resource requirements, execution time, location of execution, and their dependencies on the previous state are formulated linearly. The optimization module considers the grid service request and newly arrived jobs along with jobs in the queue, which ensures the feasibility and helps the operator with dynamic decision-making of pausing or transferring jobs. The effectiveness of the proposed approach is shown on the subset of data from the ALIBABA GPU cluster.**

***Index Terms*—Computational job scheduling, Energy management in data centers, grid support by data center, MILP**

I. Introduction

Variable generation from low-cost renewable resources and growing footprint of power demand from the data center loads and disproportionate growth in transmission capacity, necessitate scheduling large computational jobs at a suitable location and time of the day other than it's origin. Moreover, scheduling or deferring the job's start at the intended location enables the scheduler to provide grid services. Scheduling jobs while minimizing the cost of computation with time-varying energy prices is complex. To address the multifaceted challenges, there are multiple attempts in the literature with a focus on different aspects, such as meeting the service license agreement (SLA) between host and user, reduction of energy cost, providing grid service etc.

To address the aforementioned challenges, an algorithm for computational job routing and scheduling for multiple datacenter is proposed [1], which minimizes electricity and bandwidth cost. The dispatchable jobs are assumed to be continuously divisible, but in reality, they can only be split into predefined sub-tasks. Hence, the intricacies of holding an executing job, transferring a job into to separate data center(DC) are not addressed. Considering renewable co-located with the DCs, an MILP formulation is proposed in [2] to reduce the cost of computation. In [2]–[4], the resource and time constraints of the job are presented in an aggregated way without modeling details of individual job scheduling, such as job types and resource requirements, deadline, and execution time in the optimization formulation.

Considering deferrable computation requests, an approach for optimal power scheduling of a DC is given in [5]. As a single data center is considered, the price-sensitive job scheduling is limited to maximum processing delay without the option of transferring the job [3], [5]. A MILP-based model is developed in [6] considering the dispatchable and transferable jobs over a time horizon without any consideration of the existing jobs in the queue. As the workload allocation problem is NP-hard, a game theory-based workload management heuristic is presented in [7] where every job tries to minimize the cost of operation, thereby minimizing over all cost of the computation. But in practice, the jobs are submitted by the user without consideration of the processing capability of DCs. In [8], [9], the job arrival and service rate with the uncertainty are considered for a queuing model-based energy optimization without any clear strategy for the incoming jobs scheduling. Though methods presented in [8], [9] seem to be robust, their application in multi-data center remains a challenge.

From a computational point of view, A MILP formulation is proposed in [10] to minimize the makespan of the scheduled parallel task in the batch job, reducing the cost of operation. Using this formulation along with application signature to estimate the energy consumption, a framework is proposed for energy-aware task scheduling in DCs [11]. In [10]–[12], an integer programming is proposed to reduce the energy consumption by assigning jobs to the minimum number of servers to reduce the static power consumption without any provision to hold or transfer the task to another DC. In [13], [14], a scheduling algorithm and solution approach are formulated to reduce the execution time considering multiple DCs and network constraints. Jianpeng *et al.* propose a two-stage algorithm for job scheduling and resource allocation considering the time constraint of jobs in [15]. A rolling-horizon scheduling architecture for real-time task scheduling in virtualized clouds is presented in [16], which essentially adds Virtual Machines(VM) to finish the task within a time frame and stops or migrates VM to consolidate resources and save energy. The scheduling algorithm presented in [10]–[16] focuses on reducing the makespan of the job, thereby reducing the cost considering constraints related to computational job,

979-8-3315-8138-1

data transfer delay and cost, and resources of the server. However, the energy consumption related to servers and DCs, the cost of electricity at different locations, effective utilization of resources available by deferring a job are not considered. From a practical implementation perspective, the articulated aspects are equally important, including the energy consumption calculation. Server power consumption is modeled as linearly dependent on CPU utilization . The resulting cooling load at the CRAH is derived from total server power and propagated upstream to the chiller and cooling tower [2], [5], [6], [16], [17].

