Resilient and Cost-Efficient Job Scheduling for Heterogeneous Data Centers: A Soft-Constrained Approach

Hengwen Xu
Business School
Central South University
Changsha, China
hengwenxu@csu.edu.cn

Hao Yin*
The Research Center for Digital Production and Smart Logistics, Tsinghua University
Beijing, China
yinh24@mails.tsinghua.edu.cn

***Abstract*—Conventional rigid scheduling models often struggle to maintain feasibility under peak loads or resource scarcity, largely due to inflexible physical and grid constraints. To address this, we propose a resilient scheduling framework for heterogeneous data centers that recasts these hard boundaries within a soft-constrained Mixed-Integer Linear Programming (MILP) formulation. We tackle the computational intractability of this NP-hard problem by introducing two novel classes of valid cuts: startup validity and rapid cycle prevention, which tighten the linear relaxation. Experiments show that these cuts reduce solution times by up to 62% on large-scale instances compared to standard baselines. A sensitivity analysis further confirms the resilience of the model, ensuring operational continuity and graceful degradation even when power availability drops to 50% of peak demand.**

***Index Terms*—Heterogeneous Data Centers, Resilient Scheduling, Soft-Constrained Optimization, Mixed-Integer Linear Programming (MILP), Valid Cuts, Graceful Degradation.**

I. Introduction and Related Work

Balancing operational efficiency against the limitations of physical infrastructure remains a fundamental challenge in modern data centers. As data centers account for a growing share of global electricity demand [1], operators face mounting pressure to reduce operating costs. Yet approaches that treat infrastructure as a pure abstraction often fail to account for the mechanical wear and electrical degradation induced by dynamic operations [2].

Energy-efficient scheduling has been studied extensively. Earlier work on dynamic consolidation [3] and electricity price arbitrage [4], [5] has reported substantial cost savings. However, such idealized models frequently overlook the penalties associated with state transitions. Lin *et al.* [6] note that server cycling incurs considerable costs reflecting both energy lag and hardware wear, which can offset the savings achieved through dynamic power management.

Grid interaction adds further complexity. To preserve grid stability, utilities impose strict power ramping limits, which are typically formulated as rigid hard constraints in the demand response literature [7]–[9]. In production environments subject to traffic bursts, such rigid boundaries often make the scheduling problem mathematically infeasible, causing solver failures precisely when availability matters most.

More recently, the growth of AI and machine-learning workloads has further intensified operational stress on data centers, motivating a new generation of carbon- and grid-aware scheduling approaches that exploit both temporal and spatial flexibility. Wiesner *et al.* [10] show that temporal workload shifting can substantially reduce cloud carbon emissions, while Radovanović *et al.* [11] report a production-grade carbon-aware compute platform that reshapes hourly load to follow a lower-carbon supply. Acun *et al.* [12] generalize these ideas through a holistic carbon-aware data center design framework that combines workload shifting with investments in renewable energy and storage. Although these works show substantial potential for decarbonization, they typically keep hard feasibility envelopes around capacity, ramping, or carbon limits, and so inherit the same brittleness under acute scarcity.

A robust scheduler must prioritize feasibility through controlled degradation rather than outright system failure. While soft constraints are standard in real-time operating systems [13], their adoption in holistic data center scheduling remains limited [14], [15]. From this perspective, we treat *resilience* as the ability of the scheduler to retain a feasible dispatching decision when one or more physical envelopes (aggregate power capacity, ramping headroom, carbon budget, and hardware on/off durations) are disturbed or driven into scarcity, by trading off violations across SLA, energy, environmental, and load-balance dimensions in a controlled, penalty-driven manner rather than returning an infeasible status. To bridge this gap, we propose a resilient scheduling framework based on a penalty-driven optimization model. By reformulating constraints on SLAs, power capacity, ramping rates, and carbon emissions as soft constraints, the proposed model keeps the problem feasible even under extreme loads. To handle the computational complexity of this formulation, we also derive valid cuts that tighten the search space and improve solvability.

