

Degradation-Aware Single-Machine Scheduling with Price-Driven Thresholding and Waiting Triggers

Zhaoyi Zhang¹, Chumpol Yuangyai^{1,*}, Chen-Yang Cheng², Ranon Jientrakul¹
and Nagoor Basha Shaik¹

1 Department of Industrial Engineering, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, Lat Krabang, Lat Krabang, Bangkok, 10520, Thailand.

2 Department of Industrial Engineering and Management, National Taipei University of Technology, Minhui Village, Da'an, Taipei, 106344, Republic of China.

*Corresponding Author E-mail: chumpol.yu@kmitl.ac.th

Received: Apr 10, 2026; Revised: Jul 01, 2026; Accepted: Jul 07, 2026

Abstract

This paper studies a degradation-aware non-preemptive single-machine scheduling problem under time-of-use (TOU) and flat (FLAT) electricity tariffs. The objective is to minimize total cost, including processing energy, optional waiting energy, preventive maintenance (PM), and weighted tardiness. The model links reliability degradation to processing power through an interpretable reliability-power mapping and uses an MTBF-calibrated exponential reliability function with PM restoration. We propose a price-driven adaptive reliability threshold $r_{th}(t)$, bounded by r_{min} and r_{max} , and a conservative waiting trigger that is activated only when due-date slack is sufficient and the expected price decrease is meaningful. Under FLAT pricing, the adaptive term vanishes and the policy degenerates to the static baseline, providing a direct consistency check. For computation, PM and waiting decisions are generated by policy rules, whereas metaheuristics optimize the job permutation. A permutation-based Tabu Search (TS) is compared with GA, SA, ACO, and DE-RK on OR-Library WT40/WT50/WT100 instances. The results show rational TOU responsiveness, small-instance optimality references, and non-additive interactions between threshold adaptiveness and waiting under strong due-date pressure. The framework provides auditable decision logic for scheduling and maintenance under time-varying electricity prices.

Keywords: Time-of-use Electricity Pricing, Degradation-aware Scheduling, Reliability Threshold Policy, Preventive Maintenance, Weighted Tardiness

1. Introduction

Time-varying electricity tariffs have turned energy consumption from a secondary overhead into a first-order operational concern in manufacturing. Under TOU pricing, the same operation can cost substantially more in peak windows than in valley periods, making production schedules economically time-sensitive rather than purely capacity-driven [1],[2]. At the same time, strict delivery requirements remain dominant in many workshops, where weighted tardiness penalties can outweigh energy savings[2]. This creates a direct trade-off: shifting processing to cheaper tariff intervals may reduce energy cost, but it can delay completion and increase tardiness [3],[4].

A second tension lies in the endogeneity of energy demand. Machines do not always consume a constant amount of power per unit processing time. During operation, equipment degradation can increase friction, heat, or control effort, so the power required to sustain throughput may rise as condition deteriorates [5],[6]. The marginal energy cost is therefore time-dependent through $h(t)$ and state-dependent through degradation. Preventive maintenance can restore the machine to an as-good-as-new state, reducing degradation-induced power increases and controlling reliability risk, but it also introduces downtime and direct maintenance cost [7].

Waiting before starting a job offers a further operational lever because it can bypass high-price windows without changing the job sequence [3],[8].

However, waiting consumes idle power and can increase tardiness if applied without regard to due-date slack. A useful scheduling policy should therefore explain when the system should trigger PM earlier, when it should process aggressively to protect due dates, and when short waiting is economically justified.

Motivated by these coupled trade-offs, this work develops a degradation-aware single-machine scheduling framework under TOU and FLAT tariffs. The proposed reliability-threshold policy is tariff-responsive, bounded, and auditable. It degenerates to a static baseline under FLAT pricing and responds transparently to TOU signals when price volatility is present [2–4].

The framework is intended for production settings in which a single bottleneck machine, such as a CNC machining center, heat-treatment furnace, precision press, coating line, or high-load testing station, faces both time-varying electricity prices and delivery penalties. In such environments, condition-dependent power demand and PM timing affect not only energy cost but also order completion. The proposed policy can therefore be embedded in shop-floor decision support systems or digital-twin scheduling modules where tariff signals, due dates, and machine-condition indicators are available.

2. Literature Review and Research Gap

Energy-aware scheduling under TOU tariffs has mainly modeled electricity cost by combining a tariff profile with fixed or exogenous processing power [1],

[2],[4]. Related TOU-driven single-machine and batch-scheduling studies address release dates, machine on/off switching, parallel-batch processing, or group-processing features [9–11]. These studies clarify how electricity prices affect sequencing and timing, but they do not simultaneously include reliability-dependent power, PM restoration, conservative waiting, weighted tardiness, and an interpretable price-driven threshold.

Maintenance-integrated scheduling has studied PM insertion, remaining-useful-life constraints, and on/off control during maintenance-related decisions [7],[12]. These studies show the value of coordinating production and maintenance, but maintenance rules are often static, periodic, or condition-based without an explicit link to electricity-price signals.

Idle or waiting decisions provide another route for tariff exploitation. Prior single-machine work on

dynamic electricity pricing has shown that controlled idle time can reduce energy cost [3], while flexible job-shop work has considered idle arrangement and machine-speed choices under disruptions [8]. The benefit of waiting is nevertheless strongly affected by due-date pressure. This motivates a conservative waiting trigger that requires both sufficient slack and a meaningful expected price drop.

Algorithmic studies for energy-aware production scheduling commonly use metaheuristics for large instances [13–17]. Because the present objective combines time-dependent prices, state-dependent power, PM restoration, waiting, and weighted tardiness, validation should report both algorithmic performance and mechanism-level behavior.

The comparative positioning of the related studies and the proposed framework is summarized in **Table 1**.

Table 1 Comparative positioning of related scheduling studies and the proposed framework

Study	Pricing	Deg.-power	PM	Wait/idle	Tardiness	Threshold rule
Wu et al. [2]	TOU	No	No	Release-date timing	Objective	No
Lee et al. [3]	Dynamic price	No	No	Idle control	Due-date related	No
Park and Ham [4]	TOU	Mode/shop energy	No	Not central	Shop objective	No
Cheng et al. [9]	TOU	No	No; on/off only	Machine on/off	No	No
Tian and Zheng [10]	TOU	No	No	Batch timing	No	No
Yuan et al. [11]	TOU	No	No	Group timing	No	No
Gong et al. [12]	Energy-aware	No	Maint./on-off	Machine on/off	Not central	No price-driven rule
He et al. [7]	No	RUL focus	PM integrated	No	Constraints	No
Duan and Wang [8]	Energy-aware	No explicit map	Breakdown-related	Idle arrangement	Yes	No
This study	TOU and FLAT	Reliability-power map	As-good-as-new	Conservative trigger	Yes	Price-driven

Table 1 shows that the incremental contribution is not any single component in isolation. The contribution lies in coupling TOU pricing, reliability-dependent processing power, PM restoration, optional waiting, weighted tardiness, and an interpretable price-driven threshold within one auditable evaluation framework.

Based on this gap, this paper develops an interpretable framework that couples TOU/FLAT tariffs, reliability-driven degradation, PM restoration, and due-date performance. The problem contains n non-preemptive jobs available at $t=0$. Each job i has processing time p_i , due date d_i , and weight w_i . The machine degrades during processing, and reliability $R(t)$ is mapped from accumulated processing lifetime since the most recent PM. PM lasts T_{pm} , costs C_{pm} , and restores the machine to an as-good-as-new condition. Electricity price follows a 24-hour periodic function under TOU and is constant under FLAT pricing.

The objective is to minimize total cost, including processing energy, optional waiting energy, maintenance cost, and weighted tardiness. The technical challenge is that energy cost depends on start time through $h(t)$, processing power depends on reliability, and PM and waiting decisions change completion times. Even for a

fixed permutation, accurate evaluation requires event-based accounting of tariff bands, reliability evolution, PM restoration, and idle windows.

We model degradation with an exponential reliability function parameterized by MTBF and map reliability to processing power using an interpretable piecewise function. PM restores reliability to its initial condition, forming a closed-loop degradation-restoration mechanism that drives energy, maintenance, and delivery trade-offs.

The key policy innovation is a price-driven adaptive reliability threshold $r_{th}(t)$, bounded by r_{min} and r_{max} . When prices are high, the threshold increases, making the policy more conservative and encouraging earlier PM to reduce low-reliability processing under expensive electricity. When prices are low, the threshold relaxes to mitigate due-date pressure.

We further incorporate an optional waiting policy to bypass expensive price windows. Waiting is activated only when due-date slack is sufficient and the expected price decrease exceeds the present threshold, subject to a maximum waiting cap. This design avoids indiscriminate idling and makes the energy-tardiness trade-off explicit.

For computational tractability, we present a

conceptual mixed-integer model but implement a policy-induced evaluator. In this evaluator, PM and waiting decisions are generated by threshold and trigger rules, while metaheuristics optimize only the job permutation. TS is developed as the primary solver and benchmarked against GA, SA, ACO, and DE-RK under four policy scenarios.

The main contributions of this paper are as follows:

First, we propose a tariff-responsive reliability-threshold policy that degenerates to a static baseline under FLAT pricing and responds rationally under TOU tariffs.

Second, we build a unified degradation-energy-maintenance-tardiness framework that connects reliability-driven degradation with a piecewise reliability-power mapping and PM restoration.

Third, we implement policy-restricted optimization with a structured validation protocol, including degeneration consistency, enumeration-based optimality references, multi-algorithm benchmarking, mechanism evidence, and sensitivity analysis on OR-Library WT40/WT50/WT100.

