An Energy and Migration-Aware VM Placement Model and Heuristics for Green Cloud Data Centers

Okan Çandır, Ali Haydar Özer
Marmara University, Istanbul, Turkey
okan.candir@marun.edu.tr, haydar.ozer@marmara.edu.tr

***Abstract*—Cloud platforms consume a substantial share of global electricity, making schedulers that jointly consider performance and energy increasingly important. We present *EMM-Sched*, an energy-aware, multi-period, migration-aware framework that unifies admission control and multidimensional VM placement across discrete intervals. The model captures migration costs between periods, heterogeneous server efficiencies, and request flexibility via compact AND/OR structures. We instantiate the framework as (i) a mixed-integer program (MIP) that provides an optimal reference and (ii) two scalable heuristics: Energy-Minimizing First-Fit Decreasing (EMM-FFD) and Best-Fit Decreasing (EMM-BFD). On data-center–scale synthetic workloads, the exact solver yields the highest objective but incurs steep runtimes at scale, whereas the heuristics return solutions within 10–20% of optimal in minutes. EMM-FFD consistently lowers energy and migration costs; EMM-BFD achieves higher utilization at slightly higher energy. Overall, EMM-Sched exposes a tunable optimality–runtime trade-off suitable for both online operation and capacity planning.**

I. Introduction

The widespread adoption of cloud computing has led to the rapid expansion of large-scale data centers, which now account for a significant portion of global electricity usage. As energy costs rise and environmental concerns increase, improving the energy efficiency of cloud infrastructures has become a priority. Virtual machine (VM) placement, that is deciding where to run user workloads within a pool of physical machines (PMs), is a key determinant of both operational efficiency and energy usage.

Clouds host heterogeneous workloads whose demand and placement feasibility change over time. Meeting these demands efficiently requires coupling admission, placement, and migration across successive planning intervals, while accounting for server energy efficiency and the operational cost of migrating running VMs.

We propose *EMM-Sched*, an Energy-aware, Multi-period, Migration-aware scheduling framework that formulates this joint problem over a finite horizon. Each evaluation interval executes a solver that

(i) admits a subset of incoming VM requests,
(ii) places admitted VMs subject to multi-resource constraints, and
(iii) optionally migrates already running VMs when the expected benefit outweighs migration cost.

Request flexibility is expressed via compact AND/OR structures, enabling the scheduler to choose among alternative VM bundles or configurations.

Our contributions are:

- We give a mixed-integer programming (MIP) model that unifies admission, placement, and migration with energy cost terms and capacity/efficiency heterogeneity.
- We design two fast heuristics, EMM-FFD and EMM-BFD, that approximate the MIP while scaling to data-center sizes and delivering predictable runtimes suitable for periodic operation.
- We provide an empirical study across capacity and demand density using 2,532 test instances. The exact MIP yields the highest objective, but its runtime grows sharply with problem size; the heuristics remain stable and fast while providing solutions within $\approx$10–20% of the exact objective. EMM-FFD minimizes energy and migration cost; EMM-BFD offers higher utilization with higher migration cost; and the exact solver reaches the highest utilization at significant compute time.

II. Related Work

The problem of virtual machine (VM) placement in cloud data centers has been a topic of extensive research. Due to the computational complexity of the virtual machine (VM) placement problem, metaheuristics and nature-inspired algorithms have emerged as prominent solutions, offering scalability and adaptability in large-scale cloud environments.

Pourghebleh et al. [1] provided a comprehensive survey on the application of metaheuristic algorithms in VM consolidation, highlighting trends in swarm intelligence, evolutionary computation, and hybrid methods. Dashti et al. [2] introduced a particle swarm optimization (PSO)-based technique that dynamically adapts VM allocation to reduce energy consumption while maintaining system stability.

Tang et al. [3] proposed a hybrid genetic algorithm that integrates local and global search capabilities to optimize energy efficiency during VM allocation. Zheng et al. [4] explored biogeography-based optimization (BBO) for multi-objective VM consolidation, balancing energy, resource utilization, and performance degradation.

Abdel-Basset et al. [5] designed a Lévy-flight-enhanced Whale Optimization Algorithm (WOA) for bandwidth-aware VM placement, achieving improved convergence speed and reduced power consumption.

Importantly, Özer and Özturan [6] proposed a model and corresponding heuristic algorithms for solving multi-unit non-discriminatory combinatorial auction problems. Their study

also proves the NP-hard nature of the winner determination problem, which our VM placement model generalizes.

Collectively, these approaches demonstrate the effectiveness of bio-inspired algorithms in achieving energy-aware, SLA-conscious, and performance-efficient VM scheduling.

While minimizing energy consumption is critical in cloud data centers, it must not come at the cost of violating service-level agreements (SLAs) or degrading the quality of service (QoS). Consequently, many studies have aimed to design VM placement mechanisms that maintain high reliability and user satisfaction.

Li et al. [7] proposed a dynamic consolidation strategy that balances SLA violations with energy consumption, offering adaptive migration control based on workload fluctuations. Fu et al. [8] introduced a dual-criteria migration technique that selects VMs for migration based on both energy savings and SLA constraints, improving responsiveness to changing demand.