Hence, a MILP-based job scheduling approach is proposed here, which explicitly considers the grid service request and allows the transfer or delay in job execution in multi-DC environment. The novelty of the proposed approach are:

- As the framework tracks the execution of the job and considers the current state of the controllable jobs in the queue after a step of execution, it can be used together with the job scheduling software.
- The optimization module allows the scheduler to consider the grid service request, state of charge of the co-located energy storage system (ESS), and power constraints.
- The non-linearity arising from control action, such as transfer of a job or holding(or pausing) execution of a job and its previous state, is avoided by reformulating the constraint into a set of linear inequalities.

II. Framework for job scheduling

A. Definitions and Input to the framework

The MILP scheduling algorithm considers a list of computational jobs, energy price, state of charge of the energy storage device, a list of data center(DC: $k \in K$) with their available computational resources, power constraint and service requests posed by the grid operator as input. For the scheduling algorithm, a typical job $j \in J$ can be described as following

- Job resource utilization (R_j^{job}).
- Arrival time: Ta_j, Execution time: Te_j, Deadline: Td_j.
- Reward from computational (C_j^{reward}), Penalty from transfer & pause $(C_j^{transfer}, C_j^{pause})$.

Different job types are considered for defining the scope decision-making such as if a job is transferable. A list of job considered in the formulation is presented below.

- Deferrable job deadline Td_j greater than execution Te_j.
- Non-deferrable job deadline Td_j is equal to Te_j.
- Transferable job can be executed at a different location.
- Non-transferable job needs to be executed at the origin.

As the cost of energy varies over time and with the location of the DC, the cost of energy is defined for each location & time and provided as an input to the framework. In a typical setup, several CPU and GPU cores with certain RAM and storage capacities is presented as a unit, called a node. Multiple nodes are mounted in servers (e.g. Blade server). The servers are arranged in racks, and racks together represent the DC. For simple representation, we have considered the node-level resources per job. Here, the DCs are described as a set of

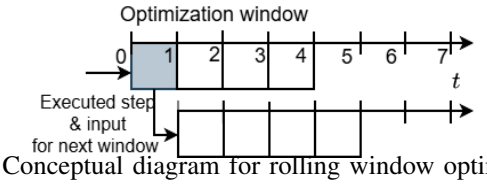

Fig. 1: Conceptual diagram for rolling window optimization

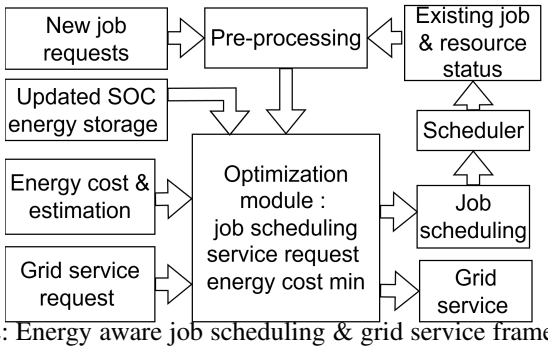

Fig. 2: Energy aware job scheduling & grid service framework

servers ($s \in S$) with a set of Nodes. The battery is co-located with the DC. The size and charging, and discharging capacities are provided as the input to the optimization formulation. The optimization module allows the following actions for the jobs.

- A job can be scheduled to run at any server at any given time slot within it's deadline.
- Execution of a job can be paused or held for a duration.
- A paused/held job can be released for run.
- A job can be transferred to any DC for execution.

During the entire execution timeline of the job, the framework keeps track of the job's status, remaining execution time, location of execution, etc.

B. Framework for Scheduling

The proposed scheduling algorithm is scheduled to run periodically, and it is a rolling window optimization over a time horizon from time $t = 0$ to T. The first step of the scheduling result is executed or implemented. The next optimization window considers the time horizon from time $t = 1$ to $T + 1$. It is illustrated with an example in the Fig.1. The previously executed time step is given as an input to the next optimization window to provide the state of currently running jobs and the location of the job execution.

The overall framework is presented in the Fig.2. It has three major parts, A. the pre-processing, B. the MILP optimization module, C. the execution and feedback loop. In the pre-processing step, list of jobs is updated by appending new job requests to the current list and removing the completed job. This pre-processing also ensures the feasibility of the optimization problem by calculating the maximum computational resource requirement and energy consumption of the job list. The central part of the framework is the mixed integer optimization module, which is explained in the Section III. The optimization module considers the energy price at different locations, the preprocessed job list and the associated cost of transferring the data from one location to another, and

computational revenue generated from job executions, and provides a schedule for the execution of the job. The battery energy storage system (BESS) model is incorporated into the optimization module. The result of the optimization module is provided to the scheduler, which uses the result of the first time interval and makes a decision. The executed step is fed back to the pre-processing for the next time window of optimization.