The remainder of this paper is organized as follows. Section 3 details the model, adaptive threshold and waiting policies, mathematical formulation, and Tabu Search design. Section 4 presents experimental protocol, optimality verification, benchmark-scale comparison, and mechanism/sensitivity analyses. Section 5 concludes the paper and discusses practical implications and future research directions.

3. Proposed Methodology

We study a single-machine scheduling problem with n jobs $N = \{1, \dots, n\}$. Each job i has processing time p_i , due date d_i , and weight w_i . The machine degrades during processing, and the resulting reliability $R(t)$

affects processing power and energy cost. Under TOU or FLAT tariffs, the scheduler trades off processing energy, waiting energy, PM cost, and weighted tardiness.

3.1 Assumptions and time convention

The problem is first described through a conceptual mixed-integer event-based formulation. In computation, a policy-induced evaluator is used, where PM and waiting decisions are generated by the proposed threshold and trigger rules. We make the following assumptions:

- Single machine, non-preemptive processing;
- All jobs available at $t=0$;
- Degradation is driven by accumulated processing time only (no degradation during idle);
- PM can be inserted between consecutive jobs, lasts T_{pm} , costs C_{pm} , and restores the machine to as-good-as-new;
- Energy is charged according to the tariff over time (event/segment-based accounting). The notation is summarized in **Table 2**.

The no-idle-degradation assumption is a modeling boundary rather than a universal claim. It is appropriate for machines whose dominant wear mechanism is load-dependent processing, such as cutting, pressing, heating, or high-torque operation. For machines in which standby time, thermal cycling, startup, shutdown, or auxiliary systems also contribute materially to degradation, the lifetime update can be extended by adding idle-state or transition-state degradation terms. The present paper keeps degradation processing-driven to isolate the interaction among TOU prices, reliability-dependent processing power, PM timing, and weighted tardiness.

Table 2 Notation and Parameters

N, n	Set of jobs and number of jobs, $N = \{1, 2, \dots, n\}$
i, k, t	Job index, sequence-position index, and continuous time
p_i, d_i, w_i	Processing time, due date, and tardiness weight of job i
x_{ik}	Binary assignment variable equal to 1 if job i is processed at position k
M	Sufficiently large positive constant for Big-M linking constraints
S_k, Z_k	Start and completion time of the job processed at position k
C_i, T_i	Completion time and tardiness of job i
c_{tard}	Unit weighted-tardiness penalty coefficient
L_k, R_k	Accumulated processing lifetime and reliability before position k
$\lambda, MTBF$	Degradation rate and mean time between failures
z_k	Binary PM indicator before position k
T_{PM}, C_{PM}	PM duration and PM cost
$P_{nom}, P_{inc}, P_{idle}$	Nominal processing power, degradation-induced incremental power, and idle power
r_1^E, r_2^E	Reliability breakpoints in the processing-power mapping
$h(t), \bar{h}$	Electricity price at time t and its horizon average
$\Delta h(t)$	Deviation of the current electricity price from the average price
r_0, r_{min}, r_{max}	Baseline, lower-bound, and upper-bound reliability thresholds
α_{price}	Price-sensitivity coefficient in the adaptive threshold
$r_{th}(t)$	Price-driven adaptive reliability threshold
$Clip(\cdot)$	Truncation operator enforcing the lower and upper threshold bounds

Table 2 Notation and Parameters (cont.)

u_k	Waiting duration before position k
$\text{slack}_i(t)$	Waiting-trigger slack of job i at time t, defined as $\text{slack}_i(t) = d_i - \hat{C}_i^{\text{no wait}}(t)$
$W_{\max}, \Delta h_{\min}$	Maximum waiting duration and minimum price decrease required to wait
$C_E^{\text{proc}}, C_E^{\text{idle}}$	Processing-energy and idle-energy cost components
C_M, C_T	Maintenance and weighted-tardiness cost components
$F(\pi)$	Total cost of permutation

A conversion factor η may be used to unify time units into hours if needed.

For clarity, the main decision variables are x_{ik} (job-position assignment), z_k (PM before position k), u_k (waiting duration before position k), S_k and Z_k (start and completion time of position k), C_i and T_i (completion time and tardiness of job i), L_k (accumulated processing lifetime), R_k (reliability), and the cost-accounting variables for processing energy, idle energy, maintenance, and tardiness.

3.2 Degradation modeling via reliability

In reliability engineering, under a (locally) constant hazard-rate assumption, lifetime follows an exponential distribution and the survival function (reliability) decays exponentially. This form is monotone, easy to calibrate via MTBF, and naturally coupled with PM restoration. Hence, we map accumulated processing lifetime since the last PM to reliability.

Let L_k denote the accumulated processing lifetime (in hours) before processing the job in position k, measured since the most recent PM. Reliability is defined as

$$R_k = \exp(-\lambda L_k), \forall k = 1, \dots, n \quad (1)$$

The degradation rate is calibrated by MTBF:

$$\lambda = \frac{1}{\text{MTBF}} \quad (2)$$

In this study, λ is used as a deterministic degradation-rate parameter mapping accumulated processing lifetime L_k to expected reliability R_k , rather than as a stochastic Good/Fail transition probability.

Eq. (1) enforces a monotone decrease of reliability with accumulated processing. Eq. (2) provides an engineering-meaningful parameterization, because λ is directly determined by MTBF [6].

In this deterministic scheduling model, R_k is used as an expected condition index derived from the survival function, not as a binary stochastic state. A stochastic good/fail state model would require transition probabilities and failure-state recovery dynamics. The present formulation instead uses the MTBF-calibrated reliability trajectory to determine when PM should restore the machine and how degradation affects processing power.

If PM is performed before a position, the machine is restored to an as-good-as-new condition, as defined by Eq. (3):

$$z_k = 1 \Rightarrow L_k = 0, R_k = 1 \quad (3)$$

This action incurs a maintenance duration, T_{pm} , and maintenance cost, C_{pm} .

Here $L_k \geq 0$ denotes the accumulated effective processing lifetime since the most recent PM, and R_k in $(0,1]$ denotes the corresponding reliability level before the job at position k is processed. For example, in a CNC spindle or heat-treatment furnace, L_k represents the load-bearing operating time accumulated after the last maintenance action. Performing PM resets L_k to 0 and R_k to 1, which represents the as-good-as-new condition used in the evaluator.

As the machine degrades, friction increases and mechanical efficiency decreases, which raises the power demand per unit processing time. To retain interpretability while keeping the model computationally manageable, we adopt a piecewise-linear power mapping: processing uses a nominal power at high reliability; power increases linearly in a degradation region; and very low reliability is treated as infeasible, with PM triggered to avoid excessive failure risk.

Let P_{nom} be the nominal processing power, P_{inc} the maximum incremental power caused by degradation, and r_1^E and r_2^E the reliability breakpoints in the energy model, with $r_1^E > r_2^E$. The processing power is defined by the following piecewise function:

$$P^{\text{proc}}(R) = \begin{cases} P_{\text{nom}}, & R \geq r_1^E, \\ P_{\text{nom}} + P_{\text{inc}} \cdot \frac{r_1^E - R}{r_1^E - r_2^E}, & r_2^E < R < r_1^E, \\ \text{infeasible}, & R \leq r_2^E. \end{cases} \quad (4)$$

Eq. (4) maps the machine reliability level to the corresponding processing power. The linear segments in the reliability-power mapping are engineering approximations rather than statistical estimates. The coefficients express a monotone condition-energy relationship: high reliability uses nominal processing power, moderate degradation increases power gradually, and very low reliability is treated as operationally infeasible. This structure keeps the model interpretable and allows the sensitivity analysis to examine how conclusions change when the degradation and power parameters are perturbed.

Electricity price is modeled as a time function $h(t)$ with a 24-hour cycle:

$$h(t + 24) = h(t) \quad (5)$$

Eq. (5) defines the 24-hour periodic electricity-price function used in the tariff model. Under TOU, $h(t)$ follows peak/flat/valley (and optional super-peak) intervals; under FLAT, $h(t) \equiv h_{\text{flat}}$ for all t .

When the electricity price increases, the marginal energy cost rate $h(t) \times P^{\text{proc}}(R)$ rises. Eq. (6) defines the price-deviation term used by the adaptive rule, and Eq. (7) defines the bounded reliability threshold applied in expensive and cheaper periods.

$$\Delta h(t) = h(t) - \bar{h} \quad (6)$$

$$r_{th}(t) = \text{Clip}(r_0 + \alpha_{\text{price}} \Delta h(t); [r_{\min}, r_{\max}]) \quad (7)$$

In the policy-induced evaluator, PM before position k is triggered when $R_k \leq r_{th}(S_k)$, and it is forcibly triggered when $R_k \leq r_2^E$. In either case, $z_k = 1$; otherwise, $z_k = 0$. Therefore, z_k is generated by this threshold rule rather than freely optimized in the implemented metaheuristics.

$$\text{Clip}(x; [r_{\min}, r_{\max}]) = \min\{\max(x, r_{\min}), r_{\max}\} \quad (8)$$

Eq. (8) defines the clipping operator that enforces the lower and upper threshold bounds. With idle waiting enabled, the system may wait before starting a job to bypass expensive price windows, at the risk of increasing completion times and tardiness. We thus adopt a conservative trigger: waiting is allowed only when slack is sufficient, and the price drop is meaningful, under a maximum waiting cap:

$$0 \leq u_k \leq W_{\max} \quad (9)$$

Eq. (9) imposes the maximum waiting cap for each waiting duration. Before defining the waiting trigger, we define slack as follows: $\text{slack}_i(t) = d_i - \hat{C}_i^{\text{no wait}}(t)$

where $\hat{C}_i^{\text{no wait}}(t)$ denotes the estimated completion time if job i starts immediately at time t . The waiting trigger is then expressed as:

$$u_k \leq \text{slack}_i(t) \text{ and } h(t) - h(t + u_k) \geq \Delta h_{\min} \quad (10)$$

The evaluator scans a waiting duration $u \in [0, W_{\max}]$ with a fixed step and selects the earliest feasible value for which the slack and price-drop conditions in Eq. (10) both hold. If no such value exists, then $u_k = 0$. Thus, u_k is generated by the waiting trigger rather than freely optimized in the implemented metaheuristics.