*Corresponding author: yinh24@mails.tsinghua.edu.cn

II. Formulation

A. Problem Description

To make the setting concrete, consider an operator that manages a heterogeneous cluster of roughly ten servers, comprising a mix of general-purpose CPU nodes and accelerator GPU nodes with markedly different idle/peak power and wear profiles, and that serves a queue of about forty batch training and inference jobs over a multi-hour planning horizon. At each scheduling tick (e.g., every fifteen minutes), the operations team must decide, for the upcoming horizon, (i) which job is dispatched to which server and at which slot, (ii) which servers are activated, shut down, or kept idle, and (iii) how the aggregate power trajectory tracks time-varying electricity prices and grid-imposed ramping limits, while staying within a daily carbon emission envelope and respecting the SLA deadline of each job. When any of these envelopes tightens unexpectedly, for instance under a coincident peak-pricing window, a curtailment of available power, and a stricter ramping cap, a rigidly formulated model may become infeasible at precisely the moment when scheduling decisions matter most.

We therefore address the joint scheduling and resource management problem in heterogeneous data centers subject to strict physical and grid constraints. The core challenge is to assign batch jobs to servers with diverse power and wear characteristics while complying with dynamic ramping and carbon limits. To keep the problem solvable under peak loads, we relax the hard boundaries on capacity, deadlines, and load balancing into soft constraints. The objective is to minimize a generalized total cost that balances operational expenditures, including electricity and hardware wear, against weighted penalties for constraint violations, so that the system remains resilient.

B. Mathematical Model

Table I summarizes the sets, parameters, and decision variables used in the model. The Mixed-Integer Linear Programming (MILP) formulation of the resilient scheduling problem is given below:

TABLE I
Nomenclature and Parameter Definitions

Symbol	Definition
Sets and Indices	
S, k	Set of heterogeneous servers, index $k \in \{1, \dots, n\}$
J, j, i	Set of batch jobs, indices $j, i \in \{1, \dots, m\}$
T, t	Set of discrete time slots, index $t \in \{1, \dots, \lvert T\rvert\}$
Job & Workload Parameters	
$D_{j,k}$	Processing time of Job j on Server k
DL_j	Soft deadline for Job j
System & Physical Parameters	
P_k^{idle}	Power consumption of Server k (Idle/ON)
P_k^{peak}	Power consumption of Server k (Full Load)
e_t	Carbon emission intensity at time t
$C_{\text{DC}}^{\max}$	Maximum aggregate power capacity
$C_{\text{carb}}^{\max}$	Maximum allowable carbon emission cap
ΔP_{ramp}	Maximum allowable power ramping rate
$T_{\min}^{\text{on}}$	Minimum continuous ON time
$T_{\min}^{\text{off}}$	Minimum continuous OFF time
Δ^{bal}	Target threshold for load imbalance
Cost & Penalty Coefficients	
c_t^{elec}	Time-varying electricity price at time t
c^{wear}	Base hardware wear cost coefficient
c^{start}	Single server startup cost
β_k	Wear sensitivity weight for Server k
ρ^{SLA}	Penalty per unit of deadline violation
ρ^{pow}	Penalty per unit of power capacity violation
ρ^{ramp}	Penalty per unit of ramping rate violation
ρ^{carb}	Penalty per unit of carbon cap violation
ρ^{hw}	Penalty for min ON/OFF violation
ρ^{bal}	Penalty per unit of load imbalance violation
Decision Variables	
$st_{j,k,t}$	Binary: 1 if Job j starts on k at t
$x_{j,k,t}$	Binary: 1 if Job j executes on k at t
$u_{k,t}$	Binary: 1 if Server k is ON at t
$z_{k,t}$	Binary: 1 if Server k starts up at t
$P_{k,t}^{\text{total}}$	Total power of Server k at t
Slack Variables (Soft-Constraint Violations)	
v_j^{SLA}	Deadline (SLA) violation magnitude for Job j
v_t^{pow}	Power capacity violation at time t
v_t^{ramp}	Power ramping violation at time t
$v_{k,t}^{\text{on}}$	Minimum ON-time violation for Server k at t
$v_{k,t}^{\text{off}}$	Minimum OFF-time violation for Server k at t
v^{carb}	Carbon emission cap violation
$v_{k,k'}^{\text{bal}}$	Load imbalance violation between Servers k, k'