Algorithm 1: Pre-processing

Input: List the newly arrived jobs & list the existing jobs and their status.
Output: Matrix: $x_{j,s,t-1}$, $v^{\text{tie}}_{j,k,t-1}$ & list of new jobs.
Step 1: Update the current jobs input matrix:
1. If $\sum_{s=1}^{\text{S}} x_{j,s,t} = 1$ then $Te_j = Te_j - 1$ Else return Te_j
2. If $\sum_{s=1}^{\text{S}} x_{j,s,t} = 0$ then $Tl_j = Tl_j + 1$ Else return Tl_j
3. For each j, update the $x_{j,s,t-1}$
4. For each j, update the $v^{\text{tie}}_{j,k,t-1}$

Step 2: If $Te_j = 0$ remove the current job from the Job list.
Step 3: Initialize $x_{j,s,t-1}$, Tl_j & $v^{\text{tie}}_{j,k,t-1}$ to 0.
Step 4: Create an updated list by appending new jobs.
Step 5: Updated input matrix of $x_{j,s,t-1}$ & $v^{\text{tie}}_{j,k,t-1}$.
Step 6: Update the vectors: Td_j, Te_j, Tl_j

III. MILP FORMULATION

The following section explains the MILP formulation. The constraints are grouped for presentation, and together, they ensure the job execution without violating the service license agreement(SLA), the power consumption & the energy storage facility co-located with the data center(DC). The objective of the optimization module is to reduce the cost of computation (energy cost $C^{ele}_{k,t}$, job transfer C^{trans}_{j}, pause cost C^{pau}_{j}) and maximize the revenue generated by computational jobs (C^{reward}_{j}) and by providing grid services($C^{grid}_{k,t}$).

$$\sum_{j=1}^{\text{J}}\sum_{s=1}^{\text{S}}\sum_{t=1}^{Td_j} C^{reward}_{j}\cdot x_{j,s,t} + \sum_{k=1}^{\text{DC}}\sum_{t=1}^{\text{T}} C^{grid}_{k,t}\cdot p^{service}_{k,t} - \sum_{k=1}^{\text{DC}}\sum_{t=1}^{\text{T}} C^{ele}_{k,t}\cdot p^{tot}_{k,t} - \sum_{j=1}^{\text{J}}\sum_{t=1}^{Tr_j}\left(C^{pau}_{j}\cdot v^{pau}_{j,t} + C^{trans}_{j}\cdot v^{trans}_{j,t}\right) \tag{1}$$

A. Constraint pertaining to Job and SLA

The SLA ensures that each job (j) is executed within the deadline Td_j. For every job with arrival time Ta_j and expected execution time Te_j, the scheduling must satisfy $Ta_j + Te_j \leq Td_j$, and it can be delayed without any consequence by $Td_j - Te_j - Ta_j$. Tl_j is introduced to track the delay in the execution. Hence, the job 'j' must be executed ($\sum \Delta t \cdot x_{j,s,t}$) for at least its required processing time within the optimization time window($Tr_j \cdot \Delta t$) to meet its deadline. This also accounts for cases where the deadline lies beyond the optimization horizon. In essence, the total time a job is assigned to any server (i.e., actively running) must be no less than its required execution time. The second constraint in (2) ensures that the total execution time of the job over the entire optimization horizon will be equal to required execution time.

$$\begin{aligned}\sum_{s=1}^{\text{S}}\sum_{t=1}^{Tr_j}\Delta t\cdot x_{j,s,t} &\geq Tr_j\cdot\Delta t - Td_j + Te_j + Tl_j \\ \sum_{s=1}^{\text{S}}\sum_{t=1}^{Tr_j}\Delta t\cdot x_{j,s,t} &\leq Te_j\end{aligned} \quad \forall j \tag{2}$$

This constraint (3) ensures that at any time step t, each job j can be assigned to at most one server. The resource requirements ('R^{job}_j') of all jobs 'j' assigned to server 's' at time 't' (i.e., where '$x_{j,s,t} = 1$') must be less than or equal to the server's available capacity, which is '$N_s - R^{\text{exp}}_{s,t}$' (expected reserved resources for future).