3.3 Mathematical Formulation

The problem is formulated as:

$$\min F = C_E^{\text{proc}} + C_E^{\text{idle}} + C_M + C_T \quad (11)$$

$$C_E^{\text{proc}} = \sum_{k=1}^n \int_{S_k}^{Z_k} h(t) P_{\text{proc}}(R_k) dt \quad (12)$$

$$\min F = C_E^{\text{proc}} + C_E^{\text{idle}} + C_M + C_T \quad (11)$$

$$C_E^{\text{idle}} = \sum_{k=1}^n \int_{S_k - u_k}^{S_k} h(t) P_{\text{idle}} dt \quad (13)$$

$$C_M = \sum_{k=1}^n C_{PM} \cdot z_k \quad (14)$$

$$C_T = c_{\text{tard}} \sum_{i \in N} w_i \cdot T_i \quad (15)$$

Eq. (11) defines the total objective, whereas Eqs. (12)–(15) specify the processing-energy, idle-energy, maintenance-cost, and weighted-tardiness components, respectively.

s.t.

$$\sum_{i \in N} x_{ik} = 1, \forall k = 1, \dots, n \quad (16)$$

$$\sum_{k=1}^n x_{ik} = 1, \forall i \in N \quad (17)$$

$$S_1 \geq u_1 + T_{PM} z_1 \quad (18)$$

$$S_k \geq Z_{k-1} + u_k + T_{PM} z_k, k = 2, \dots, n \quad (19)$$

$$Z_k = S_k + \sum_i p_i x_{ik}, \forall k \quad (20)$$

$$C_i \geq Z_k - M(1 - x_{ik}), \forall i, k \quad (21)$$

$$C_i \leq Z_k + M(1 - x_{ik}), \forall i, k \quad (22)$$

$$T_i \geq C_i - d_i, \forall i \quad (23)$$

$$T_i \geq 0, \forall i \quad (24)$$

$$L_1 = 0 \quad (25)$$

$$L_{k+1} \geq L_k + \sum_i p_i x_{ik} - M z_{k+1}, k = 1, \dots, n-1 \quad (26)$$

$$L_{k+1} \leq L_k + \sum_i p_i x_{ik} + M z_{k+1}, k = 1, \dots, n-1 \quad (27)$$

$$L_{k+1} \leq M(1 - z_{k+1}), k = 1, \dots, n-1 \quad (28)$$

$$R_k = \exp(-\lambda L_k), \forall k \quad (29)$$

$$x_{ik}, z_k \in \{0, 1\}; S_k, Z_k, C_i, T_i, L_k, R_k, u_k \geq 0 \quad (30)$$

Constraints in Eqs. (16)–(17) form the permutation assignment. Constraints in Eqs. (18)–(20) propagate start and completion times. Constraints in Eqs. (21)–(22) link job completion times to position completion times. Constraints in Eqs. (23)–(24) define tardiness. Constraints in Eqs. (25)–(29) model lifetime accumulation, PM reset, and reliability update. Constraint in Eq. (30) defines the variable domains.

The formulation above describes the full conceptual optimization structure. Computational experiments use a policy-induced version of this structure. In that version, PM and waiting are not free binary decisions optimized jointly with the permutation; they are generated by the adaptive threshold and waiting trigger. This distinction explains why the implemented metaheuristics optimize only the permutation while still evaluating all four cost components.

3.4 Proposed Algorithm Design

We solve the considered single-machine scheduling problem by a permutation-based Tabu Search (TS). A candidate solution is encoded as a job permutation

$$\pi = (\pi_1, \pi_2, \dots, \pi_n) \quad (31)$$

In Eq. (31), jobs are processed sequentially in the listed order. Here, π denotes a job permutation, whereas π_i denotes the processing time of job i . For any permutation π , the evaluator reproduces the operational logic defined in the model.

The optimization goal is to minimize the total cost of a permutation π :

$$F(\pi) = C_E^{proc}(\pi) + C_E^{idle}(\pi) + C_M(\pi) + C_T(\pi) \quad (32)$$

The four terms in Eq. (32) aggregate processing energy cost, idle energy cost, preventive-maintenance cost, and weighted tardiness under the specified tariff and policy. For algorithms written in a unified maximize-fitness form, fitness is defined as the negative total cost in Eq. (33).

$$fitness(\pi) = -F(\pi) \quad (33)$$

Starting from a current permutation, TS samples swap-neighborhood candidates. Each swap exchanges two positions and evaluates the resulting permutation through the same policy-induced evaluator used for all algorithms. A tabuUntil matrix records the iteration until which each swap move is forbidden. A tabu move is accepted only when the aspiration criterion is satisfied, namely when it improves the best-so-far cost.

All five algorithms use the same permutation representation and the same policy-induced evaluator; they differ only in their search mechanisms.

Algorithm 1 summarizes the permutation-based TS procedure used in this study.

For clarity, the principal search mechanisms of the five algorithms under the common policy-induced evaluator are compared in **Table 3**.

Algorithm 1 Pseudocode of the proposed TS

Step	Operation	Description
1	Initialize	Generate an initial permutation π , evaluate $F(\pi)$, set best π^* , and initialize $\text{tabuUntil}(a,b)=0$.
2	Sample neighborhood	At each iteration, sample at most N_{neigh} swap moves (a,b) .
3	Evaluate candidates	For each candidate π' , compute $F(\pi')$ with the policy-induced evaluator.
4	Check tabu status	A move is tabu if $\text{iter} < \text{tabuUntil}(a,b)$ or $\text{iter} < \text{tabuUntil}(b,a)$.
5	Apply aspiration	A tabu move is admissible only when it improves the best-so-far cost $F(\pi^*)$.
6	Move and update	Select the best admissible candidate, update the incumbent, and set $\text{tabuUntil}(a,b)=\text{iter}+\text{tabuTenure}$ symmetrically.
7	Return	After maxIter iterations, return π^* , $F(\pi^*)$, and the recorded convergence history.

Table 3 Algorithm comparison under the common policy-induced evaluator

Algorithm	Representation	Initialization	Main search operator	Main parameters	Evaluation model
TS	Permutation	Initial permutation	Swap-neighborhood tabu search	$\text{tabuTenure}=10$; $N_{\text{neigh}}=40$	Same policy-induced evaluator
GA	Permutation	Random population	Selection, crossover, swap mutation	$\text{popSize}=40$; $\text{pc}=0.9$; $\text{pm_swap}=0.2$	Same policy-induced evaluator
SA	Permutation	Initial permutation	Swap move with annealing acceptance	$T_0=1$; $T_{\text{end}}=0.001$; $\text{cool}=0.97$	Same policy-induced evaluator
ACO	Permutation	Pheromone-guided construction	Probabilistic construction and pheromone update	$\text{ants}=15$; $\text{rho}=0.2$; $\text{patience}=20$	Same policy-induced evaluator
DE-RK	Permutation from random keys	Random-key population	Differential mutation and crossover, then decoding	$\text{popSize}=40$; $F=0.7$; $\text{CR}=0.9$	Same policy-induced evaluator

4. Results and Discussion

This section evaluates the proposed framework in four stages: model consistency, optimality and algorithmic effectiveness, mechanism analysis, and robustness assessment. We first validate the legitimacy of the proposed price-driven adaptive reliability-threshold mechanism by contrasting FLAT (low price volatility) and TOU tariffs, confirming (i) degeneration consistency under FLAT and (ii) rational responsiveness under TOU. Subsequent sections establish global optimality references via small-scale enumeration, benchmark performance on

medium-to-large instances, and provide ablation, mechanism curves, and sensitivity analyses to interpret how dynamic thresholds and waiting (idle) policies work in practice, concluding with actionable managerial implications. This validation is not intended to claim universal improvement on every single instance. Rather, it validates the consistency of degeneration under FLAT, interpretable responsiveness under TOU, and the resulting structural trade-offs; statistical effectiveness and generalizability are established by subsequent benchmark-scale evaluations and robustness analyses.

4.1 Experimental Setup and Reproducibility Protocol

To ensure fairness and reproducibility, we adopt the OR-Library weighted tardiness benchmark family (WT40, WT50, and WT100) and evaluate Instances 1–10 for each dataset. Each instance is solved with 10 independent runs (Seeds 1–10) to mitigate randomness. We compare Tabu Search (TS), Genetic Algorithm (GA), Simulated Annealing (SA), Ant

Colony Optimization (ACO), and a Differential-Evolution variant (DE-RK) under four scenario combinations: Static/Dynamic $\times$ NoIdle/Idle. Static uses a fixed reliability threshold r_0 , whereas Dynamic applies a price-driven adaptive threshold $r_{th}(t)$; NoIdle prohibits waiting and Idle allows waiting to bypass high-price periods when triggers are satisfied. The complete protocol is summarized in **Table 4**.