$$\begin{aligned}\min Z = &\sum_{t\in T}\sum_{k\in S}\left(c_t^{\text{elec}}P_{k,t}^{\text{total}} + \beta_k c^{\text{wear}}u_{k,t} + c^{\text{start}}z_{k,t}\right)\\ &+\sum_{j\in J}\rho^{\text{SLA}}v_j^{\text{SLA}} + \sum_{t\in T}\rho^{\text{pow}}v_t^{\text{pow}}\\ &+\sum_{t\in T}\rho^{\text{ramp}}v_t^{\text{ramp}} + \rho^{\text{carb}}v^{\text{carb}}\\ &+\sum_{t\in T}\sum_{k\in S}\rho^{\text{hw}}\left(v_{k,t}^{\text{on}} + v_{k,t}^{\text{off}}\right)\\ &+\rho^{\text{bal}}\sum_{k\in S}\sum_{\substack{k'\in S\\ k'>k}}v_{k,k'}^{\text{bal}} \qquad (1)\end{aligned}$$

$$\text{s.t.}\quad P_{k,t}^{\text{total}} = P_k^{\text{idle}}u_{k,t} + (P_k^{\text{peak}} - P_k^{\text{idle}})\sum_{j\in J}x_{j,k,t}, \quad \forall k,t \qquad (2)$$

$$\sum_{k\in S}\sum_{t=1}^{|T|}st_{j,k,t} = 1, \quad \forall j\in J \qquad (3)$$

$$x_{j,k,t} = \sum_{\tau=\max(1,\,t-D_{j,k}+1)}^{t}st_{j,k,\tau}, \quad \forall j,k,t \qquad (4)$$

$$\sum_{j\in J}x_{j,k,t} \le 1, \quad \forall k\in S, t\in T \qquad (5)$$

$$x_{j,k,t} \le u_{k,t}, \quad \forall j,k,t \qquad (6)$$

$$u_{k,t} - u_{k,t-1} \le z_{k,t}, \quad \forall k, t\ge 2 \qquad (7)$$

$$\sum_{k\in S}\sum_{t\in T}(t + D_{j,k} - 1)st_{j,k,t} \le \mathrm{DL}_j + v_j^{\text{SLA}}, \ \forall j \qquad (8)$$

$$\sum_{k\in S} P_{k,t}^{\text{total}} \le C_{\text{DC}}^{\max} + v_t^{\text{pow}}, \quad \forall t \in T \tag{9}$$

$$\sum_{k\in S}(P_{k,t}^{\text{total}} - P_{k,t-1}^{\text{total}}) \le \Delta P_{\text{ramp}} + v_t^{\text{ramp}},\ \forall t \ge 2 \tag{10}$$

$$\sum_{k\in S}(P_{k,t-1}^{\text{total}} - P_{k,t}^{\text{total}}) \le \Delta P_{\text{ramp}} + v_t^{\text{ramp}},\ \forall t \ge 2 \tag{11}$$

$$\sum_{\tau=t-T_{\min}^{\text{on}}+1}^{t} z_{k,\tau} \le u_{k,t} + v_{k,t}^{\text{on}}, \quad \forall k,t \tag{12}$$

$$\begin{aligned} &u_{k,t-1} - u_{k,t} \ \le\ 1 - u_{k,\tau} + v_{k,t}^{\text{off}}, \\ &\quad \forall k \in S,\ \forall t \in \{2,\dots,|T|\}, \\ &\quad \forall \tau \in \{t+1,\dots,\min(t+T_{\min}^{\text{off}},\ |T|)\} \end{aligned} \tag{13}$$

$$\sum_{t\in T}\sum_{k\in S} P_{k,t}^{\text{total}} \cdot e_t \le C_{\text{carb}}^{\max} + v^{\text{carb}} \tag{14}$$

$$\sum_{t\in T} u_{k,t} - \sum_{t\in T} u_{k',t} \le \Delta^{\text{bal}} + v_{k,k'}^{\text{bal}},\ \ \forall k,k' \in S \tag{15}$$

$$\sum_{t\in T} u_{k',t} - \sum_{t\in T} u_{k,t} \le \Delta^{\text{bal}} + v_{k,k'}^{\text{bal}},\ \ \forall k,k' \in S \tag{16}$$

$$st, x, u, z \in \{0,1\};\ \ v^{(\cdot)} \ge 0 \tag{17}$$

The objective function in (1) minimizes the generalized total cost, which combines electricity expenses, hardware wear, startup penalties, and weighted penalties for soft constraint violations. On the side of physical operations, Constraints (2)–(6) define the power consumption, ensure unique job assignments, and restrict execution to active servers, while Eq. (7) governs the startup logic.

