Avg-Frontier: A Low-Overhead Multi-objective Batch Scheduler for Heterogeneous Edge Environments

Aleksei Chernigov and Vladimir Korkhov(✉)

St Petersburg State University, St Petersburg 199034, Russia
v.korkhov@spbu.ru

Abstract. Task scheduling in heterogeneous edge environments involves conflicting objectives related to completion time and energy consumption. Classical low-overhead heuristics usually optimize one dominant criterion, whereas population-based multi-objective methods provide richer trade-off analysis at a higher decision cost. This paper studies a batch scheduler, called *avg_frontier*, intended for repeated scheduling in such environments. The proposed method is organized as a two-stage procedure. First, it constructs an aggregated surrogate model of available devices using average time and energy coefficients over feasible tasks. The scheduler solves a discrete bi-objective allocation problem in the space of task counts, builds a surrogate Pareto frontier, and selects a compromise point according to the *area_ratio* policy controlled by α. Second, the selected allocation is restored into a real task-to-device assignment and re-evaluated by exact simulation. The algorithm is evaluated in SimGrid on two heterogeneous testbeds: a compact three-node profile with 60 tasks and a generated 100-node profile with 6000 tasks. The comparison includes classical heuristics and Pareto-based evolutionary methods. The results show that *avg_frontier* provides a controllable transition between time-oriented and energy-oriented regimes while keeping the scheduler overhead substantially lower than population-based Pareto baselines. The method is not intended to dominate all alternatives in every region of the trade-off, but to provide a computationally cheap mechanism for navigating this trade-off under repeated scheduling.

Keywords: Edge Scheduling · Heterogeneous Systems · Multi-Objective Optimization · Energy-Aware Scheduling · Simulation

1 Introduction

Edge and fog environments are characterized by pronounced hardware and network heterogeneity. In such systems, the scheduler has to operate not only with different computational capabilities of devices, but also with non-uniform communication costs, energy models, and changing availability conditions [14,16,19]. In practice, this leads to a direct conflict between at least two objectives. On

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16763, pp. 493–510, 2027.
https://doi.org/10.1007/978-3-032-30530-5_31

the one hand, the system must process tasks with low completion time. On the other hand, the same decisions influence energy consumption of both computing devices and communication channels.

A significant part of the scheduling literature relies on classical heuristics. These methods remain relevant because they are computationally simple and often provide reasonable schedules in heterogeneous systems [1,11–13,15,22,24]. However, such methods are usually built around a single dominant criterion or around a fixed deterministic rule. As a consequence, they are less convenient in the case when one needs to move between time-oriented and energy-oriented regimes without changing the scheduling framework itself.

Another direction is the use of multi-objective optimization methods, including evolutionary algorithms and decomposition-based approaches [6,7,23,25,26]. These methods are natural for approximating the Pareto set. At the same time, their use inside an operational scheduler raises a practical question: if the scheduling decision is repeated many times during system execution, the decision procedure itself becomes part of the total system cost.

The purpose of this paper is to study a lightweight multi-objective batch scheduler for heterogeneous edge environments. The scheduler, called *avg_frontier*, is based on a different principle than direct item-level Pareto optimization. Instead of searching in the full combinatorial space of task-to-device assignments, it first compresses the current batch into a surrogate allocation problem defined over the numbers of tasks assigned to devices. In this aggregated space, the method constructs an explicit discrete Pareto frontier with respect to surrogate makespan and surrogate active energy, selects one compromise point according to a prescribed front policy, and only after that restores the corresponding item-level assignment. Thus, the multi-objective search is performed in a substantially smaller space, while the final plan remains connected to the exact simulated execution model.

The contribution of the paper can be summarized as follows.

1. A compact mathematical formulation of the batch scheduling problem is given together with its surrogate aggregated version in the space of device load counts, where makespan and active energy remain explicit optimization criteria.
2. A low-overhead scheduler, *avg_frontier*, is described. The method explicitly builds a discrete surrogate Pareto frontier, selects one operating point using the *area_ratio* policy controlled by parameter α, restores the corresponding item-level assignment, and then performs exact re-evaluation of the obtained plan.
3. A simulation-based evaluation is performed on two heterogeneous testbeds: a compact three-node profile and a generated 100-node profile. The comparison includes classical heuristics and Pareto-based evolutionary algorithms and explicitly considers both schedule quality and scheduler overhead.

The rest of the paper is organized as follows. Section 2 summarizes related work. Section 3 introduces the problem formulation. Section 4 describes the pro-

posed scheduler. Section 5 presents the simulation environment and the experimental profiles. Section 6 discusses the obtained results. Section 7 outlines the main limitations of the study. Section 8 concludes the paper.

2 Related Work

2.1 Scheduling in Heterogeneous Environments

Task mapping in heterogeneous systems has been extensively studied. Surveys and comparative studies cover both static scheduling and independent-task mapping under common assumptions [1,12]. Classical methods such as OLB, MET, MCT, Min-Min, Max-Min, and Sufferage remain widely used baselines because they are simple and have low decision cost [13,15]. For dependent tasks, HEFT remains a standard example of low-complexity list scheduling [22].

These heuristics are attractive in practice, but they are usually centered on one dominant criterion, most often completion time. This makes them less convenient when the scheduler must move between time-oriented and energy-oriented regimes within one framework.