Table 4 Experiment Parameter Settings

Item	Setting
Hardware	CPU: [Intel Core i7-7500]
Software	MATLAB: [2024a]
Datasets	OR-Library weighted tardiness: WT40 / WT50 / WT100
Instances	Instance 1–10 for each dataset
Independent runs	10 independent runs (Seeds 1–10)
Compared algorithms	TS, GA, SA, ACO, DE-RK
Scenarios	Static/Dynamic $\times$ NoIdle/Idle (4 scenarios)
Termination conditions	GA: generation limit gens=120; TS/SA/ACO/DE-RK: iteration limit maxIter=120 (SA may terminate upon reaching Tend; runtime is recorded for each run.)
Key hyperparameters	GA: popSize=40, pc=0.9, pm_swap=0.2; TS: tabuTenure=10, N_neigh=40; SA: T0=1, Tend=0.001, cool=0.97; ACO: ants=15, rho=0.2, patience=20; DE-RK: popSize=40, F=0.7, CR=0.9
Output fields	Total, Energy, Maint, Tard, EnergyProc, EnergyIdle, PM, IdleHours, mean threshold, mean processing price, runtime in seconds, best permutation

The tabu tenure was set to 10 based on preliminary tuning on representative WT40 instances, balancing cycling prevention and search diversification. Since the aim of this study is comparative policy evaluation rather than exhaustive algorithm tuning, all algorithm parameters were kept fixed across datasets.

Beyond the experimental workflow, reproducibility also depends on consistent model parameterization. **Table 5** reports the degradation, power, maintenance, tardiness, tariff, adaptive-threshold, and waiting parameters used in all algorithmic comparisons. These parameters are fixed across algorithms and scenarios unless explicitly stated. Because iteration semantics differ across metaheuristics, we report both solution quality and runtime in seconds (T_{run}) rather than relying only on nominal iteration limits.

To clarify computational effort, all algorithms use the same objective evaluator but do not perform the same number of objective-function evaluations per nominal iteration. With the parameters in **Table 4**, TS evaluates approximately $1 + \maxIter \times N_{neigh} = 4,801$ permutations per run; GA and DE-RK evaluate approximately $popSize + gens \times popSize = 4,840$ permutations per run; ACO evaluates approximately $\maxIter \times ants = 1,800$ constructed permutations per run; and SA evaluates approximately $1 + \maxIter = 121$ candidate permutations, unless early stopping occurs. For this reason, the comparison reports both solution quality and runtime in seconds (T_{run}) rather than relying only on nominal iteration limits.

Table 5 Cost and model parameters

Category	Parameter	Value / Definition
Reliability	MTBF	3000 h
Reliability	$\lambda = 1/MTBF$	$1/3000 \text{ h}^{-1}$
Power model	P_{nom}	10 kW
Power model	P_{inc}	6 kW
Power model	P_{idle}	2 kW
Power model	r_1^E	0.90
Power model	r_2^E	0.60
Maintenance	T_{PM}	1.0 h
Maintenance	C_{PM}	500 per PM
Tardiness	c_{tard}	1.0
Tariff	TOU price levels	super-peak 1.0337; peak 0.9440; flat 0.6346; valley 0.3342
Tariff	TOU intervals	peak [10,15],[18,21]; flat [7,10],[15,18],[21,23]; valley [23,24],[0,7]
Adaptive threshold	r_{min}, r_{max}	0.65, 0.98

Table 5 Cost and model parameters (cont.)

Category	Parameter	Value / Definition
Adaptive threshold	r_0	0.85
Adaptive threshold	α_{price}	0.10
Adaptive threshold	$r_{\text{th}}(t) = \text{Clip}(r_0 + \alpha_{\text{price}}\Delta h(t); [r_{\min}, r_{\max}])$	price-driven bounded threshold
Waiting	$W_{\max}$	3.0 h
Waiting	$\Delta h_{\min}$	0.10

Note: $\bar{h}$ denotes the average electricity price over the planning horizon. $\text{Clip}(x, [r_{\min}, r_{\max}]) = \min\{\max[x, r_{\min}], r_{\max}\}$. This function truncates its argument into $[r_{\min}, r_{\max}]$. The monotonic direction is set such that higher prices yield a more conservative reliability threshold within the prescribed bounds.

4.2 Model Legitimacy Verification

This section validates legitimacy from two perspectives: degeneration consistency and rational responsiveness. First, under FLAT pricing (nearly no price volatility), the price-driven adaptive threshold should degenerate to the fixed baseline and the waiting policy should not create artificial gains. Second, under TOU pricing, the adaptive threshold and waiting policy should respond to the tariff signal in an interpretable way and induce rational cost-structure trade-offs. To isolate mechanism correctness from search-algorithm effects, we fix the job permutation and conduct an ablation study on a representative WT40 instance ($n=40$, $\text{Inst}=1$), comparing Static/Dynamic and NoIdle/Idle under FLAT and TOU tariffs in **Table 6**.

Under the FLAT tariff, **Table 6** shows that all four combinations (Static/Dynamic $\times$ NoIdle/Idle) yield identical total and component cost. This aligns with the adaptive rule: when $h(t) = \bar{h}$, $\Delta h(t) = 0$ and $r_{\text{th}}(t) = r_0$. The adaptive threshold therefore degenerates to the fixed

baseline under FLAT pricing, confirming consistency of degeneration.

Under the TOU tariff, to avoid confounding by waiting, we first compare NoIdle combinations. Relative to Static_NoIdle_TOU, Dynamic_NoIdle_TOU reduces energy cost (Energy: 11374.396 $\rightarrow$ 10635.515) while increasing maintenance triggers and cost (PM: 3 $\rightarrow$ 4; Maint: 1500 $\rightarrow$ 2000), with a slight change in tardiness (Tard: 16846 $\rightarrow$ 16893), resulting in a lower total cost (Total: 29720.396 $\rightarrow$ 29528.515). This indicates a structural trade-off among energy, maintenance, and delivery performance rather than a trivial single-term optimization (see **Table 6**). Moreover, in TOU scenarios with waiting enabled, positive IdleHours (7–8 hours) confirms that the waiting policy can bypass high-price windows, providing observable evidence for the subsequent mechanism figures linking $h(t)$ and $r_{\text{th}}(t)$. This legitimacy check is designed to validate degeneration consistency and interpretable responsiveness, rather than to claim guaranteed improvement in every instance.

Table 6. Ablation results for legitimacy verification (WT40, $n=40$, $\text{Inst}=1$)

Scenario	Tariff	Total	Energy	Maint	Tard	PM	IdleHours
Static_NoIdle_FLAT	FLAT	31661.877	13315.877	1500	16846	3	0
Dynamic_NoIdle_FLAT	FLAT	31661.877	13315.877	1500	16846	3	0
Static_Idle_FLAT	FLAT	31661.877	13315.877	1500	16846	3	0
Dynamic_Idle_FLAT	FLAT	31661.877	13315.877	1500	16846	3	0
Static_NoIdle_TOU	TOU	29720.396	11374.396	1500	16846	3	0
Dynamic_NoIdle_TOU	TOU	29528.515	10635.515	2000	16893	4	0
Static_Idle_TOU	TOU	30075.185	11265.185	1500	17310	3	8
Dynamic_Idle_TOU	TOU	30124.544	10825.544	2000	17299	4	7

To further validate the model's legitimacy and mechanism via visualization, we provide two complementary figures. **Figure 1** compares the cost composition under FLAT and TOU tariffs for fixed-threshold (Fixed) and adaptive-threshold (Adaptive) policies, decomposing total cost into Energy, Maintenance, and Tardiness, thereby visually examining the consistency of degeneration under FLAT and the structural changes under TOU. **Figure 2** plots the coupling between the electricity price signal $h(t)$ and the threshold trajectory $r_{\text{th}}(t)$, with the fixed baseline superimposed, to verify whether the adaptive threshold responds to tariff variations in the

intended direction and magnitude, thereby supporting the subsequent interpretation of the mechanism.

As shown in **Figure 1**, Fixed and Adaptive nearly overlap under the FLAT tariff, indicating that the electricity price is approximately constant. Under the TOU tariff, the cost structure changes in an interpretable manner, suggesting a rebalancing among energy, maintenance, and tardiness. **Figure 2** further shows that $r_{\text{th}}(t)$ co-varies with $h(t)$ while remaining bounded around r_0 and the prescribed limits, providing mechanistic support for the tariff-responsive threshold design.

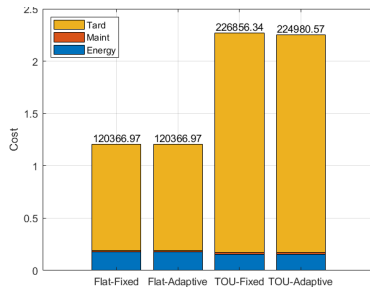


Figure 1 Cost composition comparison under Fixed vs Adaptive (FLAT vs TOU)

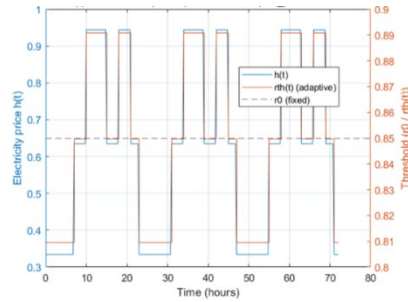


Figure 2 Electricity price $h(t)$ versus adaptive threshold $r_{th}(t)$.