The soft constraints are given in (8)–(16). Eq. (8) permits deadline violations subject to a penalty. To preserve grid stability and hardware longevity, Constraints (9)–(13) regulate the power capacity, ramping rates, and minimum ON/OFF durations. In particular, Eq. (13) enforces that whenever Server k is shut down at slot t (i.e., $u_{k,t-1} - u_{k,t} = 1$), it must remain off at every subsequent slot τ within the minimum OFF window $\{t{+}1,\dots,t{+}T_{\min}^{\text{off}}\}$, with $v_{k,t}^{\text{off}}$ absorbing any forced reactivation within that window. Finally, (14) and (15)–(16) cover the environmental and load balancing targets, allowing controlled deviations to keep the problem feasible.

III. Algorithm Design

A. Branch-and-Cut Algorithm Framework

We solve the MILP model formulated in Section II with a Branch-and-Cut (B&C) algorithm [16]. The problem is NP-hard: even without the soft-constraint extensions, it embeds the unrelated parallel-machine scheduling problem with deadlines, shown to be strongly NP-hard by Lenstra *et al.* [17], together with discrete server activation and minimum on/off decisions that generalize the unit commitment problem, which is also known to be NP-hard [18]. The integrality constraints on the binary variables (e.g., $x_{j,k,t}$, $z_{k,t}$) are relaxed to the continuous interval $[0,1]$ following standard principles of integer programming [19]. A *Best-Bound Search* strategy is used for node selection so that the most promising regions of the search tree are explored first.

Algorithm 1 Branch-and-Cut Algorithm Framework

Require: MILP Model parameters (Jobs J, Servers S, Time T)
Ensure: Optimal Schedule S^* and Objective Value Z^*

Initialize: Global Upper Bound $UB \leftarrow \infty$, Lower Bound $LB \leftarrow -\infty$
Create root node N_0 with LP relaxation; $\mathcal{L} \leftarrow \{N_0\}$
while $\mathcal{L} \neq \emptyset$ **do**
  Select node $N_k \in \mathcal{L}$ using *Best-Bound* strategy; $\mathcal{L} \leftarrow \mathcal{L} \setminus \{N_k\}$
  repeat
    Solve LP relaxation of N_k to obtain x_k^* and value Z_{LP}
    if LP is infeasible **or** $Z_{LP} \ge UB$ **then**
      Prune node N_k; **break**
    end if
    if x_k^* is integral **then**
      if $Z_{LP} < UB$ **then**
        $UB \leftarrow Z_{LP}$; $S^* \leftarrow x_k^*$
      end if
      Prune by optimality; **break**
    end if
    Separation Procedure: Check violations of Eqs. (18), (19)
    if Violated cuts found **then**
      Add cuts to N_k; **continue** (re-solve LP)
    else
      break (proceed to branching)
    end if
  until Node is pruned or ready for branching
  if N_k is not pruned **then**
    Select fractional variable v (priority to $x_{j,k,t}$)
    Generate children $N_k^0(v=0)$, $N_k^1(v=1)$; Add to $\mathcal{L}$
  end if
end while
return S^*, UB

Within this framework, a separation procedure dynamically generates valid cuts violated by the current fractional solution x^* to tighten the relaxation. When no violated cut is found, a *most-fractional branching strategy* is applied, choosing the variable with the highest integer infeasibility to partition the search space.

B. Valid Cuts

To accelerate convergence, the LP relaxation is strengthened by two classes of valid cuts that explicitly rule out physically or logically invalid state transitions frequently seen in fractional LP solutions:

$$z_{k,t} \leq u_{k,t}, \quad \forall k \in S, t \in T \tag{18}$$

$$z_{k,t} + z_{k,t+1} \leq 1, \quad \forall k \in S, t \in \{1, \ldots, |T| - 1\} \tag{19}$$

The startup validity cut in Eq. 18 enforces logical consistency by linking startup costs to operational states. It ensures that a startup cost $z_{k,t}$ is not assigned to an inactive server for which $u_{k,t} = 0$. The rapid cycle prevention cut in Eq. 19 addresses numerical anomalies by ruling out fractional "micro-cycling": it forbids startups in two consecutive time slots, a condition that is often missed by standard relaxations. Algorithm 1 gives the complete procedure.