2.2 Energy-Aware and Multi-objective Scheduling

Joint consideration of time and energy has led to energy-aware scheduling methods for heterogeneous and distributed systems [4,10,17]. Specialized examples include energy-efficient scheduling for batch-of-tasks applications and weighted energy- and makespan-aware scheduling in cloud environments [18,21].

Multi-objective optimization provides a natural framework for such problems [6]. Frequently used methods include NSGA-II, SPEA2, MOEA/D, and multi-objective local search [7,9,23,26]. These methods approximate the Pareto set, but their decision cost becomes important when optimization is repeated during normal scheduler operation. This creates an intermediate niche between lightweight single-solution heuristics and more expensive Pareto-based procedures.

Thus, a gap remains between two classes of approaches. On one side, there are lightweight heuristics with limited multi-objective flexibility. On the other side, there are more expressive Pareto-based procedures with a substantially larger decision cost. The present work addresses this intermediate space.

2.3 Edge Orchestration and Simulation Environments

Edge computing has intensified interest in resource management, placement, and orchestration under heterogeneous constraints [14,16,19]. Examples include fog application deployment, fuzzy workload orchestration, and multi-objective provisioning in edge clouds [2,5,20]. These studies emphasize that operational decisions must often be made repeatedly under changing conditions.

Simulation is useful here because real edge infrastructures are costly to control and reproduce. iFogSim is a common tool in this area [8]. In this work,

SimGrid is used because it provides a scalable basis for modeling distributed platforms [3]. This allows explicit representation of device heterogeneity, communication parameters, and energy-aware metrics while preserving repeatability of the experiments.

3 Problem Formulation

Let

$$J = \{1, \ldots, N\}$$

be the current batch of independent tasks and

$$I = \{1, \ldots, M\}$$

be the set of available compute devices.

Each task $j \in J$ is characterized by its input data size, computational demand, output data size, and source node. In the present study, the scalar quantity

$$w_j = \max(0, \text{computeSize}_j)$$

is used as the main measure of task computational complexity.

A scheduling decision is defined as a mapping

$$\phi : J \to I,$$

where $\phi(j) = i$ means that task j is assigned to device i. In the present model, the communication route is not an independent decision variable. For every task–device pair (j, i), the route used for input and output transfer is determined by the routing policy and the current system state. Therefore, feasibility and the corresponding time and energy estimates are defined for the induced task–device assignment rather than optimized over routes separately.

For each feasible pair (j, i), the simulator provides route-aware estimates of input transfer time, execution time, and output transfer time,

$$t_{ij}^{in}, \qquad t_{ij}^{ex}, \qquad t_{ij}^{out},$$

as well as the corresponding energy estimates,

$$e_{ij}^{in}, \qquad e_{ij}^{ex}, \qquad e_{ij}^{out}.$$

Accordingly, the estimated completion-time contribution of assigning task j to device i is

$$T_{ij} = t_{ij}^{in} + t_{ij}^{ex} + t_{ij}^{out},$$

and the corresponding active-energy contribution is

$$E_{ij} = e_{ij}^{in} + e_{ij}^{ex} + e_{ij}^{out}.$$

Since devices may already be partially occupied at the scheduling instant, the final completion time of the whole batch depends not only on the task-to-device mapping itself, but also on the current device loads, queueing effects, and the order in which assigned tasks are executed. Therefore, the exact batch makespan is a functional of the full assignment ϕ and of the current system state. We denote it by

$$T(\phi).$$

Similarly, the exact active energy of the obtained plan is denoted by

$$E(\phi).$$

The scheduling problem considered in this paper is the following bi-objective optimization problem:

$$\min_{\phi} \big(T(\phi), E(\phi)\big),$$

subject to feasibility of all assignments contained in ϕ.

Thus, the scheduler must assign all tasks in the current batch to heterogeneous devices while balancing completion time and active energy. Direct optimization in the full assignment space is combinatorial and becomes impractical when the decision is rebuilt repeatedly. Therefore, the proposed method uses a lower-dimensional surrogate problem, restores its solution into an item-level assignment, and re-evaluates the result by exact simulation.

4 Avg-Frontier Scheduling Algorithm

The proposed scheduler solves the original batch assignment problem indirectly. Its construction consists of two stages. At the first stage, the current batch is replaced by an aggregated surrogate allocation problem. Instead of assigning each task to a concrete device immediately, the scheduler first determines only the number of tasks that should be sent to each device. This reduced problem is used to construct a discrete surrogate Pareto frontier with respect to makespan and active energy. At the second stage, the selected surrogate allocation is restored into a concrete item-level assignment, after which the obtained plan is re-evaluated by exact simulation. Thus, the main idea of *avg_frontier* is to perform multi-objective search in a lower-dimensional aggregated space and only then reconstruct a feasible detailed schedule. The overall workflow of the proposed method is illustrated in Fig. 1.