Zhou et al. [9] designed a high-reliability VM placement method that improves system fault tolerance and reduces downtime, thereby minimizing the risk of SLA breaches. Alharbi et al. [10] introduced an intelligent scheduling method for SLA-aware workload classification and allocation, which considers resource demand predictability in the placement decision.

These contributions show that integrating SLA-awareness into VM consolidation enhances cloud infrastructure reliability while maintaining energy efficiency and workload stability.

Cloud virtual machine placement involves multiple conflicting objectives, such as minimizing energy consumption, reducing SLA violations, and achieving high resource utilization. Multi-objective optimization (MOO) frameworks are widely adopted to balance these trade-offs effectively.

Ahmad et al. [11] introduced an adaptive multi-objective task scheduling approach using a chaos-based whale optimization algorithm to reduce energy consumption and improve load balance. Singh et al. [12] developed a hybrid model based on Particle Swarm Optimization (PSO) and Grey Wolf Optimization (GWO) for VM consolidation, targeting SLA awareness and energy reduction.

Alshathri et al. [13] focused on improving load balancing in multi-clouds using a hybrid multi-objective algorithm that incorporates latency, load variation, and energy trade-offs. Talwani et al. [14] presented a fuzzy-based adaptive VM placement strategy that dynamically adapts to workload changes while optimizing power usage and SLA adherence.

Tran et al. [15] proposed a reinforcement learning-based framework to handle energy and performance objectives simultaneously, demonstrating its effectiveness in dynamic and heterogeneous cloud environments.

These approaches demonstrate how MOO strategies enable dynamic and scalable solutions for VM placement by handling energy, SLA, and performance objectives in a unified decision framework.

Most earlier work either looks at placement only at a single point in time, or considers migration only indirectly, which hides how decisions are linked across time and the actual cost of moving VMs. We instead formulate a multi-period, energy and migration-aware MIP and derive fast heuristics tailored to that formulation. Meroni and Guitart propose energy-aware allocation using mathematical programming with micro/macro modeling and warm starts [16]; our framework complements that direction by making migration cost explicit over time and by characterizing the accuracy–runtime trade-off under varied capacity and load.

III. EMM-Sched Model Overview

This section gives an intuitive tour of *EMM-Sched* before its formal definition in Section IV. We outline the system context and request language, then the period-by-period loop, and finally key features.

We target an Infrastructure-as-a-Service provider operating a set of physical machines P. Each machine $p \in P$ is described by a resource capacity vector $\langle c_{p,\mathrm{CPU}}, c_{p,\mathrm{RAM}}, c_{p,\mathrm{SSD}}, c_{p,\mathrm{HDD}}, \dots \rangle$ and a linear power model $\langle i_p, M_p \rangle$, where i_p and M_p denote idle and peak power, respectively. An energy price e_p (\$/kWh) is attached to every machine, allowing location-aware optimization.

Workload arrives as VM requests. A request r is an AND/OR tree that decomposes into sub-requests $s \in S_r$, each of which offers a finite set V_{rs} of VM configurations in a disjunctive OR alternatives. EMM-Sched may mix alternatives in V_{rs} for the same sub-request (i.e., q_{rs} VMs can be split across types). For every alternative $v \in V_{rs}$ we record (i) a resource-demand vector d^a_{rsv}, (ii) a unit price π_{rsv} (commercial clouds), and (iii) a migration penalty μ_{rsv}, set to $+\infty$ for stateful or latency-critical VMs. The sub-request specifies a quantity q_{rs} of VMs; the scheduler decides how to distribute q_{rs} across machines and alternatives $v \in V_{rs}$.

Periods are separated by a configurable evaluation interval. At the boundary between two periods we perform three steps (Figure 1):

i. *State collection*: observe completions and newly arrived requests.
ii. *Optimization*: solve the MIP of Section IV to place legacy and new VMs jointly.
iii. *Migration window*: execute prescribed migrations if their benefit outweighs μ_{rsv}.

This closed loop adapts placement to workload change and time-of-day energy prices.

The design features of the model are:

- At each boundary we re-solve a single-period MIP using the observed legacy state. Migration/reconfiguration penalties μ_{rsv} prevent unnecessary changes; repeated single-period solves implement a multi-period policy.
- Migrations are decided alongside placement, and only if their benefit outweighs μ_{rsv}.
- The model maximizes utility net of migration and energy costs (see (1)). To avoid price chasing of a single expensive alternative, we value each sub-request at the average of its offered prices; the mix chosen among OR alternatives does not change revenue.