IV. Numerical Experiments

A. Experimental Setup and Instances

The experiments were run on a workstation with an AMD Ryzen 7 5800H CPU (3.20 GHz) and 32 GB of RAM. All algorithms were implemented in Python 3.10.5, and the MILP models were solved using Gurobi Optimizer 11.0.1 with a MIP gap tolerance of 0.01%. To test performance across different levels of complexity, a dataset of 15 instances was generated and grouped into small, medium, and large scales. The workload reflects realistic heterogeneity: servers have distinct idle power P_k^{idle} and peak power P_k^{peak} characteristics. Random noise was also added to the job processing times $D_{j,k}$ to simulate architectural affinity. Dynamic environmental factors, including the electricity prices c_t^{elec} and the carbon intensity e_t, follow realistic peak-valley fluctuations so that the temporal scheduling capability of the model can be evaluated.

TABLE 2
Scale Definitions of Test Instances

Scale	Servers ($\|S\|$)	Jobs ($\|J\|$)	Time Horizon ($\|T\|$)	Count
Small	3	15	10	5
Medium	5	24	12	5
Large	10	40	20	5

B. Parameter Settings and Comparative Schemes

The generalized total cost combines operational expenditures with penalties for soft constraint violations. The penalty coefficients ρ were calibrated to be substantially larger than the standard operational costs, so that the solver prioritizes physical feasibility and relaxes constraints only under extreme resource contention. To simulate resource scarcity, the power capacity $C_{\text{DC}}^{\max}$ was capped at 70% of the peak demand. Two solver configurations are benchmarked: the *Base* model, which is the standard MILP formulation, and the *Base+Cuts* model, which adds the valid cuts defined in Eq. 19 and Eq. 18 to tighten the linear relaxation. The solver time limits are set to 20, 100, and 600 seconds for the Small, Medium, and Large instances, respectively.

C. Experimental Results and Analysis

Table 3 reports the computational performance of both the standard formulation and the proposed cut-augmented model, with a focus on solution time and the number of explored Branch-and-Cut nodes. Across all fifteen instances, both configurations converged to the globally optimal solution with a MIP gap of 0.00%. This confirms that the added valid cuts strictly tighten the LP relaxation without pruning any feasible integer solutions. The primary metrics for evaluation are therefore the reduction in the search space (Nodes) and the resulting algorithmic efficiency (Time).

TABLE 3
Detailed Experimental Results of All Configurations on Test Instances

Scale	ID	Method	Nodes	Time (s)
Small	1	Base	27,433	3.3
		Base+Cuts	10,741	1.8
	2	Base	51,487	6.2
		Base+Cuts	19,640	3.2
	3	Base	8,424	2.3
		Base+Cuts	6,198	1.5
	4	Base	96,131	5.8
		Base+Cuts	32,191	2.1
	5	Base	7,541	1.3
		Base+Cuts	756	0.4
Medium	1	Base	49,362	30.5
		Base+Cuts	17,030	13.0
	2	Base	107,799	41.4
		Base+Cuts	26,346	16.3
	3	Base	8,587	11.8
		Base+Cuts	4,352	4.6
	4	Base	7,614	9.9
		Base+Cuts	6,757	5.5
	5	Base	17,757	16.0
		Base+Cuts	10,539	10.1
Large	1	Base	20,526	509.1
		Base+Cuts	17,921	313.1
	2	Base	11,345	482.2
		Base+Cuts	4,279	182.3
	3	Base	55,165	451.6
		Base+Cuts	83,898	387.8
	4	Base	111,872	403.2
		Base+Cuts	65,673	222.9
	5	Base	6,635	524.9
		Base+Cuts	8,599	515.2

Note: All instances reach the optimal solution (OPT).

The node counts help explain where the speedups come from. For small- and medium-scale instances, the Base+Cuts configuration substantially reduced the size of the search tree. In the medium-scale dataset, the average number of explored nodes dropped by roughly 48%. This reduction indicates that the valid cuts effectively pruned the fractional solutions that violated physical logic (e.g., rapid cycling or invalid startups), allowing the solver to discard suboptimal branches earlier in the search. The pruning translates directly into the observed runtime improvements: the Base+Cuts configuration achieved

an average speedup of 42.7% over the standard model on the medium instances.