Let $F_i \subseteq J$ be the set of tasks feasible for device i in the current system state. For every feasible pair (i, j), the simulator provides route-aware estimates of input, execution, and output times

$$t_{ij}^{in}, \quad t_{ij}^{ex}, \quad t_{ij}^{out},$$

and the corresponding energy estimates

$$e_{ij}^{in}, \quad e_{ij}^{ex}, \quad e_{ij}^{out}.$$

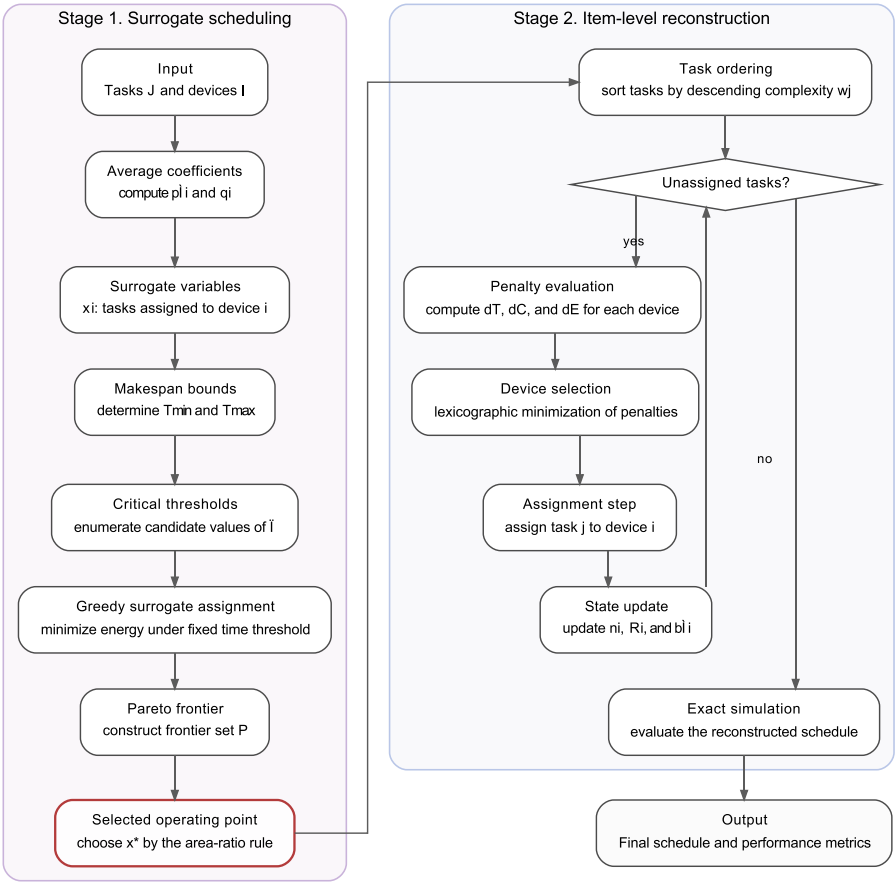

Fig. 1. Pipeline of the *avg_frontier* scheduling algorithm.

Using the task complexity measure w_j, define

$$W_i^{feas} = \sum_{j \in F_i} w_j.$$

Then the average time coefficients per unit complexity are

$$p_i^{in} = \frac{\sum_{j \in F_i} t_{ij}^{in}}{W_i^{feas}}, \qquad p_i^{ex} = \frac{\sum_{j \in F_i} t_{ij}^{ex}}{W_i^{feas}}, \qquad p_i^{out} = \frac{\sum_{j \in F_i} t_{ij}^{out}}{W_i^{feas}},$$

and the average energy coefficients are

$$q_i^{in} = \frac{\sum_{j \in F_i} e_{ij}^{in}}{W_i^{feas}}, \qquad q_i^{ex} = \frac{\sum_{j \in F_i} e_{ij}^{ex}}{W_i^{feas}}, \qquad q_i^{out} = \frac{\sum_{j \in F_i} e_{ij}^{out}}{W_i^{feas}}.$$

The total specific active-energy coefficient is

$$q_i = q_i^{in} + q_i^{ex} + q_i^{out}.$$

To approximate the time contribution of device i, the scheduler uses

$$\hat{p}_i = p_i^{ex} + \beta_{in} p_i^{in} + \beta_{out} p_i^{out},$$

where $\beta_{in} \geq 0$ and $\beta_{out} \geq 0$ control the influence of communication on the surrogate time model. In the reported experiments we use $\beta_{in} = 1$ and $\beta_{out} = 0$. This choice reflects the execution model adopted in the simulator.

In addition, each device is characterized by its current ready time b_i and by the maximum number of currently feasible tasks

$$u_i = |F_i|.$$

Let

$$W = \sum_{j=1}^{N} w_j, \qquad \bar{c} = \frac{W}{N}$$

be the total and average complexity of the current batch. The surrogate decision variable is the integer vector

$$x = (x_1, \ldots, x_M),$$

where x_i denotes the number of tasks assigned to device i. It satisfies

$$\sum_{i=1}^{M} x_i = N, \qquad 0 \leq x_i \leq u_i.$$

Under the average-complexity approximation, the surrogate completion time of device i is

$$T_i(x_i) = b_i + \hat{p}_i \bar{c}\, x_i.$$

Hence, the surrogate makespan and active energy are defined as

$$T(x) = \max_{i \in I} \left(b_i + \hat{p}_i \bar{c}\, x_i \right), \qquad E(x) = \sum_{i=1}^{M} q_i \bar{c}\, x_i.$$

Thus, the first stage solves the discrete bi-objective problem

$$\min_x \big(T(x), E(x) \big)$$

subject to the above feasibility constraints.