4.3 Optimality Verification on Small Instances

To establish a global optimum reference on tractable problem sizes, we apply enumeration to small instances ($n = 5-8$). The small-instance enumeration results use the original cost scale of the extracted sub-instances and are intended only as optimality references for GAP computation. Therefore, their absolute total-cost scale should not be directly compared with the subsequent medium- and large-scale benchmark results.

At these small sizes, fixed and adaptive optima are close in most cases. This indicates that the small enumeration setting is useful primarily as an auditable optimality reference, while the larger benchmark experiments are needed to evaluate algorithmic effectiveness and policy behavior under stronger combinatorial pressure.

The fixed- and adaptive-policy enumeration results are summarized in **Table 7** and **Table 8**, respectively. The enumerated optima in these tables serve as references for computing GAP in the next section, enabling systematic evaluation of approximation quality and stability.

Table 7 Fixed-policy enumeration results on small WT40 instances (Static Idle TOU, means across Instances 1–10)

n	Total	Energy	Maint.	Tard.	PM	Idle hours
5	2663030.45	5501.85	200.00	2657328.60	0.40	0.40
6	5150994.81	7148.41	250.00	5143596.40	0.50	0.20
7	5546116.38	4892.08	300.00	5540924.30	0.60	0.90
8	7107611.87	9106.67	350.00	7098155.20	0.70	0.90

Table 8 Adaptive-policy enumeration results on small WT40 instances (Dynamic Idle TOU, means across Instances 1–10)

n	Total	Energy	Maint.	Tard.	PM	Idle hours	Relative improvement over fixed (%)
5	2663030.45	5501.85	200.00	2657328.60	0.40	0.40	0.000
6	5150994.81	7148.41	250.00	5143596.40	0.50	0.20	0.000
7	5546116.38	4892.08	300.00	5540924.30	0.60	0.90	0.000
8	7107660.52	9104.92	400.00	7098155.60	0.80	0.90	-0.001

Note: Fixed uses $r_0 = 0.85$ and $\alpha_{price} = 0$. Adaptive uses $r_0 = 0.85$ and $\alpha_{price} = 0.10$. The relative improvement is defined below. A negative value means that the adaptive policy is slightly worse than the fixed policy in that small-instance setting. PM and IdleHours are reported as averages over instances; Improvement (%): $(Total_{Fixed} - Total_{Adaptive}) / Total_{Fixed} * 100\%$

4.4 Algorithm Legitimacy Verification vs. Enumeration

To validate implementation correctness and approximation quality, we evaluate TS, GA, SA, ACO, and DE-RK on enumerable WT40 instances ($n = 5-8$). Enumerated global optima are used as references, and **Table 9** reports the mean and standard deviation of GAP(%) and runtime in seconds under Dynamic Idle TOU.

Table 9 Approximation quality against enumeration (mean $\pm$ std GAP)

n	Algorithm	GAP (%)	T_run (s)
5	TS	0.000 $\pm$ 0.000	0.300 $\pm$ 0.076
5	GA	0.023 $\pm$ 0.127	0.290 $\pm$ 0.065
5	SA	0.185 $\pm$ 0.312	0.039 $\pm$ 0.007
5	ACO	0.140 $\pm$ 0.767	0.138 $\pm$ 0.033

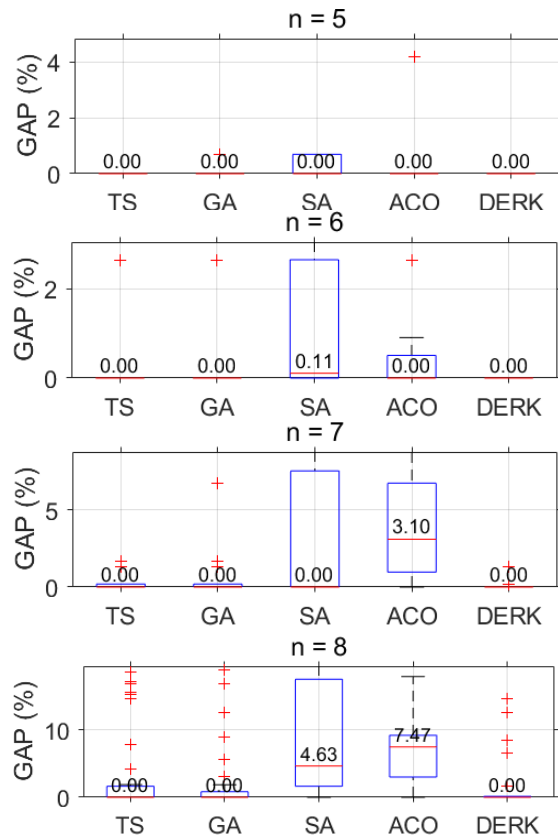
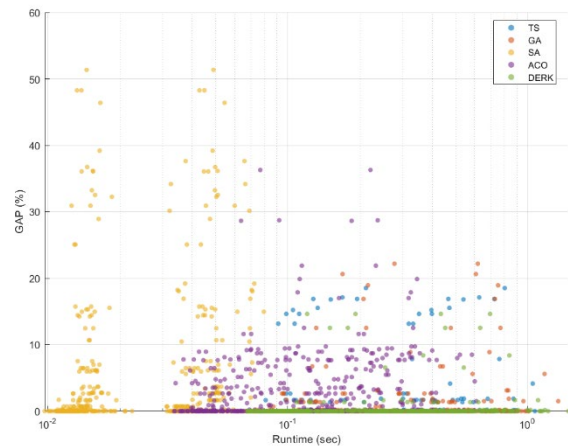
Table 9 Approximation quality against enumeration (mean $\pm$ std GAP) (cont.)

n	Algorithm	GAP (%)	T_run (s)
5	DE-RK	0.000 $\pm$ 0.000	0.293 $\pm$ 0.057
6	TS	0.088 $\pm$ 0.483	0.377 $\pm$ 0.101
6	GA	0.264 $\pm$ 0.807	0.415 $\pm$ 0.119
6	SA	2.481 $\pm$ 5.426	0.046 $\pm$ 0.009
6	ACO	0.701 $\pm$ 1.458	0.196 $\pm$ 0.067
6	DE-RK	0.000 $\pm$ 0.000	0.409 $\pm$ 0.104
7	TS	0.712 $\pm$ 2.421	0.424 $\pm$ 0.098
7	GA	0.435 $\pm$ 1.288	0.530 $\pm$ 0.152
7	SA	6.154 $\pm$ 12.307	0.051 $\pm$ 0.013
7	ACO	3.657 $\pm$ 3.195	0.249 $\pm$ 0.094
7	DE-RK	0.148 $\pm$ 0.410	0.515 $\pm$ 0.125

Table 9 Approximation quality against enumeration (mean $\pm$ std GAP) (cont.)

n	Algorithm	GAP (%)	T run (s)
8	TS	0.660 ± 1.649	0.536 ± 0.180
8	GA	1.562 ± 4.351	0.659 ± 0.195
8	SA	8.929 ± 10.303	0.056 ± 0.008
8	ACO	6.652 ± 4.862	0.305 ± 0.140
8	DE-RK	0.531 ± 1.902	0.684 ± 0.208

Table 9 shows how approximation quality and runtime change with n . GAP(%) is computed against the enumerated optimum for each instance and seed, so lower values indicate closer agreement with the global optimum. Runtime is reported in seconds to make the quality-time trade-off explicit. In contrast, SA achieves markedly shorter runtime but shows larger mean and variance of GAP, suggesting occasional convergence to suboptimal solutions even on small instances. ACO and GA lie between these extremes in terms of quality-time trade-off. These patterns are further visualized by the GAP boxplots (**Figure 3**) and the runtime-GAP scatter plot (**Figure 4**).

**Figure 3** Boxplot of GAP(%) to enumeration optimum for different algorithms**Figure 4** Runtime-GAP(%) scatter plot

4.5 Effectiveness on Medium/Large Instances

This section evaluates five metaheuristics on the OR-Library WT40/WT50/WT100 benchmarks under the main Dynamic_Idle_TOU setting. Results are paired by dataset, instance, and seed. **Table 10** reports total cost and runtime in seconds, **Table 11** reports cost decomposition and behavioral statistics, and Tables 12 and 13 report non-parametric statistical tests on the paired seed-level totals.

Because the algorithms have different update mechanisms and do not use exactly identical objective-evaluation budgets, the benchmark comparison should be interpreted as a quality-runtime comparison under practical parameter settings rather than a strictly evaluation-budget-matched algorithmic dominance test.

Table 10 shows that TS achieves the lowest mean total cost across WT40, WT50, and WT100. SA is faster but has substantially higher total cost, while GA, ACO, and DE-RK occupy intermediate quality-time regimes.

Because stochastic metaheuristic outputs need not satisfy normality assumptions, we conducted paired non-parametric tests on the instance-seed total costs. For each dataset, a Friedman test compared all five algorithms, followed by Holm-corrected Wilcoxon signed-rank tests comparing TS with GA, SA, ACO, and DE-RK. The results are reported in **Tables 12–13**.

Table 11 indicates that differences in total cost are mainly driven by weighted tardiness, while energy and maintenance provide secondary contributions. Together with the statistical results in **Tables 12–13**, these results support TS as the primary solver for the medium and large-scale benchmark evaluations.

Table 10 Large-scale performance summary under Dynamic Idle TOU (mean $\pm$ std)

Dataset	Algorithm	Total cost	T run (s)
WT40	TS	76236.38 ± 120980.48	5.561 ± 1.432
WT40	GA	167248.00 ± 291164.48	3.417 ± 1.366
WT40	SA	150053.48 ± 257029.79	0.170 ± 0.035
WT40	ACO	229895.50 ± 410038.10	0.879 ± 0.375
WT40	DE-RK	134084.06 ± 236766.39	4.063 ± 1.378

Table 10 Large-scale performance summary under Dynamic Idle TOU (mean $\pm$ std) (cont.)