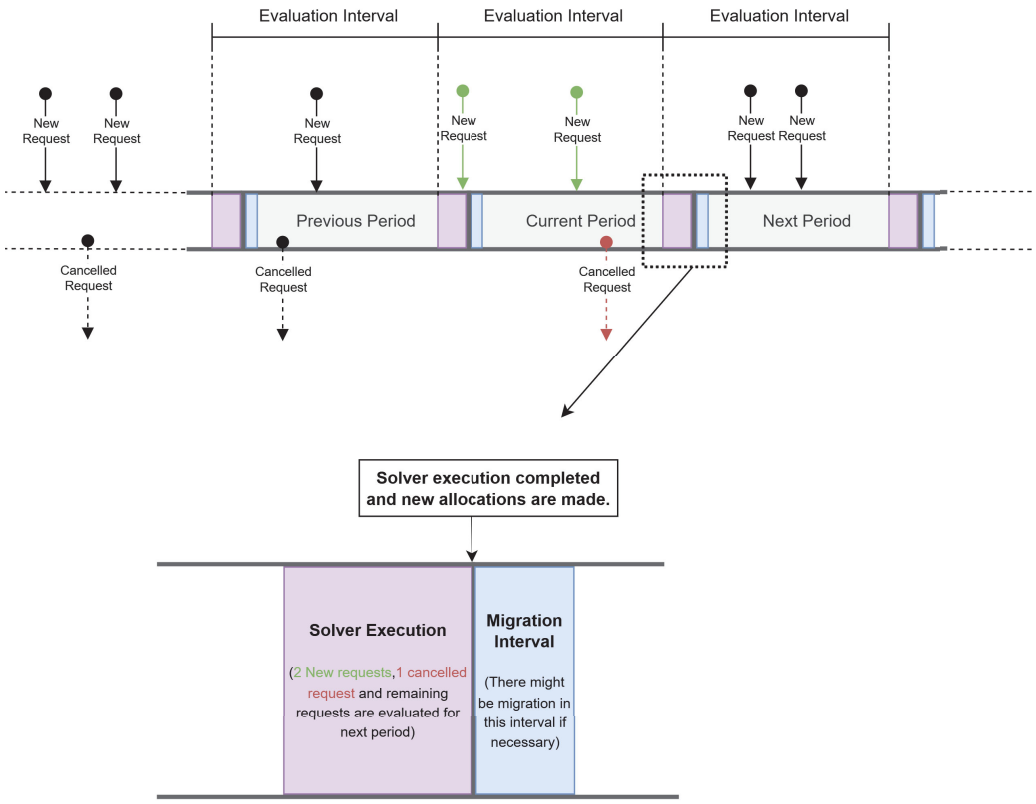

Fig. 1. High-level scheduling loop executed by EMM-Sched at every period boundary

- An exact MIP yields optimal solutions for moderate instances, whereas the heuristics of Section V scale to thousands of VMs within seconds.

IV. Mathematical Model

We schedule incoming VM requests, place legacy VMs, and decide which machines to power on so as to maximize revenue (or priority), while paying migration and energy costs.

Notation:

- P set of physical machines p.
- A set of resource attributes a (e.g., CPU, RAM).
- R set of new VM requests r; each has sub-requests S_r.
- R^{old} legacy requests already running at period start.
- V_{rs} alternative VM types for sub-request (r, s) (indexed by v).
- $c_{p,a}$ capacity of attribute a on machine p.
- d^a_{rsv} demand of attribute a by alternative v.
- q_{rs} number of VMs in sub-request (r, s).
- π_{rsv} unit price of alternative v.
- μ_{rsv} migration cost of alternative v (set to $+\infty$ for non-migratable VMs).
- i_p idle power of machine p; M_p peak power.
- e_p energy price for machine p; t period length (hours).
- y^{old}_{rsvp} legacy VMs of type v from (r, s) residing on p at period start.

Decision variables:

- $x_r \in \{0, 1\}$ equals 1 if request r is accepted.
- $y_{rsvp} \in \mathbb{Z}_{\geq 0}$ VMs of type v from (r, s) placed on p.
- $m_{rsvp} \in \mathbb{Z}_{\geq 0}$ (defined only for $r \in R^{\text{old}}$) VMs of type v reconfigured away from p.
- $u_{p,a} \in \mathbb{R}_{\geq 0}$ total load of attribute a on p.
- $z_p \in \{0, 1\}$ equals 1 if machine p is powered on.
- $P_p \in \mathbb{R}_{\geq 0}$ electrical power drawn by p.

Mixed-Integer Linear Program (MIP):

$$\max \underbrace{\sum_{r \in R \cup R^{\text{old}}} \Big(x_r \sum_{s} \frac{q_{rs}}{|V_{rs}|} \sum_{v \in V_{rs}} \pi_{rsv} \Big)}_{\text{revenue / priority}} - \underbrace{\sum_{r \in R^{\text{old}}} \sum_{s,v,p} \mu_{rsv} m_{rsvp}}_{\text{migration}} - \underbrace{t \sum_{p} e_p P_p}_{\text{energy}} . \tag{1}$$

$$s.t. \quad x_r = 1 \qquad \forall r \in R^{\text{old}} \qquad (2)$$

$$\sum_{v,p} y_{rsvp} = q_{rs} x_r \qquad \forall r,\ s \qquad (3)$$

$$\sum_{r,s,v} d^a_{rsv} y_{rsvp} = u_{p,a} \qquad \forall p,\ a \qquad (4)$$

$$u_{p,a} \le c_{p,a} z_p \qquad \forall p,\ a \qquad (5)$$

$$m_{rsvp} \ge y^{\text{old}}_{rsvp} - y_{rsvp} \qquad \forall r \in R^{\text{old}}, s, v, p \qquad (6)$$

$$m_{rsvp} \le y^{\text{old}}_{rsvp} \qquad \forall r \in R^{\text{old}}, s, v, p \qquad (7)$$

$$i_p\, z_p + (M_p - i_p)\, \tfrac{u_{p,1}}{c_{p,1}} = P_p \qquad \forall p \qquad (8)$$

Since sub-requests are OR alternatives, constraint (3) lets the solver split q_{rs} across multiple alternatives v and machines p. A decrease in y_{rsvp} versus y^{old}_{rsvp} is charged by m_{rsvp}; this includes moving to another machine and changing type v (even on the same machine), hence the term *migration/reconfiguration*. Constraint (2) retains all legacy requests. Constraints (4)–(5) govern per-machine utilization and capacities. Constraints (6)–(7) capture migration requirements. Constraint (8) models power using CPU (attribute 1) as the proxy.