To construct the surrogate frontier, for a fixed time threshold τ define the capacity of device i as

$$\mathrm{cap}_i(\tau) = \max\left(0, \min\left\{ N,\ u_i,\ \left\lfloor \frac{\tau - b_i}{\hat{p}_i \bar{c}} \right\rfloor \right\}\right).$$

The value τ is feasible if

$$\sum_{i=1}^{M} \mathrm{cap}_i(\tau) \geq N.$$

Therefore, the minimum achievable surrogate makespan is

$$T_{\min} = \min \left\{ \tau : \sum_{i=1}^{M} \mathrm{cap}_i(\tau) \geq N \right\}.$$

In practice, it is sufficient to enumerate only critical values

$$\tau = b_i + \hat{p}_i \bar{c}\, r, \qquad r = 0, 1, \ldots, \min(N, u_i),$$

because the capacities change only at these points.

For each feasible τ, the corresponding minimum-energy allocation is obtained by minimizing

$$E(x) = \sum_{i=1}^{M} q_i \bar{c}\, x_i$$

under the constraints

$$\sum_{i=1}^{M} x_i = N, \qquad 0 \leq x_i \leq \mathrm{cap}_i(\tau).$$

Since the objective is linear, this allocation is constructed greedily: devices are sorted by nondecreasing $(q_i, \hat{p}_i, b_i, i)$, and tasks are assigned sequentially until all N tasks are distributed. Separately, the scheduler constructs the energy-extreme feasible allocation

$$x^E = \arg\min E(x)$$

subject to

$$\sum_{i=1}^{M} x_i = N, \qquad 0 \leq x_i \leq u_i.$$

That is, x^E is the feasible surrogate allocation with minimum active energy among all allocations assigning all N tasks. Since the objective is linear, x^E is obtained greedily by sorting devices by nondecreasing $(q_i, \hat{p}_i, b_i, i)$ and filling their capacities until all N tasks are assigned. We then define

$$T_{\max} = T(x^E).$$

Then all critical values $\tau \in [T_{\min}, T_{\max}]$ are enumerated. For each such τ, the corresponding energy-minimizing allocation $x(\tau)$ is constructed, and the point

$$\big(T(x(\tau)), E(x(\tau))\big)$$

is added to the candidate set. After dominated points are removed, the remaining set

$$P = \{x^{(1)}, x^{(2)}, \ldots, x^{(K)}\}$$

forms the discrete surrogate Pareto frontier.

One frontier point is then selected using the *area_ ratio* rule. Assume that the frontier points are ordered by increasing surrogate makespan. Define the ideal corner

$$(T^{\dagger}, E^{\dagger}) = \big(T(x^{(1)}), E(x^{(K)})\big).$$

For each adjacent pair of frontier points, compute the triangle area

$$A_k = \mathrm{Area}\Big((T^{\dagger}, E^{\dagger}), (T(x^{(k)}), E(x^{(k)})), (T(x^{(k+1)}), E(x^{(k+1)}))\Big), \qquad k = 1, \ldots, K-1,$$

and then the cumulative area ratio

$$\rho_k = \frac{\sum_{s=1}^{k-1} A_s}{\sum_{s=1}^{K-1} A_s}, \qquad k = 1, \ldots, K.$$

The selected surrogate point is

$$x^{\star} = \arg\min_{x^{(k)} \in P} |\rho_k - \alpha|,$$

where $\alpha \in [0, 1]$ is the preference parameter. Small α values correspond to the time-oriented part of the frontier, while large values correspond to the energy-oriented part.

After the surrogate point

$$x^{\star} = (x_1^{\star}, \ldots, x_M^{\star})$$

has been selected, the scheduler restores a real item-level assignment. For each device i, the initial restoration targets are

$$n_i^{(0)} = x_i^{\star}, \qquad R_i^{(0)} = \bar{c} x_i^{\star}, \qquad \tilde{b}_i^{(0)} = b_i,$$

where n_i is the remaining number of tasks to assign, R_i is the remaining target complexity, and $\tilde{b}_i$ is the current virtual surrogate load. Tasks are processed in nonincreasing order of complexity in order to preserve the selected surrogate structure more accurately.

For the current task j with complexity w_j, each feasible device i with $n_i > 0$ is evaluated. The updated virtual load is

$$\tilde{b}_i^{+} = \tilde{b}_i + \hat{p}_i w_j,$$

and the corresponding time penalty is

$$\Delta T_i = \max\{0, \tilde{b}_i^{+} - T(x^{\star})\}.$$

The structural state is updated as

$$R_i^{+} = R_i - w_j, \qquad n_i^{+} = n_i - 1.$$

If the remaining unscheduled tasks have total complexity W_{rem} and count N_{rem}, define

$$\bar{w}_{rem} = \begin{cases} W_{rem}/N_{rem}, & N_{rem} > 0, \\ 0, & N_{rem} = 0. \end{cases}$$

Then the structural deviation penalty is

$$\Delta C_i = \begin{cases} |R_i^+|, & n_i^+ = 0, \\ \left| \dfrac{R_i^+}{n_i^+} - \bar{w}_{rem} \right|, & n_i^+ > 0. \end{cases}$$

If $R_i^+ < 0$, the penalty is additionally increased by $|R_i^+|$. The approximate energy increment is

$$\Delta E_i = q_i w_j.$$

The selected device is the one that lexicographically minimizes

$$(\Delta T_i, \Delta C_i, \Delta E_i, \tau_i, \pi_i, \nu_i),$$

where τ_i is the exact completion-time estimate of the candidate assignment, π_i is a deterministic route-based tie-break, and ν_i is the node identifier. After assigning the task, the corresponding values of n_i, R_i, and $\tilde{b}_i$ are updated. If no device with $n_i > 0$ yields a feasible candidate, a fallback rule is used: the task is assigned to the feasible device with the minimum exact expected completion time.