$$\sum_{s=1}^{\text{S}} x_{j,s,t} \leq 1 \quad \forall j, \forall t \tag{3}$$

$$\sum_{j=1}^{\text{J}} R^{\text{job}}_j \cdot x_{j,s,t} \leq N_s - R^{\text{exp}}_{s,t} \quad \forall s, \forall t \tag{4}$$

TABLE I: Truth table for job-pause operations

$\sum x_{j,s,t-1}$	$\sum x_{j,s,t}$	$v^{pause}_{j,t}$
0	0	0
0	1	0
1	0	1
1	1	0

$$\begin{aligned} 0 \leq v^{pau}_{j,t} \leq \sum_{s=1}^{\text{S}} x_{j,s,t-1}, \qquad v^{pau}_{j,t} &\leq 1 - \sum_{s=1}^{\text{S}} x_{j,s,t} \\ v^{pau}_{j,t} &\geq \sum_{s=1}^{\text{S}} x_{j,s,t-1} - \sum_{s=1}^{\text{S}} x_{j,s,t} \end{aligned} \tag{5}$$

$$\begin{aligned} 0 \leq v^{trans}_{j,t} \leq \sum_{s=1}^{\text{S}} x_{j,s,t-1} + \sum_{k=1}^{\text{DC}} v^{\text{tie}}_{j,k,t-1}, \\ v^{trans}_{j,t} \leq \sum_{s=1}^{\text{S}} x_{j,s,t}, \\ v^{trans}_{j,t} \geq \sum_{s\in[k]} x_{j,s,t} - \sum_{s\in[k]} x_{j,s,t-1} - (1 - \sum_{s=1}^{S} x_{j,s,t-1}) \\ v^{trans}_{j,t} \geq \sum_{s\in[k]} x_{j,s,t} + \sum_{dc\neq k} v^{\text{tie}}_{j,dc,t-1} - 1, \end{aligned} \tag{6}$$

The optimization module allows the scheduler to pause or hold a job to allow high-priority job execution or facilitate grid services. This action can be mathematically described as $\sum_s x_{j,s,t} = 0$ with $\sum_s x_{j,s,t-1} = 1$. However, pausing job execution, delays job completion and increases cost as the job needs to be saved to secondary memory and then retrieved. Hence, the action is negatively penalized. The linearized constraints in (5) model the logic presented in Table-I to capture the jobs that are decided to be paused. The constraints in (6) are designed to detect and control when a job is transferred between DC from one time step to the next using the variable $v^{\text{tie}}_{j,dc,t-1}$. $v^{\text{tie}}_{j,dc,t-1}$ stores the DC previously assigned to job. This is important for tracking migration penalties.

B. Modeling the power requirement

In the comprehensive modeling, the energy usage of DC is usually categorized into two parts: server with storage & cooling requirement [18]. In this section, the total DC power consumption is presented with the required details connecting the computational resource to the corresponding cooling requirement, and the constraints arising from it. The power consumption of server (computation power) can be split into dynamic and static where the static power is consumed by the server when it is on. If at least one job is assigned to a server at a time, than server must be ON and in (7), the variable '$v_{s,t}^{s\_on}$' reflects whether any job is assigned to server.

$$x_{j,s,t} \leq v_{s,t}^{s\_on} \leq \sum_{j=1}^{\mathrm{J}} x_{j,s,t}, \quad \forall s, \forall t, \forall j \tag{7}$$

The dynamic power consumption of a server scales linearly with utilization [18]and is computed based on the fraction of resources used by assigned jobs. The constraints in (8) ensure that total server power at each time step accounts for both dynamic and static components.

$$p_{s,t}^{serve} = P_s^{max} \cdot \sum_{j=1}^{\mathrm{J}} \left(\frac{R_j^{\mathrm{job}}}{N_s} \cdot x_{j,s,t} \right) + P_s^{static} \cdot v_{s,t}^{s\_on} \quad \forall s, \forall t \tag{8}$$

The cooling requirement (CRAH, chiller) for each DC $k \in K$ at each time step is proportional to the total heat generated by the servers [18]. Hence, the power consumed by the cooling system is represented with a linear function of $p_{s,t}^{serve}$ in (9) with factor of η_k.