Finally, this optimization problem is NP-hard since it generalizes the winner determination problem of the multi-unit nondiscriminatory combinatorial auction, which is NP-hard [6].

V. Heuristic Algorithms

To scale EMM-Sched to thousands of VMs and multi-period horizons, we implement two heuristic solvers: EMM-FFD and EMM-BFD. Each period executes the same loop: (i) maintain legacy instances (remove those with time-to-live TTL$=1$, decrement TTL, clear migration flags), (ii) optionally migrate surviving instances according to the solver's criterion, (iii) admit and place new requests arriving at the current period with full rollback on failure of any instance belonging to a request, and (iv) persist per-machine assignments and the period objective. TTL is the number of remaining scheduling periods that the instance is supposed to stay alive.

Both heuristics use the same power/energy model as Section IV: per-machine power is the sum of an idle term and a linear utilization term,

$$P_p \;=\; i_p z_p + (M_p - i_p)\,\frac{u_{p,1}}{c_{p,1}},$$

and the incremental energy used for placement/migration decisions is derived from this linear form, multiplied by the data center's energy price (optionally adjusted by PUE).

The period objective matches the model's profit,

$$\text{profit} \;=\; \sum (\text{realized prices of placed instances}) - \text{migration cost} \;-\; \text{energy cost}, \qquad (9)$$

with prices taken from the selected OR alternatives (i.e., realized income from the concrete VM types that are placed). Migration cost is charged per instance when a move occurs.

Algorithm 1 EMM-FFD (Energy-Minimizing First-Fit Decreasing)

Input: machine set P, requests R with AND/OR sub-requests
for period $\tau = 0, 1, \ldots$ **do**
  Maintain legacy: remove TTL$=1$, decrement TTL, clear migration flags
  for all migratable legacy instance i **do**
    pick $p^\star \in P$ minimizing $\text{EnergyCost}(p, i)$ over feasible p
    if $\text{EnergyCost}(p^\star, i) < \text{EnergyCost}(\text{current}(i), i)$ and $p^\star \neq \text{current}(i)$ **then**
      migrate i to $p^\star$; charge migration penalty
    end if
  end for
  Order new requests by descending average vCPU
  for request r arriving at τ **do**
    $accepted \leftarrow true$
    for each sub-request $s \in S_r$ **do**
      for $h = 1..q_{rs}$ **do**
        {repeat for required instance count}
        if s is AND-type **then**
          choose $p^\star$ minimizing $\text{EnergyCost}(p, s\downarrow)$ over feasible p
        else
          {OR-type}
          choose pair $(v^\star, p^\star)$ minimizing $\text{EnergyCost}(p, s\downarrow v)$ over feasible (p, v)
        end if
        if no feasible target found **then**
          $accepted \leftarrow false$; **break**
        end if
        place instance on $p^\star$; record realized price of chosen $v^\star$ (if OR)
      end for
    end for
    if not $accepted$ **then**
      rollback all placements of r
    end if
  end for
  Return placements
end for

A. EMM-FFD: Energy-Minimizing First-Fit Decreasing

EMM-FFD (see Algorithm 1) targets low energy with restrained migration. At every period, newly arrived requests are processed in descending order of average vCPU demand across their sub-requests.

Admission and placement:

- *AND sub-requests*: For each required instance, scan all PMs and select the target that minimizes incremental energy for the corresponding sub-request detail, subject to feasibility. If any instance cannot be placed, reject the whole request and roll back interim placements.

Algorithm 2 EMM-BFD (Best-Fit Decreasing)

for period $\tau = 0, 1, \ldots$ **do**
  Maintain legacy as in Alg. 1
  for all migratable legacy instance i **do**
    $s_0 \leftarrow \mathrm{CurrentFitScore}(\mathrm{current}(i))$
    pick $p^\star$ minimizing $s(p) = \mathrm{BestFitScore}(p, i)$ over feasible p
    if $p^\star \neq \mathrm{current}(i)$ **and** $s(p^\star) < s_0$ **then**
      migrate i to $p^\star$; charge migration penalty
    end if
  end for
  Admit & place new requests as in Alg. 1, but minimize $\mathrm{BestFitScore}$ (and record energy for costing)
  Return placements
end for

- *OR sub-requests*: For each required instance, jointly choose the alternative type and PM that minimize incremental energy among feasible pairs; apply the same rollback rule upon any failure.

Migration rule: For every migratable legacy instance, evaluate the best feasible target PM by incremental energy and migrate the instance only if the target's per-instance energy cost is strictly lower than the instance's current recorded energy cost. Charge the per-instance migration penalty on a move.

B. EMM-BFD: Best-Fit Decreasing

EMM-BFD (see Algorithm 2) follows a best-fit policy guided by a machine-level fit score. The score $\mathrm{BestFitScore}(p, \cdot)$ is non-negative for feasible placements and aggregates residual-capacity fragmentation and/or incremental power; lower is better.