Finally, after the full item-level assignment has been restored, the scheduler performs exact simulation of the obtained fixed plan. Let

$$\phi^\star : J \to I$$

be the reconstructed mapping. The simulator evaluates its exact characteristics

$$\big(T_{exact}(\phi^\star), E_{exact}(\phi^\star)\big),$$

and these exact values replace the surrogate estimates in the final stored decision. Therefore, the surrogate stage is used only to construct a compact search space and to select a compromise operating point, whereas the final schedule is always validated by the exact execution model.

5 Simulation Environment

5.1 General Setup

The experiments were conducted in a SimGrid-based discrete-event environment [3]. Platform descriptions included an XML topology, CSV files for device and channel types, and CSV files for workload and scenario description. This representation allowed the experimental profile to be changed without modifying the scheduling code.

Each compute node had a device type describing performance, idle and busy power, and transfer capabilities. Channels were described by bandwidth, latency, transmission and reception power, and startup and tail effects. The scheduler was invoked in *batch* mode: at each invocation it observed the current state, scheduled the newly available batch, and the simulation then proceeded to execution.

5.2 Workload Design

One task family was used to keep the comparison focused on scheduling behavior rather than on interactions between workload classes. The tasks were independent and batch-generated; for each task, the simulator modeled input transfer, computation, and output transfer. The small testbed served as an interpretable time–energy case, while the large testbed was used to evaluate decision cost under more devices and repeated scheduling calls.

5.3 Experimental Setup

Two testbeds were used. The small testbed depicted in Fig. 2, contained three heterogeneous compute nodes, $N0$, $N1$, and $N2$, with different performance, power, and communication characteristics. Its communication layer included WiFi-, USB-, and cellular-type links.

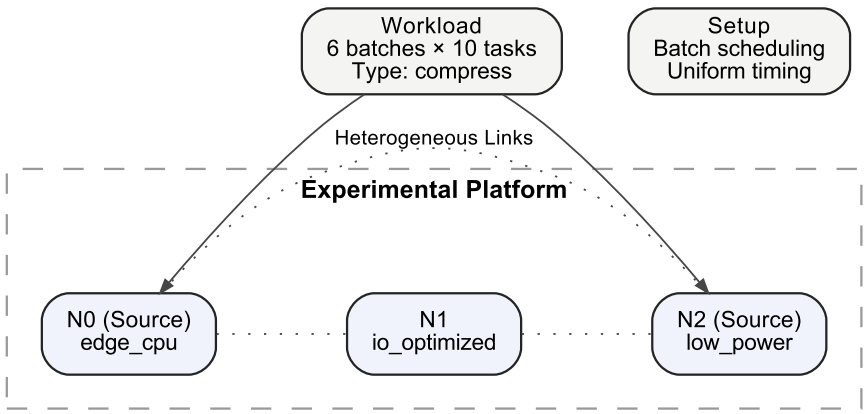

Fig. 2. Small experimental profile used in the simulation study. The platform contains three heterogeneous compute nodes connected by heterogeneous communication links. The workload consists of six consecutive batches of independent *compress* tasks generated from the available source nodes.

The small workload consisted of six batches of 10 independent *compress* tasks, for a total of 60 tasks, generated from $N0$, $N1$, and $N2$.

The large testbed was a generated 100-node profile. Node types were assigned cyclically from three classes: *edge_ cpu*, *io_ optimized*, and *low_ power*. The topology used offset-based connectivity; each logical edge had WiFi-, USB-, and cellular-type links. Five source nodes were placed approximately uniformly: $N0$, $N25$, $N50$, $N74$, and $N99$. The workload contained 600 batches of 10 independent tasks, for a total of 6000 tasks.

5.4 Baselines and Metrics

To evaluate the proposed method, it was compared with several groups of baseline schedulers.

The comparison included three groups of schedulers. The first group contained classical heuristics: `source_only`, `rr`, `olb`, `met`, `mct`, `min_min`, `max_min`, and `sufferage`. The second group contained energy-aware or hybrid heuristics: `minmin_energy`, `minmin_energy_time`, `minsuff`, `lws_greedy`, and `g2`. The third group contained Pareto-based methods: `nsga2`, `moead`, `spea2`, and `me_mls`. The `me_mls` method was evaluated only on the small testbed because it did not complete on the large testbed within practical time.

Implementation parameters were fixed as follows. For NSGA-II, MOEA/D, SPEA2, and ME-MLS, when used, the population size was 64, the archive size was 64, and the number of main iterations/generations was 60. For MOEA/D, the neighborhood size was 10 and the replacement limit was 2. For ME-MLS, the number of local-search iterations was 200. The crossover and mutation probabilities were set to 0.90 and 0.10, respectively. Seed solution generation used $\alpha_{seed} = 0.5$ and $\beta_{seed} = 0.5$. All experiments were run with a fixed random seed equal to 42. Representative points reported for multi-objective baselines were selected from the obtained nondominated sets using the same *area_ ratio* rule as for *avg_ frontier*. For `lws_greedy` and `g2`, equal weights were used for time and energy components (0.5/0.5). For *avg_ frontier*, the surrogate communication weights were fixed to $\beta_{in} = 1$ and $\beta_{out} = 0$, and tasks were restored in nonincreasing order of complexity.