Dataset	Algorithm	Total cost	T run (s)
WT50	TS	49655.98 $\pm$ 61700.35	7.636 $\pm$ 1.291
WT50	GA	146132.14 $\pm$ 238274.11	4.544 $\pm$ 1.857
WT50	SA	142434.16 $\pm$ 231334.71	0.208 $\pm$ 0.020
WT50	ACO	231416.74 $\pm$ 397346.20	1.084 $\pm$ 0.396
WT50	DE-RK	125163.33 $\pm$ 204618.69	5.121 $\pm$ 1.816
WT100	TS	158206.92 $\pm$ 286292.87	17.534 $\pm$ 1.544
WT100	GA	744256.71 $\pm$ 1918996.94	10.091 $\pm$ 4.095
WT100	SA	765615.70 $\pm$ 1963699.45	0.426 $\pm$ 0.042
WT100	ACO	1147646.48 $\pm$ 3037023.62	2.303 $\pm$ 0.941
WT100	DE-RK	689697.10 $\pm$ 1879297.25	13.736 $\pm$ 4.466

Table 11 Cost decomposition and behavioral statistics (means) under Dynamic Idle

Dataset	Algorithm	Energy	Maint.	Tard.	PM	Idle hours
WT40	TS	10,818.50	1,580.00	63,837.88	3.16	12.32
WT40	GA	13,283.41	1,775.00	152,189.59	3.55	14.39
WT40	SA	13,164.19	1,720.00	135,169.29	3.44	12.05
WT40	ACO	14,580.00	1,885.00	213,430.50	3.77	12.91
WT40	DE-RK	13,648.82	1,775.00	118,660.24	3.55	13.84
WT50	TS	11,949.65	1,925.00	35,781.33	3.85	18.26
WT50	GA	15,308.30	2,245.00	128,578.84	4.49	17.36
WT50	SA	14,987.88	2,255.00	125,191.28	4.51	17.68
WT50	ACO	16,979.12	2,420.00	212,017.62	4.84	17.59
WT50	DE-RK	15,850.27	2,325.00	106,988.06	4.65	18.31
WT100	TS	24,946.13	4,360.00	128,900.79	8.72	34.05
WT100	GA	33,569.85	5,025.00	705,661.86	10.05	35.49
WT100	SA	32,638.33	5,025.00	727,952.37	10.05	33.93
WT100	ACO	35,941.80	5,250.00	1,106,454.68	10.50	35.84
WT100	DE-RK	33,399.75	4,955.00	651,342.35	9.91	37.87

Table 12 Friedman test and average ranks under Dynamic Idle TOU

Dataset	Friedman Q	Friedman p-value	Average ranks (TS, GA, SA, ACO, DE-RK)
WT40	327.776	<1.0E-12	1.00, 3.38, 3.54, 4.80, 2.28
WT50	331.448	<1.0E-12	1.00, 3.15, 3.74, 4.80, 2.31
WT100	352.320	<1.0E-12	1.01, 3.30, 3.83, 4.79, 2.07

Table 13 Holm-corrected pairwise comparisons under Dynamic Idle TOU

Dataset	Pairwise comparison	Holm-adjusted p-value	Effect size	Direction
WT40	TS vs GA	<1.0E-12	1.000	TS lower total cost
WT40	TS vs SA	<1.0E-12	1.000	TS lower total cost
WT40	TS vs ACO	<1.0E-12	1.000	TS lower total cost
WT40	TS vs DE-RK	<1.0E-12	1.000	TS lower total cost
WT50	TS vs GA	<1.0E-12	1.000	TS lower total cost
WT50	TS vs SA	<1.0E-12	1.000	TS lower total cost
WT50	TS vs ACO	<1.0E-12	1.000	TS lower total cost
WT50	TS vs DE-RK	<1.0E-12	1.000	TS lower total cost
WT100	TS vs GA	<1.0E-12	1.000	TS lower total cost
WT100	TS vs SA	<1.0E-12	1.000	TS lower total cost
WT100	TS vs ACO	<1.0E-12	1.000	TS lower total cost
WT100	TS vs DE-RK	<1.0E-12	0.980	TS lower total cost

Each paired comparison uses 100 paired observations per dataset, corresponding to 10 instances $\times$ 10 seeds.

Because repeated seeds are stochastic replications on the same benchmark instances, these p-values are interpreted as paired run-level evidence rather than as

fully independent instance-level evidence.

Note: The effect size is the paired rank-biserial effect size. Each paired comparison uses paired instance-seed observations. A positive value indicates that TS obtains a lower total cost more frequently than

the comparator; ties are counted as zero contribution.

Considering **Tables 10–13**, TS consistently achieves the lowest mean total cost across all three datasets with acceptable runtime and competitive stability. SA is fastest but substantially costlier, making it more suitable as a quick baseline rather than the main optimizer.

To provide a multi-criteria comparison, we further plot a normalized radar chart in **Figure 5**, mapping Total, Runtime, PM, IdleHours, and Std (Total) onto [0,1] to jointly reflect solution quality, computational feasibility, and stability. This visualization facilitates an engineering-oriented selection between single-metric optimality and balanced overall performance.

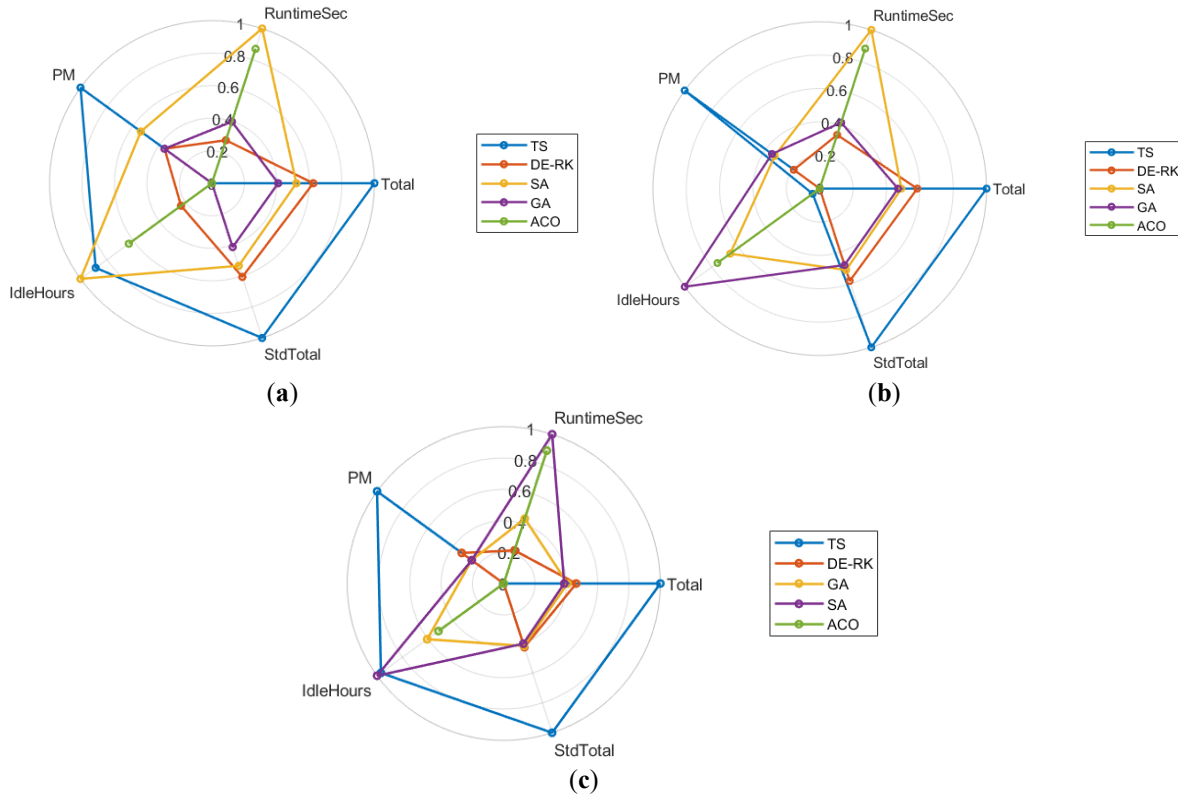


Figure 5 Radar chart of normalized performance (cost, runtime, and stability) under Dynamic_Idle across WT40/WT50/WT100.

Across WT40 (**Figure 5c**) to WT100 (**Figure 5a**), TS exhibits consistently strong performance on the key dimensions of Total and StdTotal, indicating stable high-quality solutions across scales. In contrast, SA shows a clear advantage in runtime in seconds, reflecting its speed-oriented nature, yet it performs relatively weaker on Total/StdTotal, which is consistent with a typical speed–quality–stability trade-off. DE-RK and GA generally occupy an intermediate regime, offering more balanced compromises, although their dominant dimensions vary with problem scale. ACO displays more pronounced variations on behavior-related dimensions (e.g., waiting/maintenance), suggesting stronger sensitivity to the search dynamics and parameterization. Overall, **Figure 5** corroborates the main conclusion of this section: TS is the preferred primary solver when low cost and stability are prioritized, whereas SA is suitable as a fast baseline or initializer under tight time budgets, and the remaining

methods serve as complementary references to illustrate the feasible trade-off space.