Admission and placement:

- *AND sub-requests*: For each required instance, choose the feasible PM with minimal $\mathrm{BestFitScore}$.
- *OR sub-requests*: For each required instance, choose the feasible pair (alternative type, PM) with minimal $\mathrm{BestFitScore}$; full rollback on failure.

Migration rule: For every migratable legacy instance, let $\mathrm{CurrentFitScore}$ be the score on its present machine. Migrate the instance to the feasible PM with the lowest $\mathrm{BestFitScore}$ only if that score is strictly lower than $\mathrm{CurrentFitScore}$ (no explicit comparison against the migration penalty is performed), and then accrue the migration cost.

VI. Experimental Results

We compare three solvers: the MIP (Gurobi Optimizer v12.0), Energy-Minimizing First-Fit Decreasing (EMM-FFD), and Best-Fit Decreasing (EMM-BFD). We study four metrics under varying workload densities and infrastructure sizes: Objective Value, Energy Cost, Migration Cost, and Solution Time.

We use a test-case generator and produce two scenarios per parameter set with a total of 2,532 test instances. We set the number of slots to 10. Pricing for requestable instance types is taken from Amazon EC2 On-Demand rates and kept identical across all experiments. Capacity is tested at four levels: $\{3328, 6656, 9984, 13312\}$ vCPU cores.

The remaining parameter values are:

- **Request_Density** d: demand level relative to available vCPU capacity; tested at $d \in \{0.25, 0.50, 0.75, 1.00\}$.
- **Mean_SubRequest_Count**: Poisson-distributed mean $\in \{1, 2, 3\}$; average number of sub-requests per request.
- **Mean_Alternative_Request_Count**: Poisson-distributed mean $\in \{1, 2\}$; average number of alternative bundles for OR choices.
- **Mean_Instance_Count_In_SubRequest**: Poisson-distributed mean $\in \{1, 2, 3\}$; average VMs per sub-request.
- **Mean_SubRequest_Duration**: Poisson-distributed mean $\in \{1, 2, 3\}$; average VM lifetime in slots.
- **Mean_Migration_Enable_SubRequest**: Poisson-distributed level mean $\in \{1, 2, 3\}$; share of sub-requests marked migratable.

Figure 2 shows that the mean objective increases roughly linearly for all methods as capacity and request density grow, while solution time rises the most for the MIP due to search-space growth. With more cores, more requests can be admitted, giving the MIP a clear advantage in total revenue. In contrast, EMM-BFD and EMM-FFD keep a low and nearly flat time profile thanks to simpler logic and predictable placement, revealing a direct trade-off between profit and solve time.

Figure 3 shows a consistent pattern across capacities and densities. EMM-FFD provides the lowest energy and the lowest migration, while EMM-BFD results in higher energy and the highest migration. The MIP sits between the heuristics on migration but has the highest energy overall because it admits more work and activates more machines, which also explains its best net objective despite higher absolute costs. In terms of objective, the ordering is (MIP $>$ EMM-BFD $>$ EMM-FFD) at every density.

Figure 4 separates the objective–capacity relation across four density levels and shows that slopes become steeper and method gaps widen as d increases. At low density ($d = 0.25$) curves are milder and closer; at medium to high density ($d = 0.50$–0.75) the marginal gain from extra capacity grows, and the MIP rises faster than the others. Near peak load ($d \approx 1.00$), we observe diminishing returns at very high vCPU.

Table I indicates a consistent ordering for mean vCPU usage: MIP $>$ EMM-BFD $>$ EMM-FFD. At peak load ($d = 1.00$) the gap is largest (MIP: 94.80, EMM-BFD: 85.94, EMM-FFD: 79.49), showing that higher demand lets the MIP fill capacity more effectively. At light load ($d = 0.25$) the separation narrows (MIP: 87.65, EMM-BFD: 86.08, EMM-FFD: 76.15). The MIP also shows the lowest variability at each density (e.g., std. $= 10.52$ at $d = 1.00$), while EMM-BFD and EMM-FFD remain in the $\sim$21–23 band.

Active PM percentage (ratio of powered-on PMs to all PMs) increases with demand for all methods, rising from roughly 45–46% at $d = 0.25$ to 93–98% at $d = 1.00$. EMM-FFD is typically slightly above EMM-BFD (e.g., at $d = 0.75$, 86.22% vs. 85.27%), which suggests slightly more fragmentation and the need to keep a few extra machines on to achieve similar