For each method, the following metrics were collected:

- *successRate*, that is, the fraction of tasks completed successfully;
- *makespan*, which is the time it takes to complete all tasks;
- *activeEnergyJ*, that is, the active energy consumption measured in joules;
- *schedulerRealWallTimeSec*, that is, the real wall-clock time required for all scheduling decisions.

Among these metrics, makespan and active energy correspond directly to the optimization problem considered in this work. The scheduler wall-clock time was additionally recorded in order to estimate the practical computational cost of the decision procedure.

6 Results

6.1 General Remarks on Result Interpretation

All completed runs in both testbeds achieved $successRate = 1.0$, which allows direct comparison of the collected time and energy metrics without additional penalty handling. The reported comparison is based on makespan, active energy, and scheduler wall-clock time.

The two testbeds play different roles in the analysis. The three-node profile is used as a compact interpretable case in which the assignment structure and the time–energy trade-off can be inspected directly. The 100-node profile is used as a larger stress case for repeated scheduling and for comparing the decision cost of lightweight and population-based methods. Therefore, the conclusions should be interpreted as a controlled comparison on two simulated configurations rather than as an exhaustive empirical characterization of all possible edge platforms.

6.2 Overall Behavior of *avg_frontier*

The results show that *avg_frontier* provides a controllable shift from time-oriented to energy-oriented operating modes as α increases from 0.0 to 1.0, as visualized in Fig. 3. On the small testbed, its makespan increases from 438.40 s to 1243.39 s, while active energy decreases from 64.02 kJ to 39.35 kJ. On the large testbed, the same change in α moves the method from a time-oriented point with makespan 1931.43 s and active energy 23.99 MJ to an energy-oriented point with makespan 40328.86 s and active energy 4.06 MJ.

This behavior confirms the intended role of α: small values select operating points closer to the time-oriented region, whereas large values shift the allocation toward energy-saving devices. The transition is not meant to produce the best value of every metric simultaneously. Rather, it provides a single parameter for navigating the time–energy trade-off.

The scheduler overhead also remains substantially lower than that of population-based Pareto baselines. Although the absolute wall-clock time of *avg_frontier* increases on the 100-node testbed, it stays in the range of lightweight batch heuristics and remains far below the evolutionary methods, especially under repeated scheduling.

6.3 Comparison with Classical Baselines

On the small testbed, the time-oriented setting of *avg_frontier* is stronger than the classical batch heuristics in terms of makespan. At $\alpha = 0.0$, it achieves a makespan of 438.40 s, compared with 456.14 s for `sufferage`, 456.63 s for `max_min`, and 495.61 s for `mct`. At the energy-oriented end, *avg_frontier* approaches the energy-focused heuristics: at $\alpha = 1.0$, its active energy is 39.35 kJ, while `minsuff`, `minmin_energy`, and `minmin_energy_time` obtain 37.71–40.56 kJ.

On the large testbed, the picture is more nuanced. Simple load-distribution heuristics such as `olb` and `mct` remain strong in the time-oriented region, whereas

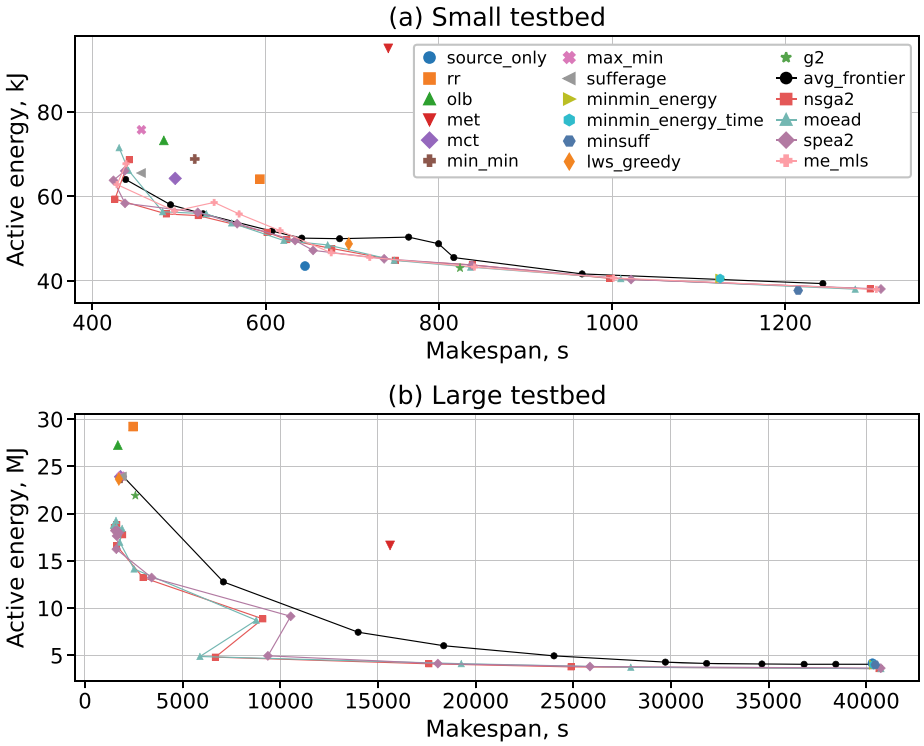

Fig. 3. Comparison of scheduling results in the makespan–active energy plane for the small and large testbeds. Lines correspond to methods producing multiple trade-off solutions, while isolated markers denote single-solution baselines. Energy is shown in kJ for the small testbed and in MJ for the large testbed.

explicitly energy-oriented heuristics obtain the lowest active energy among classical methods at the cost of very large makespan. The *avg_frontier* curve connects these regimes within one framework: as α grows, the method moves from a competitive time-oriented allocation toward energy-oriented behavior close to the energy-focused baselines.