4.6 Mechanism and Interpretability

This section elevates the proposed “adaptive threshold + waiting” strategy from a black-box optimization outcome to an interpretable causal evidence chain. We organize complementary evidence from four perspectives: (i) average cost decomposition (**Figure 6**) to identify where improvements come from; (ii) a representative machine-state timeline (**Figure 7**) to show how waiting and PM shift processing across TOU bands; (iii) marginal cost-rate dynamics (**Figure 8**) to connect decisions to high marginal-cost zones; and (iv) a 2×2 ablation (**Figure 9**) to quantify marginal contributions and interactions of adaptiveness and waiting. Together, these results explain not only what changes, but also why they change and under which conditions.

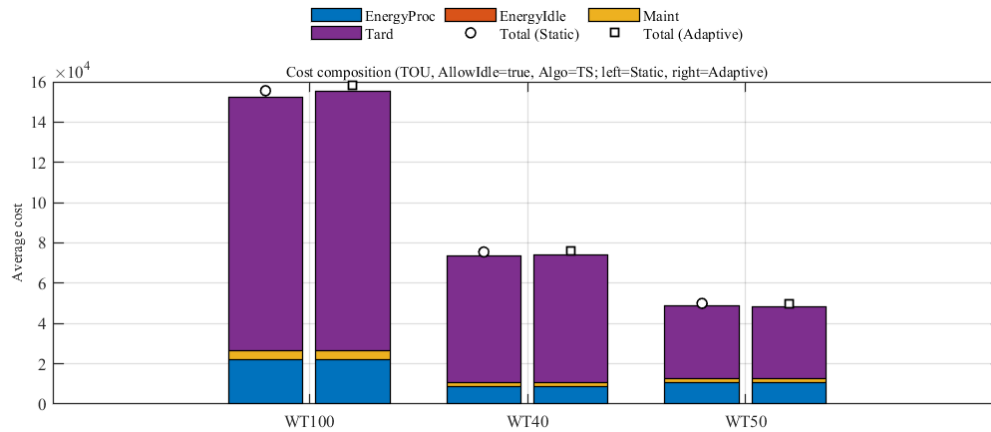


Figure 6 Average cost composition under Fixed vs Adaptive (stacked bars)

Figure 6 decomposes the average total cost under TOU into EnergyProc, EnergyIdle, Maint, and Tard, comparing the fixed-threshold and adaptive-threshold policies. The observed differences mainly follow two pathways: (i) the adaptive threshold together with waiting tends to shift portions of processing into cheaper TOU bands, thereby reducing EnergyProc (lower effective processing price); (ii) waiting delays completion and may

increase Tard, while also increasing EnergyIdle due to idle consumption during waiting. Hence, whether a net Total improvement is achieved depends on whether the reduction in EnergyProc outweighs the increments in Tard and EnergyIdle, as well as whether maintenance triggering (Maint/PM) changes structurally. This decomposition provides the interpretability backbone for the case study and marginal-cost analysis.

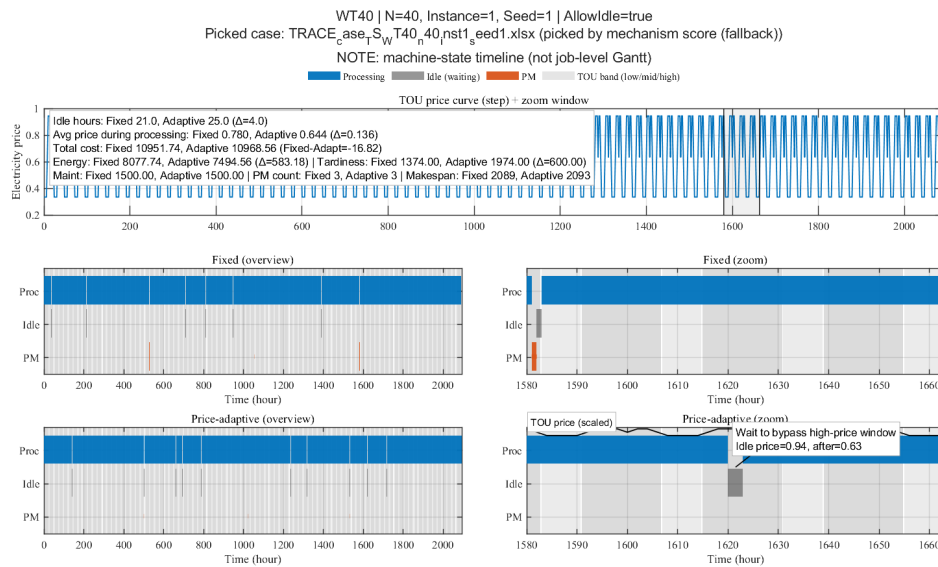


Figure 7 Machine-state timeline under TOU

To avoid obscuring the mechanism by averaging, we present a representative case under WT40 ($N=40$, Instance=1, Seed=1, AllowIdle=true) in **Figure 7**. Note that **Figure 7** is a machine-state timeline rather than a job-level Gantt chart. The case clearly illustrates the “bypass high-price windows via waiting” logic: Adaptive increases waiting time from 21.0 h (Fixed) to 25.0 h ($\Delta=4.0$), reducing the average processing price from 0.780 to 0.644 ($\Delta=0.136$). Consequently, the energy cost decreases from 8077.74 to 7494.56 (saving 583.18).

Meanwhile, the case reveals the cost-side trade-off: waiting delays completion and increases tardiness cost from 1374 to 1974 (+600), yielding a slightly higher

total cost (Fixed 10951.74 vs Adaptive 10968.56; Fixed–Adaptive = -16.82 , i.e., Adaptive is marginally worse here). Maintenance remains unchanged (Maint=1500; PM count 3 for both), and makespan increase only slightly (Fixed 2089 vs Adaptive 2093). This indicates that under strong due-date pressure, the tardiness increment from waiting may offset the energy savings, rendering the net benefit case-dependent. Thus, **Figure 7** provides an interpretable causal chain rather than a “guaranteed improvement” claim: waiting shifts processing from expensive to cheaper TOU bands $\rightarrow$ reduces energy-related cost $\rightarrow$ may raise tardiness and idle energy $\rightarrow$ the net effect is determined by the balance between these terms.

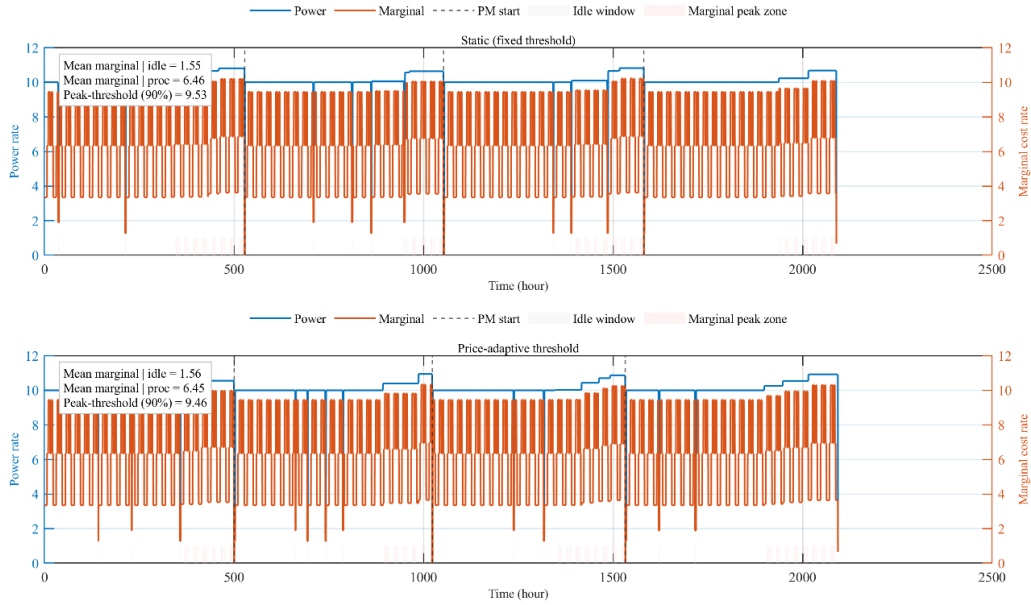


Figure 8 Power consumption rate and marginal cost rate over time

Figure 8 provides a continuous-time explanation of the decision-trigger logic by plotting both the power/load (Power) and the marginal cost rate (Marginal, effectively price $\times$ power), with PM start and idle windows annotated. The marginal cost rate forms periodic peaks aligned with TOU high-price bands, and waiting windows (and some maintenance events) tend to avoid these peak zones, reducing the

fraction of processing executed under high marginal cost. Although the mean change is modest, the alignment between marginal peak zones and idle windows provides direct mechanistic evidence that waiting is used to bypass high marginal-cost periods, supporting the causal interpretation and motivating the sensitivity analyses in Section 4.7.