In the large-scale experiments, which are the core test scenario for practical operations, the scalability benefits become more pronounced. While the standard Base model approached the 600-second time limit in several cases, the Base+Cuts configuration held up well. In Instance 2, the node count was reduced by more than 62% (from 11,345 to 4,279), and the solution time fell from 482.2s to 182.3s. Even in instances where the reduction in node count was smaller or slightly reversed due to heuristic variations (e.g., Instance 3), the solver still kept a lower runtime, suggesting that the cuts improved the quality of the lower bounds and guided the search toward the optimum more efficiently. Taken together, these results confirm that the proposed cuts work as intended: by ruling out illogical state transitions in the LP relaxation, the framework minimizes the generalized total cost while keeping operations feasible at significantly lower computational cost.

D. Sensitivity Analysis

To assess the scalability and robustness of the proposed Base+Cuts configuration, a sensitivity analysis was conducted by varying both the problem scale and the resource constraints. The experiments use a baseline setting of $|S| = 5$, $|J| = 24$, and $|T| = 12$ with a capacity ratio $\eta = 0.7$, and the results were averaged over three independent runs with a 300-second timeout. The results in Fig. 1 show that the four dimensions behave quite differently.

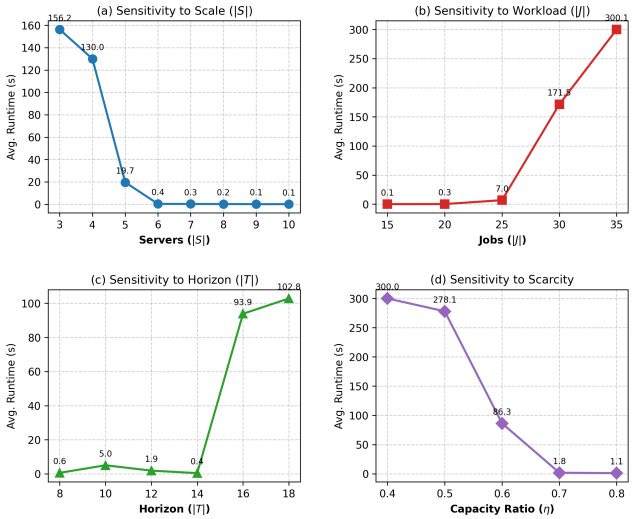

Fig. 1. Sensitivity analysis of solver runtime across four dimensions: (a) number of servers $|S|$, (b) number of jobs $|J|$, (c) time horizon $|T|$, and (d) capacity ratio η. The solver time limit is set to 300s.

Along the server dimension, resource availability and computational difficulty are clearly inversely related. As shown in Fig. 1(a), a restricted infrastructure ($|S| = 3$) forces the solver to search a highly constrained space, producing runtimes above 150 seconds. Increasing the server count, however, relaxes these feasibility boundaries, and the Branch-and-Cut algorithm finds low-cost schedules almost instantaneously (< 0.5s) when $|S| \geq 6$. The job dimension (Fig. 1(b)), by contrast, drives the main computational bottleneck. Runtime stays stable for workloads of up to 25 jobs, beyond which the combinatorial explosion of assignment variables causes a sharp spike, with the time limit consistently reached at $|J| = 35$. The time horizon (Fig. 1(c)) has a milder effect, with complexity growing in a manageable way until longer planning periods ($|T| \geq 16$) produce deeper search trees.

The capacity ratio analysis (Fig. 1(d)) highlights the resilience of the model under stress. In scenarios of extreme scarcity ($\eta \leq 0.5$), where traditional rigid models would likely return infeasible statuses, the proposed soft-constrained framework keeps optimizing the penalty trade-offs throughout the entire 300-second window. The associated high computational cost reflects the difficulty of minimizing violations within a saturated system. As resources return to normal levels ($\eta \geq 0.7$), the pressure on the soft constraints eases, and the solver returns to high efficiency. This confirms that the proposed approach achieves graceful degradation under critical load while keeping high speed during normal operations.

V. Conclusion

This study addressed the feasibility challenges of conventional rigid scheduling models by proposing a soft-constrained MILP framework for heterogeneous data centers. By reformulating strict physical and grid limitations into penalty-based objectives, the proposed model maintains operational continuity even under peak load conditions where traditional methods typically fail.