Therefore, *avg_frontier* should be interpreted not as a method that simultaneously minimizes all metrics, but as a lightweight mechanism for controlled movement along the time–energy trade-off.

6.4 Comparison with Pareto-Based Baselines

The Pareto-based baselines also produce trade-off curves in the makespan–energy plane. In several regions, especially on the large testbed, they can find better schedule-quality points than *avg_frontier*. For example, in the time-oriented region of the large testbed, NSGA-II, MOEA/D, and SPEA2 reach

lower makespan values than *avg_frontier*. This is expected because these methods search directly in a richer assignment space.

However, the main practical difference is the cost of the decision procedure itself. Figure 4 summarizes this difference on both testbeds.

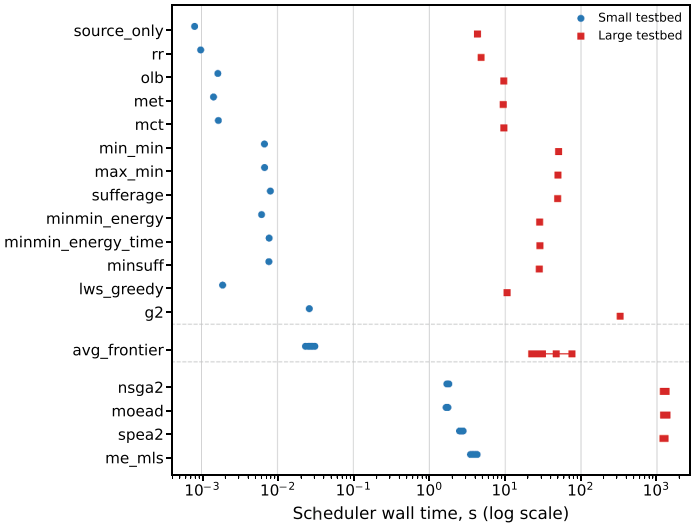

Fig. 4. Scheduler wall-clock time on the small and large testbeds. The horizontal axis uses a logarithmic scale. For methods producing several operating points, markers correspond to different α settings, while horizontal segments indicate the observed range of scheduler times.

On the small testbed, *avg_frontier* requires only a small fraction of the wall-clock time required by the population-based baselines. On the large testbed, this difference becomes much more pronounced: NSGA-II, MOEA/D, and SPEA2 require scheduler wall-clock times on the order of 10^3 seconds, whereas *avg_frontier* remains on the order of tens of seconds over all tested α values. ME-MLS was not included in the large-testbed comparison because it did not complete within practical time in preliminary runs.

Thus, the main advantage of *avg_frontier* is not unconditional dominance in the makespan–energy plane. Its advantage is the balance between schedule quality, controllability of the operating point, and decision cost. This balance is especially important when scheduling is invoked repeatedly and the scheduler itself becomes part of the system overhead.

7 Limitations

The present study has several limitations that should be taken into account when interpreting the results.

First, the evaluation is restricted to two simulated testbeds and one task family. The small testbed was designed for interpretability, whereas the large testbed was generated to study scalability and scheduler overhead. These configurations cover two different scales, but they still do not provide an exhaustive characterization over multiple workload classes, topology types, or network/compute ratios.

Second, the evaluation is simulation-based. This is appropriate for controlled comparative analysis, but practical deployment will additionally depend on aspects that are harder to capture in the present model, including operating-system overheads, measurement noise, and implementation details of the runtime layer.

Third, the surrogate frontier should not be interpreted as an exact approximation of the true global Pareto front of the original assignment problem. It is used as a compact decision mechanism whose selected allocations are then re-evaluated by exact simulation. The reported results therefore characterize the behavior of this mechanism rather than the full global Pareto structure of the scheduling problem.

These limitations do not invalidate the obtained observations, but they delineate the scope within which the conclusions of the paper should be understood.

8 Conclusion

This paper considered the problem of batch scheduling of independent tasks in a heterogeneous edge environment under conflicting time and energy requirements. A lightweight scheduler, *avg_frontier*, was studied. The method first solves an aggregated surrogate bi-objective allocation problem in the space of task counts assigned to devices, explicitly constructs the corresponding discrete Pareto frontier, selects one operating point by the *area_ratio* policy, restores the resulting item-level assignment, and then re-evaluates the obtained plan by exact simulation.

The experiments on the small and large heterogeneous testbeds lead to the following conclusions. First, the proposed scheduler provides a clear and controllable transition between more time-oriented and more energy-oriented operating modes through a single parameter α. Second, its scheduler overhead remains substantially lower than that of population-based Pareto baselines, and this difference becomes especially visible on the 100-node testbed. Third, although *avg_frontier* does not dominate all alternatives in the makespan–energy plane, it provides a favorable compromise between schedule quality, interpretability of the selected operating point, and decision cost.