$$p_{k,t}^{crah} = \sum_{s \in DC_{list}[k]} \eta_k \cdot p_{s,t}^{serve} \quad \forall k, \forall t \tag{9}$$

The total power consumption ($p_{k,t}^{total}$) of each DC at each time step is calculated as the sum of power consumption by the servers and the cooling system. The power balance equation is presented in (10) to supply the DC power requirement by the ESS and the grid. The upper and lower limit of net power drawn from the grid $p_{k,t}^{grid^{max}}, p_{k,t}^{grid^{min}}$ respectively, is used to constrain the power consumption and provide grid services.

$$\begin{aligned} p_{k,t}^{total} &= \sum_{s \in DC_{list}[k]} p_{s,t}^{serve} + p_{k,t}^{crah} \quad \forall k, \forall t \\ p_{k,t}^{grid} \pm p_{k,t}^{service} &= p_{k,t}^{total} - p_{k,t}^{discharge} + p_{k,t}^{charge} \\ p_{k,t}^{grid^{min}} &\leq p_{k,t}^{grid} \pm p_{k,t}^{service} \leq p_{k,t}^{grid^{max}} \end{aligned} \tag{10}$$

C. Energy storage model

A simplified energy storage (battery) model is presented in (11) [19], with the discretized state of charge (SoC) representation. The SOC is updated at each time step, considering the charging and discharging behavior of batteries in the corresponding period. The given energy storage model is a mixed integer linear model with the decision variable $v_{dc,t}^{cd\_decision}$ to avoid the complementarity condition $p_{dc,t}^{\mathrm{charge}} p_{dc,t}^{\mathrm{discharge}} = 0$. Also, the maximum or minimum SOC capacity and Charging and discharging rates of the battery are modeled in the constraint presented in (11).

$$\begin{aligned} E_{k,0} &= E_k^{initial} \\ E_{k,t} = E_{k,t-1} + \eta_k^{charge} \cdot p_{dc,t}^{\mathrm{charge}} \Delta t - \frac{1}{\eta_k^{discharge}} \cdot p_{dc,t}^{\mathrm{discharge}} \Delta t \\ E_k^{final} &\leq E_{k,t} \leq E_k^{max} \\ p_{dc,t}^{\mathrm{charge}} &\leq P_{dc,t}^{\mathrm{charge\_max}} \cdot v_{dc,t}^{cd\_decision} \\ p_{dc,t}^{\mathrm{discharge}} &\leq P_{dc,t}^{\mathrm{discharge\_max}} (1 - v_{dc,t}^{cd\_decision}) \end{aligned} \tag{11}$$

IV. Result

For the evaluation of the framework and algorithm presented in Section II, we have considered the subset of the dataset given in [20], which contains a hybrid of training and inference jobs running ML algorithms across over 6,500 GPUs on 1800 machines. Apart from server & data center(DC) configuration, electricity prices, cost of data transfer, energy storage configuration, etc, are motivated from [17] normalized cost structure. In this section, the result of only one window of optimization is provided due to space limitations, and the effects of different controls are articulated.

TABLE II: Normalized electricity price at different DCs

Data Center	T0	T1	T2	T3
DC-0	0.01	0.01	0.1	0.1
DC-1	0.01	0.01	0.1	0.1
DC-2	0.1	0.1	0.01	0.01

As shown in Table II, three different DC locations with an optimization window of 4 time slots are considered with 9 jobs in queue for scheduling at time "t = 00:00 hr". The formulation includes 575 variables to track executing jobs while allowing dynamic job transfer and halt operations. With the given energy price in Table II, the simulation results are presented in the Fig.3 with two cases: high (in 3(a-d)) & low (in 3(e-h)) data transfer cost. With lower energy cost at "DC-0 & DC-1" for time slot "T0 & T1" compared to "DC-2", the jobs are scheduled at "DC-0 & DC-1" as shown in Fig.3(b & f). Comparing Fig.3(b & f) during "T2", the jobs are shifted or scheduled to "DC-2" owing to higher energy prices at "DC-0 & DC-1". With higher data transfer cost in Fig.3(b) compared Fig.3(f), instead of transferring the data directly from "DC-0 & DC-1" to "DC-2", jobs are directly scheduled at " DC-2 or DC-1 & DC-0". During the period of higher and lower energy price, the energy storage at all the DCs are observed to discharge & charge, respectively in Fig.3(c & g) to minimize energy cost. Moreover, as the server consumes static power, the number of servers deployed to run the jobs is also reduced Fig.3(a & e). Also, the total power consumption of DCs and net power drawn from grid are plotted in Fig.3(d & h). It can be observed that the power consumption is minimum during higher energy prices, and energy storage is used to reduce the net power drawn from the Grid. Comparing Fig.3(d & h), cost of data transfer is apparent, and to avoid higher transfer cost, the jobs are scheduled at a higher energy price periods. As shown in [17], [20], the arrival of large computational jobs varies by day of the week and hour of the day, ranging from