To handle the computational complexity of NP-hard scheduling problems, two classes of valid cuts were derived and integrated: the startup validity and the rapid cycle prevention constraints. Numerical experiments show that these cuts significantly tighten the LP relaxation, yielding computational speedups of up to 62% on large-scale instances while preserving global optimality.

The sensitivity analysis further highlights the resilience of the proposed approach. Although the computational cost inevitably grows with workload density, the system still achieves graceful degradation under extreme resource scarcity. Rather than returning infeasible statuses during power capacity shortages, the scheduler optimizes the trade-off between service levels and operational penalties. This capability suggests that the proposed framework offers a practical and scalable solution for modern data centers that face growing uncertainty in energy supply and workload volatility.

References

[1] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey, "Recalibrating global data center energy-use estimates," *Science*, vol. 367, no. 6481, pp. 984–986, 2020.

[2] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges," *arXiv preprint arXiv:1006.0308*, 2010.

[3] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.

[4] Y. Zhang, Y. Wang, and X. Wang, "Greenware: Greening cloud-scale data centers to maximize the use of renewable energy," in *ACM/IFIP/USENIX international conference on distributed systems platforms and open distributed processing*. Springer, 2011, pp. 143–164.

[5] 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 *2010 Proceedings IEEE INFOCOM*. IEEE, 2010, pp. 1–9.

[6] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Transactions on Networking*, vol. 21, no. 5, pp. 1378–1391, 2012.

[7] N. H. Tran, D. H. Tran, S. Ren, Z. Han, E.-N. Huh, and C. S. Hong, "How geo-distributed data centers do demand response: A game-theoretic approach," *IEEE Transactions on Smart Grid*, vol. 7, no. 2, pp. 937–947, 2015.

[8] H. Wang, J. Huang, X. Lin, and H. Mohsenian-Rad, "Proactive demand response for data centers: A win-win solution," *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1584–1596, 2015.

[9] Í. Goiri, W. Katsak, K. Le, T. D. Nguyen, and R. Bianchini, "Parasol and greenswitch: Managing datacenters powered by renewable energy," *ACM SIGPLAN Notices*, vol. 48, no. 4, pp. 51–64, 2013.

[10] P. Wiesner, I. Behnke, D. Scheinert, K. Gontarska, and L. Thamsen, "Let's wait awhile: How temporal workload shifting can reduce carbon emissions in the cloud," in *Proceedings of the 22nd International Middleware Conference*, 2021, pp. 260–272.

[11] A. Radovanović, R. Koningstein, I. Schneider, B. Chen, A. Duarte, B. Roy, D. Xiao, M. Haridasan, P. Hung, N. Care *et al.*, "Carbon-aware computing for datacenters," *IEEE Transactions on Power Systems*, vol. 38, no. 2, pp. 1270–1280, 2022.

[12] B. Acun, B. Lee, F. Kazhamiaka, K. Maeng, U. Gupta, M. Chakkaravarthy, D. Brooks, and C.-J. Wu, "Carbon explorer: A holistic framework for designing carbon aware datacenters," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 2023, pp. 118–132.

[13] G. Buttazzo, G. Lipari, L. Abeni, and M. Caccamo, *Soft real-time systems: predictability vs. efficiency*. Springer, 2005.

[14] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE transactions on parallel and distributed systems*, vol. 24, no. 6, pp. 1107–1117, 2012.

[15] Z. Zhou, F. Liu, Y. Xu, R. Zou, H. Xu, J. C. Lui, and H. Jin, "Carbon-aware load balancing for geo-distributed cloud services," in *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, 2013, pp. 232–241.

[16] M. Padberg and G. Rinaldi, "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems," *SIAM review*, vol. 33, no. 1, pp. 60–100, 1991.

[17] J. K. Lenstra, D. B. Shmoys, and É. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," *Mathematical programming*, vol. 46, no. 1, pp. 259–271, 1990.

[18] P. Bendotti, P. Fouilhoux, and C. Rottner, "On the complexity of the unit commitment problem," *Annals of Operations Research*, vol. 274, no. 1, pp. 119–130, 2019.

[19] L. A. Wolsey, *Integer programming*. John Wiley & Sons, 2020.