At the same time, the present conclusions should be interpreted carefully. The evaluation remains limited to simulated configurations and one task family. Therefore, the main result of the paper is not a universal claim that *avg_frontier* dominates all alternatives, but the demonstration that this scheduler provides an interpretable and computationally cheap mechanism for navigating the time–energy trade-off under repeated scheduling.

A natural continuation of this work is associated with three directions: validation on additional platform and workload configurations, investigation of adaptive policies for selecting the target operating regime, and comparison of simulation-based predictions with execution traces from a physical prototype.

References

1. Braun, T.D., et al.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distrib. Comput. **61**(6), 810–837 (2001)
2. Brogi, A., Forti, S., Ibrahim, A.: How to best deploy your fog applications, probably. In: IEEE 1st International Conference on Fog and Edge Computing (ICFEC), pp. 105–114. IEEE (2017)
3. Casanova, H., Giersch, A., Legrand, A., Quinson, M., Suter, F.: Versatile, scalable, and accurate simulation of distributed applications and platforms. J. Parallel Distrib. Comput. **74**(10), 2899–2917 (2014)
4. Czarnul, P., Proficz, J., Krzywaniak, A.: Energy-aware high-performance computing: survey of state-of-the-art tools, techniques, and environments. Sci. Program. **2019**(1), 8348791 (2019)
5. De Maio, V., Brandic, I.: Multi-objective mobile edge provisioning in small cell clouds. In: Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering, pp. 127–138 (2019)
6. Deb, K.: Multi-objective optimisation using evolutionary algorithms: an introduction. In: Wang, L., Ng, A., Deb, K. (eds.) Multi-Objective Evolutionary Optimisation for Product Design and Manufacturing, pp. 3–34. Springer, London (2011). https://doi.org/10.1007/978-0-85729-652-8_1
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
8. Gupta, H., Vahid Dastjerdi, A., Ghosh, S.K., Buyya, R.: ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. Softw. Pract. Exp. **47**(9), 1275–1296 (2017)
9. Iturriaga, S., Nesmachnow, S., Bernabe, D., Bouvry, P.: Energy efficient scheduling in heterogeneous systems with a parallel multiobjective local search. Comput. Inform. **32**(2), 273–294 (2013)
10. Kocot, B., Czarnul, P., Proficz, J.: Energy-aware scheduling for high-performance computing systems: a survey. Energies **16**(2), 890 (2023)
11. Krzhizhanovskaya, V.V., Korkhov, V.V.: Dynamic load balancing of *black-Box* applications with a resource selection mechanism on heterogeneous resources of the grid. In: Malyshkin, V. (ed.) PaCT 2007. LNCS, vol. 4671, pp. 245–260. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73940-1_26
12. Kwok, Y.K., Ahmad, I.: Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Computi. Surv. (CSUR) **31**(4), 406–471 (1999)
13. Luo, P., Lü, K., Shi, Z.: A revisit of fast greedy heuristics for mapping a class of independent tasks onto heterogeneous computing systems. J. Parallel Distrib. Comput. **67**(6), 695–714 (2007)
14. Mach, P., Becvar, Z.: Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun. Surv. Tutor. **19**(3), 1628–1656 (2017)

15. Maheswaran, M., Ali, S., Siegel, H.J., Hensgen, D., Freund, R.F.: Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. J. Parallel Distrib. Comput. **59**(2), 107–131 (1999)
16. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on mobile edge computing: the communication perspective. IEEE Commun. Surv. Tutor. **19**(4), 2322–2358 (2017)
17. Nesmachnow, S., Dorronsoro, B., Pecero, J.E., Bouvry, P.: Energy-aware scheduling on multicore heterogeneous grid computing systems. J. Grid Comput. **11**(4), 653–680 (2013)
18. Sajid, M., Raza, Z., Shahid, M.: Energy-efficient scheduling algorithms for batch-of-tasks (bot) applications on heterogeneous computing systems. Concurr. Comput.: Pract. Exp. **28**(9), 2644–2669 (2016)
19. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)
20. Sonmez, C., Ozgovde, A., Ersoy, C.: Fuzzy workload orchestration for edge computing. IEEE Trans. Netw. Serv. Manag. **16**(2), 769–782 (2019)
21. Tarafdar, A., Debnath, M., Khatua, S., Das, R.K.: Energy and makespan aware scheduling of deadline sensitive tasks in the cloud environment. J. Grid Comput. **19**(2), 19 (2021)
22. Topcuoglu, H., Hariri, S., Wu, M.Y.: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. **13**(3), 260–274 (2002)
23. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **11**(6), 712–731 (2007)
24. Zhdanov, D., Korkhov, V.: On algorithms of task placement in computing clusters taking into account task and system topologies. Phys. Part. Nucl. **55**(3), 624–626 (2024)
25. Zhou, A., Qu, B.Y., Li, H., Zhao, S.Z., Suganthan, P.N., Zhang, Q.: Multiobjective evolutionary algorithms: a survey of the state of the art. Swarm Evol. Comput. **1**(1), 32–49 (2011)
26. Zitzler, E., Laumanns, M., Thiele, L.: Spea2: improving the strength pareto evolutionary algorithm. TIK Rep. **103** (2001)