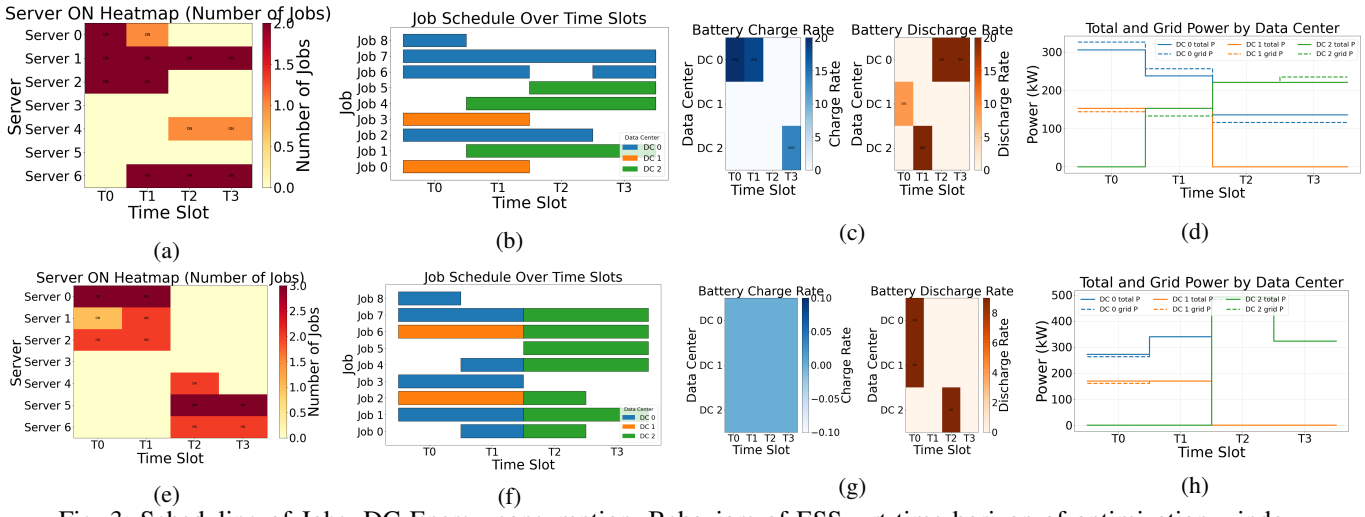

Fig. 3: Scheduling of Jobs, DC Energy consumption, Behaviors of ESS wrt time horizon of optimization window

20 to 70 per hour for a typical data center. The proposed approach is implemented and solved using commercial off-the-shelf IBM CPLEX optimization Solver, which could solve the formulation in total (root+branch&cut) 1.18 seconds for scheduling 9 jobs with 3 data centers at an instance. With the proposed rolling window optimization, the approach can be easily scaled to real data center job scheduling process.

V. Conclusion

A novel MILP-based approach is proposed for scheduling the large computational jobs, considering geographically distributed data center. Instead of aggregating the computational jobs and minimizing the cost of energy, the presented method considers individual jobs and their SLA for scheduling. With increscent implementation, i.e. pre-processing, optimization & feedback, allows the scheduler to integrate the proposed approach to the existing scheduling algorithm. The minimization of energy cost, together with explicit control to provide grid services, helps the scheduler to achieve a range of objectives. Currently, the developed algorithm is further expanded to consider actual integration into the existing scheduling process.

References

[1] C. Joe-Wong, I. Kamitsos, and S. Ha, "Interdatacenter job routing and scheduling with variable costs and deadlines," *IEEE Transactions on Smart Grid*, vol. 6, no. 6, pp. 2669–2680, 2015.