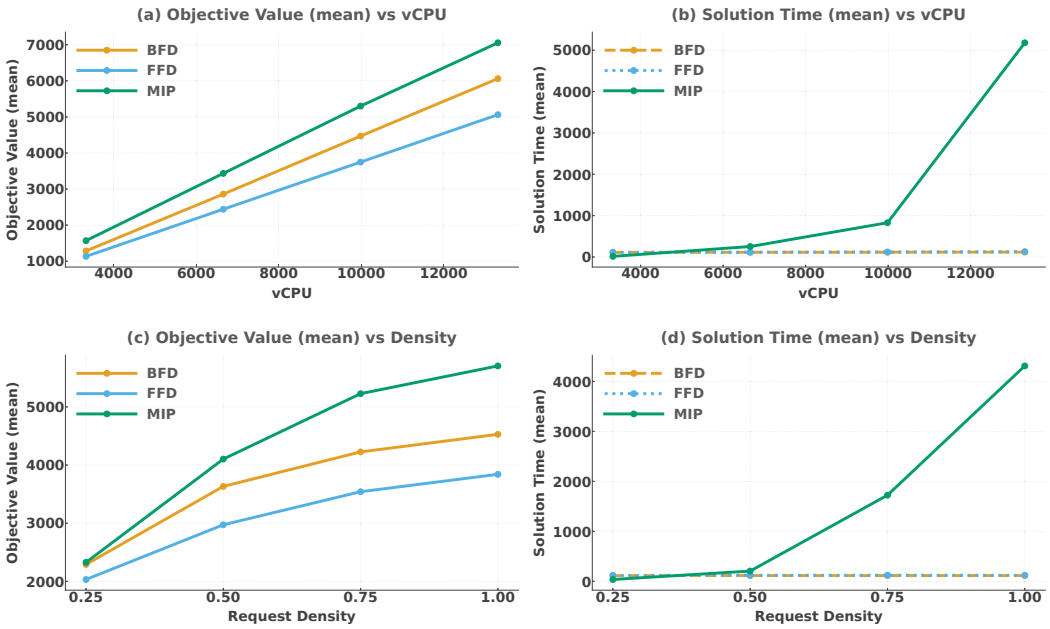

Fig. 2. General trends of mean objective value and solution time across capacity and load. (a) Objective vs. vCPU, (b) Solution_Time vs. vCPU, (c) Objective vs. Request_Density, (d) Solution_Time vs. Request_Density. Curves report method means (MIP, EMM-BFD, EMM-FFD) over densities $\{0.25, 0.50, 0.75, 1.00\}$ using vCPU $\in \{3328, 6656, 9984, 13312\}$.

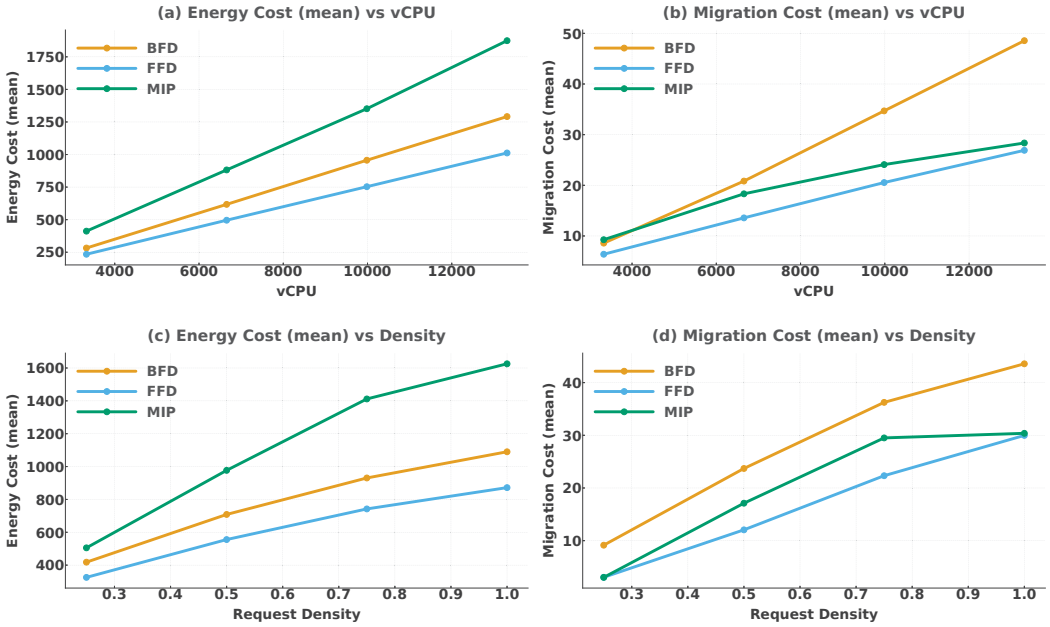

Fig. 3. Mean energy and migration costs across capacity and load. (a) Energy vs. vCPU, (b) Migration vs. vCPU, (c) Energy vs. Request_Density, (d) Migration vs. Request_Density. Curves report method means (MIP, EMM-BFD, EMM-FFD) over densities $\{0.25, 0.50, 0.75, 1.00\}$.

usage. The MIP activates the most PMs at high load (e.g., 98.03% at $d = 1.00$), consistent with its stronger acceptance and packing behavior.

Figure 5 tracks mean vCPU usage over time period for $d \in \{0.25, 0.50, 0.75, 1.00\}$. As d increases, all methods move to higher utilization with a stable ordering: the MIP at the top, EMM-BFD close behind, and EMM-FFD in third place. The time profiles are flat at low load; at medium and high load there is a short ramp followed by a plateau, indicating that admission and placement settle within the first few slots and then remain steady.

From a deployment angle, if the target is high utilization (and by extension revenue), the MIP sustains that level consistently. If energy, operational simplicity, and speed matter, EMM-BFD offers a close alternative, while EMM-FFD remains a fast and predictable baseline. Near $d \approx 1.00$, diminishing returns

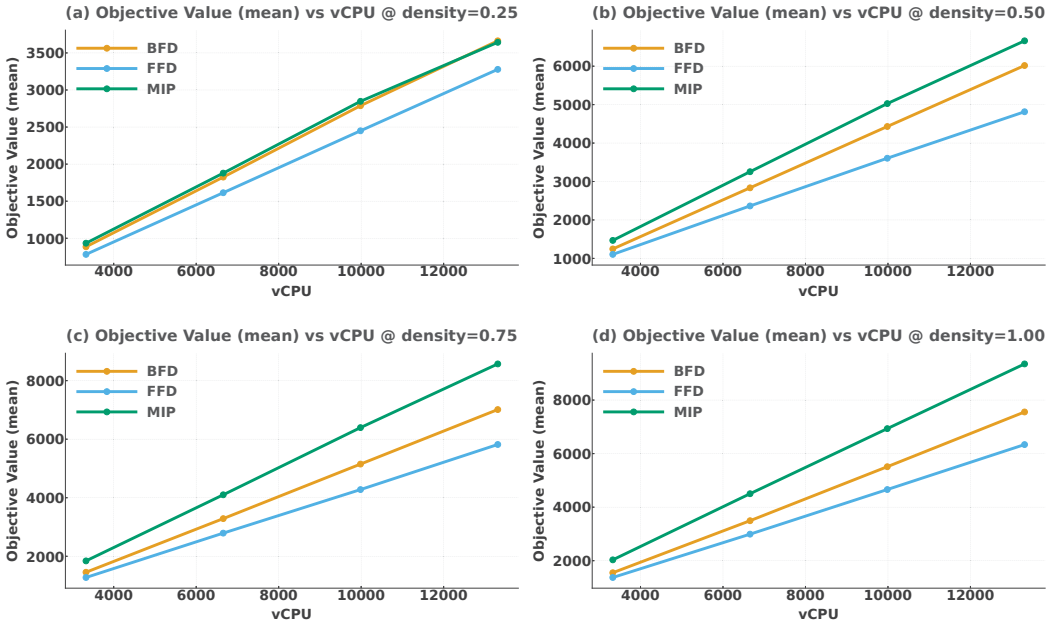

Fig. 4. Mean objective value vs. capacity across four request densities $d \in \{0.25, 0.50, 0.75, 1.00\}$. Curves report method means (MIP, EMM-BFD, EMM-FFD) over vCPU $\in \{3328, 6656, 9984, 13312\}$.

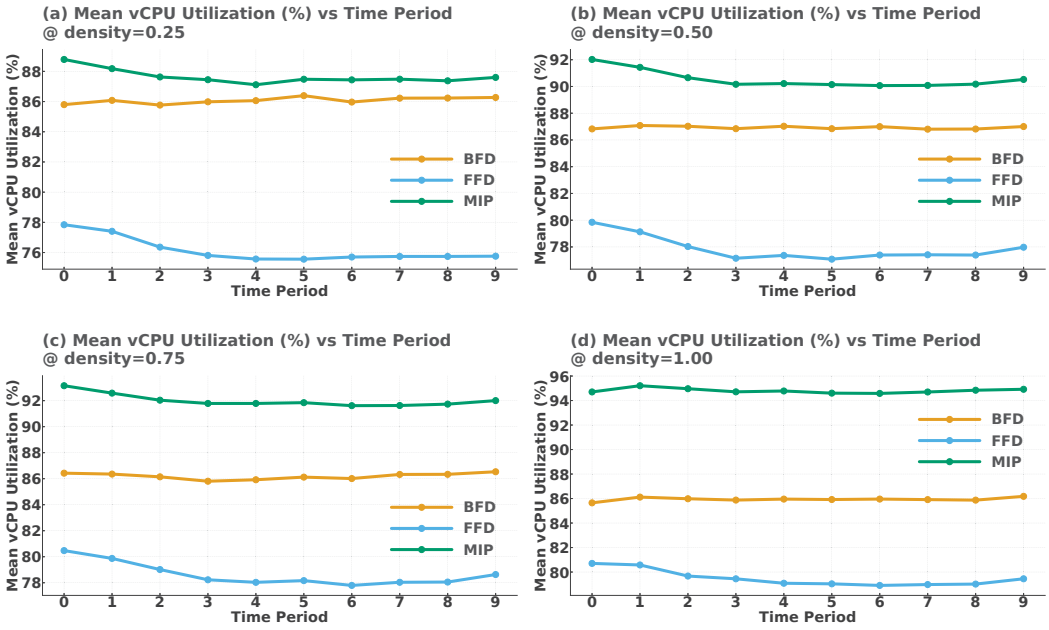

Fig. 5. Mean vCPU usage over time period at four request densities.

appear: beyond a point, extra capacity adds little to utilization, so planning should weigh usage alongside energy and migration costs.

VII. Conclusion

We introduced a unified, energy-aware, multi-period, migration-aware scheduler that optimizes admission, multi-dimensional placement, server activation, and migration under a profit-centered objective. The model integrates a linear power model with variable energy prices and exposes an expressive AND/OR request interface that lets users describe alternatives precisely. We paired an exact MIP with two scalable heuristics and evaluated them across capacities and load densities. The MIP consistently attained the highest objective, while the heuristics recovered a large share of that objective with much lower runtime and, in the case of EMM-FFD, lower energy and migration. These results clarify when exact optimization is preferable (offline planning, longer horizons) and when a heuristic is the practical choice (tight latency budgets, frequent re-optimization).