[2] D. T. Anh Nguyen, J. Cheng, N. Trieu, and D. T. Nguyen, "Optimal workload allocation for distributed edge clouds with renewable energy and battery storage," in *2024 International Conference on Computing, Networking and Communications (ICNC)*, 2024, pp. 700–705.

[3] F. Cao, Y. Wang, F. Zhu, Y. Cao, and Z. Ding, "Ups node-based workload management for data centers considering flexible service requirements," *IEEE Transactions on Industry Applications*, vol. 55, no. 6, 2019.

[4] F. Acun, I. C. Paschalidis, and A. K. Coskun, " Conductor: A Collaboration Framework for Multi-Data-Center Demand Response ," in *2024 IEEE 15th International Green and Sustainable Computing Conference (IGSC)*. Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2024, pp. 22–28. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/IGSC64514.2024.00013

[5] B. Zou, H. Zhang, G. Chen, and Y. Song, "Optimal power scheduling of data centers with deferrable computation requests," in *2021 IEEE 5th Conference on Energy Internet and Energy System Integration (EI2)*.

[6] Z. Zhao, L. Fan, and Z. Han, "Optimal energy and it service emergency schedule for internet data center," in *2024 56th North American Power Symposium (NAPS)*, 2024, pp. 1–6.

[7] N. Hogade, S. Pasricha, and H. J. Siegel, "Energy and network aware workload management for geographically distributed data centers," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 2, 2022.

[8] M. Ghamkhari and H. Mohsenian-Rad, "Energy and performance management of green data centers: A profit maximization approach," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 1017–1025, 2013.

[9] Z. Ding, S. Chen, Y. Sun, S. Shi, T. Xiao, Y. Wang, and X. Wei, "Data center job scheduling and energy management under uncertain environments," *IEEE Transactions on Industry Applications*, 2025.

[10] A. Goldman and Y. Ngoko, "A milp approach to schedule parallel independent tasks," in *2008 International Symposium on Parallel and Distributed Computing*, 2008, pp. 115–122.

[11] J. Salinas-Hilburg, M. Zapater, J. Moya, and J. Ayala, "Energy-aware task scheduling in data centers using an application signature," *Computers Electrical Engineering*, vol. 97, 12 2021.

[12] N. Liu, Z. Dong, and R. Rojas-Cessa, "Task scheduling and server provisioning for energy-efficient cloud-computing data centers," in *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*, 2013, pp. 226–231.

[13] Z. Hu, B. Li, and J. Luo, "Time- and cost- efficient task scheduling across geo-distributed data centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 3, pp. 705–718, 2018.

[14] C. Jing and P. Dan, "Jhtd: An efficient joint scheduling framework based on hypergraph for task placement and data transfer across geographically distributed data centers," *IEEE Access*, vol. 10, 2022.

[15] J. Lin, D. Cui, Z. Peng, Q. Li, and J. He, "A two-stage framework for the multi-user multi-data center job scheduling and resource allocation," *IEEE Access*, vol. 8, pp. 197 863–197 874, 2020.

[16] X. Zhu, L. T. Yang, H. Chen, J. Wang, S. Yin, and X. Liu, "Real-time tasks oriented energy-aware scheduling in virtualized clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 168–180, 2014.

[17] A. Guillen and et.al, "SustainCluster: Benchmarking Dynamic Multi-Objective Geo-Distributed Workload Management for Sustainable Data Center Cluster," in *Advances in Neural Information Processing Systems 38, 2025 Datasets and Benchmarks Track*, 2025, under Review.

[18] G. Zhabelova, M. Vesterlund, S. Eschmann, Y. Berezovskaya, V. Vyatkin, and D. Flieller, "A comprehensive model of data center: From cpu to cooling tower," *IEEE Access*, vol. 6, pp. 61 254–61 266, 2018.

[19] N. Nazir and M. Almassalkhi, "Guaranteeing a physically realizable battery dispatch without charge-discharge complementarity constraints," *IEEE Transactions on Smart Grid*, vol. 14, no. 3, pp. 2473–2476, 2023.

[20] Q. Weng, W. Xiao, Y. Yu, W. Wang, C. Wang, J. He, Y. Li, L. Zhang, W. Lin, and Y. Ding, "MLaaS in the wild: Workload analysis and scheduling in large-scale heterogeneous GPU clusters," in *19th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 22)*, 2022.