TABLE I. MEANS GROUPED BY REQUEST DENSITY AND METHOD. BOTTOM ROWS REPORT METHOD-WISE MEANS ACROSS ALL DENSITIES.

Density	Method	vCPU util. Mean	vCPU util. Std. Dev.	Active PM Ratio (%)
0.25	EMM-BFD	86.08	20.93	45.63
0.25	EMM-FFD	76.15	21.99	46.01
0.25	MIP	87.65	15.10	44.78
0.50	EMM-BFD	86.93	20.72	70.82
0.50	EMM-FFD	77.88	22.11	71.24
0.50	MIP	90.55	13.49	74.21
0.75	EMM-BFD	86.20	21.88	85.27
0.75	EMM-FFD	78.63	22.66	86.22
0.75	MIP	92.02	13.36	92.55
1.00	EMM-BFD	85.94	22.35	93.00
1.00	EMM-FFD	79.49	22.85	93.71
1.00	MIP	94.80	10.52	98.03
Overall	EMM-BFD	86.29	21.47	73.68
Overall	EMM-FFD	78.04	22.40	74.29
Overall	MIP	91.25	13.12	77.39

Future work includes (i) a piecewise-linear energy model to better capture server power curves, (ii) a multi–data-center formulation spanning different regions, and (iii) explicit inter–data-center migration costs in both objective and constraints. On the algorithmic side, we aim to strengthen heuristic functions to retain near-MIP solution quality while further reducing runtime at scale.

REFERENCES

[1] B. Pourghebleh, A. A. Anvigh, A. R. Ramtin, and B. Mohammadi, "The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments," *Cluster Computing*, vol. 24, pp. 2673–2696, 9 2021.

[2] S. E. Dashti and A. M. Rahmani, "Dynamic vms placement for energy efficiency by pso in cloud computing," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 28, pp. 97–112, 3 2016.

[3] M. Tang and S. Pan, "A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers," *Neural Processing Letters*, vol. 41, pp. 211–221, 4 2015.

[4] Q. Zheng, R. Li, X. Li, N. Shah, J. Zhang, F. Tian, K. M. Chao, and J. Li, "Virtual machine consolidated placement based on multi-objective biogeography-based optimization," *Future Generation Computer Systems*, vol. 54, pp. 95–122, 1 2016.

[5] M. Abdel-Basset, L. Abdle-Fatah, and A. K. Sangaiah, "An improved lévy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment," *Cluster Computing*, vol. 22, pp. 8319–8334, 7 2019.

[6] A. H. Özer and C. Özturan, "A model and heuristic algorithms for multi-unit nondiscriminatory combinatorial auction," *Computers & Operations Research*, vol. 36, no. 1, pp. 196–208, 2009. [Online]. Available: https://doi.org/10.1016/j.cor.2007.08.012

[7] Z. Li, X. Yu, L. Yu, S. Guo, and V. Chang, "Energy-efficient and quality-aware vm consolidation method," *Future Generation Computer Systems*, vol. 102, pp. 789–809, 1 2020.

[8] X. Fu and C. Zhou, "Virtual machine selection and placement for dynamic consolidation in cloud computing environment," *Frontiers of Computer Science*, vol. 9, pp. 322–330, 4 2015.

[9] A. Zhou, S. Wang, B. Cheng, Z. Zheng, F. Yang, R. N. Chang, M. R. Lyu, and R. Buyya, "Cloud service reliability enhancement via virtual machine placement optimization," *IEEE Transactions on Services Computing*, vol. 10, pp. 902–913, 11 2017.

[10] F. Alharbi, Y. C. Tian, M. Tang, W. Z. Zhang, C. Peng, and M. Fei, "An ant colony system for energy-efficient dynamic virtual machine placement in data centers," *Expert Systems with Applications*, vol. 120, pp. 228–238, 4 2019.

[11] F. Ahmad, M. Shahid, M. Alam, Z. Ashraf, M. Sajid, K. Kotecha, and G. Dhiman, "Levelized multiple workflow allocation strategy under precedence constraints with task merging in iaas cloud environment," *IEEE Access*, vol. 10, pp. 92 809–92 827, 2022.

[12] A. K. Singh, S. R. Swain, and C. N. Lee, "A metaheuristic virtual machine placement framework toward power efficiency of sustainable cloud environment," *Soft Computing*, vol. 27, pp. 3817–3828, 4 2023.

[13] S. Alshathri, F. M. Talaat, and A. A. Nasr, "A new reliable system for managing virtual cloud network," *Computers, Materials and Continua*, vol. 73, pp. 5863–5885, 2022.

[14] S. Talwani, K. Alhazmi, J. Singla, H. J. Alyamani, and A. K. Bashir, "Allocation and migration of virtual machines using machine learning," *Computers, Materials and Continua*, vol. 70, pp. 3349–3364, 2022.

[15] C. H. Tran, T. K. Bui, and T. V. Pham, "Virtual machine migration policy for multi-tier application in cloud computing based on q-learning algorithm," *Computing*, vol. 104, pp. 1285–1306, 6 2022.

[16] R. Meroni and J. Guitart, "Scalable energy-aware VM allocation on cloud data centers through mathematical programming models," *Future Generation Computer Systems*, vol. 174, p. 108011, 2026, online: July 16, 2025.