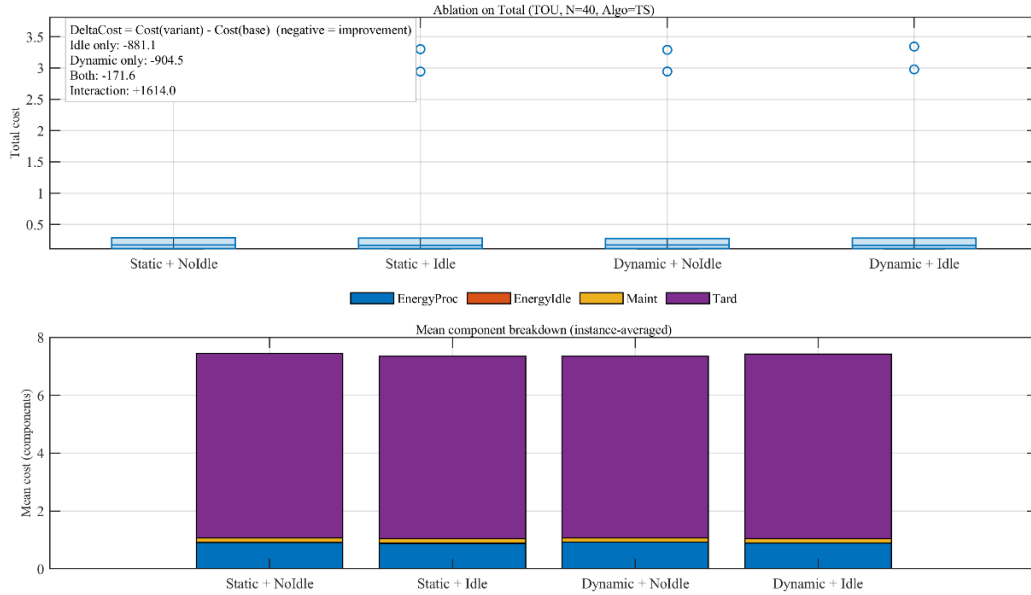


Figure 9 Ablation study (removing adaptive threshold and/or idle)

To quantify the marginal effects of adaptiveness and waiting, and their interaction, **Figure 9** employs a 2×2 ablation design (Static/Dynamic $\times$ NoIdle/Idle) with Static+NoIdle as the base. The plotted ΔCost is defined as $\text{Cost}(\text{variant}) - \text{Cost}(\text{base})$, with negative values indicating improvement. For the illustrated case (TOU, $N=40$, Algo=TS), enabling waiting only (Static+Idle) yields -881.1 ; enabling adaptiveness only (Dynamic+NoIdle)

yields -904.5 ; enabling both (Dynamic+Idle) yields -171.6 . The interaction term is therefore $\text{Interaction} = \Delta C_{\text{Dynamic+Idle}} - \Delta C_{\text{Dynamic+NoIdle}} + \Delta C_{\text{Static+Idle}} = (-171.6) - (-904.5) - (-881.1) = 1614.0$

The positive interaction indicates diminishing returns. Turning on both mechanisms can amplify time-shifting and increase completion delays or tardiness, offsetting part of the energy benefit. The component breakdown further

suggests that tardiness remains the primary driver of total cost differences across the four variants, with energy and maintenance contributing secondarily.

4.7 Robustness and Sensitivity Analysis

To assess robustness against parameter uncertainty, we conduct sensitivity analysis using OAT perturbations and a 2D grid scan. **Table 14** reports the sensitivity factors and baseline values, and **Table 15** reports the perturbation settings. Because weighted tardiness dominates several benchmark totals, the

perturbation list explicitly includes the tardiness penalty as a due-date-pressure robustness factor.

OAT (**Figure 10**) perturbs one parameter at a time while keeping other parameters at the **Table 5** baseline. The heatmap (**Figure 11**) scans (r_0 , price sensitivity) on a grid and colors by MeanTotal. To address due-date-pressure robustness without changing the reported numerical experiments, **Table 15** also records the corresponding tardiness-penalty perturbation setting.

Table 14 Sensitivity factors and baseline values

Factor	Symbol	Unit	Baseline	Main affected components
PM cost per action	C_{PM}	currency/action	500	Maint (direct)
PM duration	T_{PM}	hour/action	1.0 h	Tard (via completion time), TOU timing shift
Degradation rate	λ	1/hour	1/3000	PM frequency, Maint, and Tard
Idle power	P_{idle}	kW	2 kW	EnergyIdle (direct)
Baseline threshold	r_0	-	0.85	PM/wait conservativeness
Price sensitivity	α_{price}	-	0.10	Responsiveness of adaptive threshold
Waiting cap	W_{max}	hour	3.0 h	Trade-off: EnergyProc vs EnergyIdle and Tard
Waiting trigger	Δh_{min}	-	0.10	Activation of waiting
Tardiness penalty	C_{tard}	-	1.0	Tard, waiting, and PM trade-off

Table 15 Perturbation settings for sensitivity analysis

Factor	OAT / due-date-pressure setting	Extended range (optional)	2D grid (Figure 11)
PM cost per action	$\times \{0.9, 1.0, 1.1\}$	$\times \{0.8, 1.0, 1.2\}$	-
PM duration	$\times \{0.9, 1.0, 1.1\}$	$\times \{0.8, 1.0, 1.2\}$	-
Degradation rate	$\times \{0.9, 1.0, 1.1\}$	$\times \{0.8, 1.0, 1.2\}$	-
Idle power	$\times \{0.9, 1.0, 1.1\}$	$\times \{0.8, 1.0, 1.2\}$	-
Baseline threshold	-	-	$\{0.75, 0.80, 0.85, 0.90, 0.95\}$
Price sensitivity	-	-	$\{0, 0.05, 0.10, 0.15, 0.20\}$
Waiting cap	$\times \{0.9, 1.0, 1.1\}$	$\times \{0.8, 1.0, 1.2\}$	-
Waiting trigger	(optional) $\times \{0.9, 1.0, 1.1\}$	(optional) $\times \{0.8, 1.0, 1.2\}$	-
Tardiness penalty	$\times \{0.8, 1.0, 1.2\}$	-	-

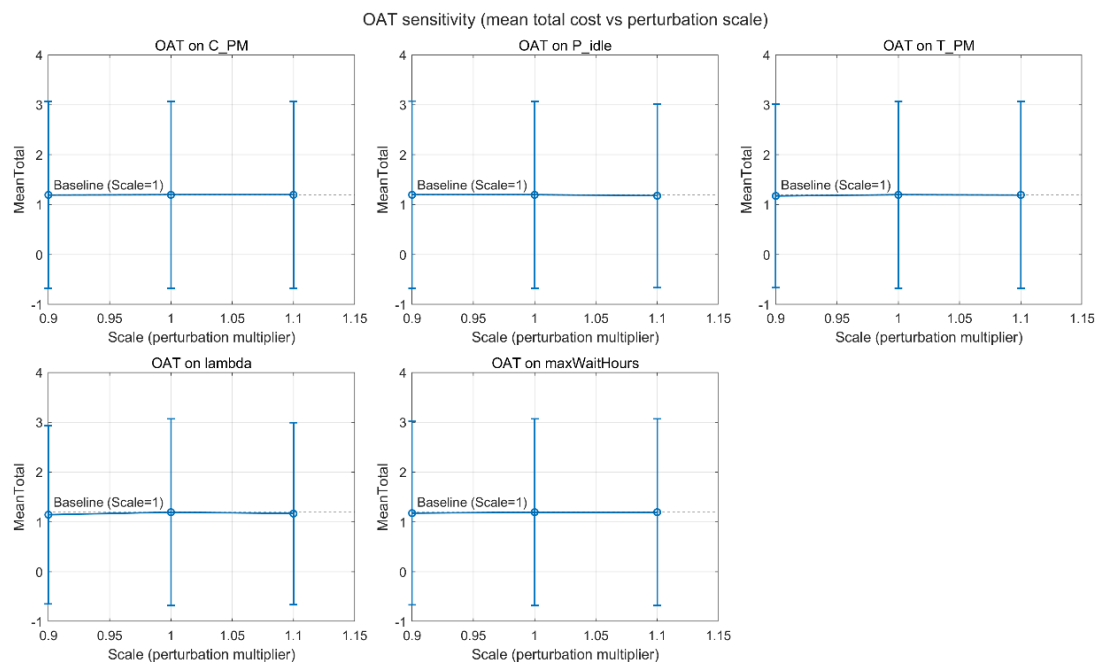


Figure 10 OAT sensitivity curves

As shown in **Figure 10**, MeanTotal remains largely unchanged when perturbing C_{pm} , T_{pm} , idle power, and W_{max} , within $\pm 10\%$. This indicates strong local robustness with no systematic monotonic degradation or improvement.

The tardiness penalty is included in **Table 15** as an additional due-date-pressure robustness design rather than as an additional plotted result. It should be evaluated with the same instance-seed protocol when the extended sensitivity run is conducted.

Because the adaptive policy is jointly determined by (r_0 , price sensitivity), and these parameters may interact, we further examine their coupled sensitivity using a two-dimensional grid scan. **Figure 11** presents MeanTotal as a heatmap with contour lines, where lower values are better. A low-cost basin emerges around certain (r_0 , price sensitivity) combinations.

In addition to parameter uncertainty, stochastic metaheuristics introduce randomness through initialization and search trajectories, making seed-level

stability important for reproducibility. **Figure 12** reports the distribution of seed-level mean total cost under Dynamic_Idle_TOU, where the mean value for each seed is computed across the tested instances within each WT dataset.

The results indicate that TS achieves a consistently lower median with narrower dispersion, demonstrating both competitive performance and strong robustness to stochasticity. Wider boxes and occasional outliers for some algorithms suggest higher seed sensitivity and sporadic convergence to inferior local optima.

In summary, this section validates the price-driven adaptive reliability-threshold mechanism through degeneration consistency under FLAT pricing and interpretable responsiveness under TOU pricing. Enumeration provides auditable global-optimal references for small instances. On medium-to-large WT benchmarks, TS achieves the best paired total-cost performance, and **Tables 12–13** show that the Friedman and Holm-corrected Wilcoxon tests support the observed algorithmic ranking.

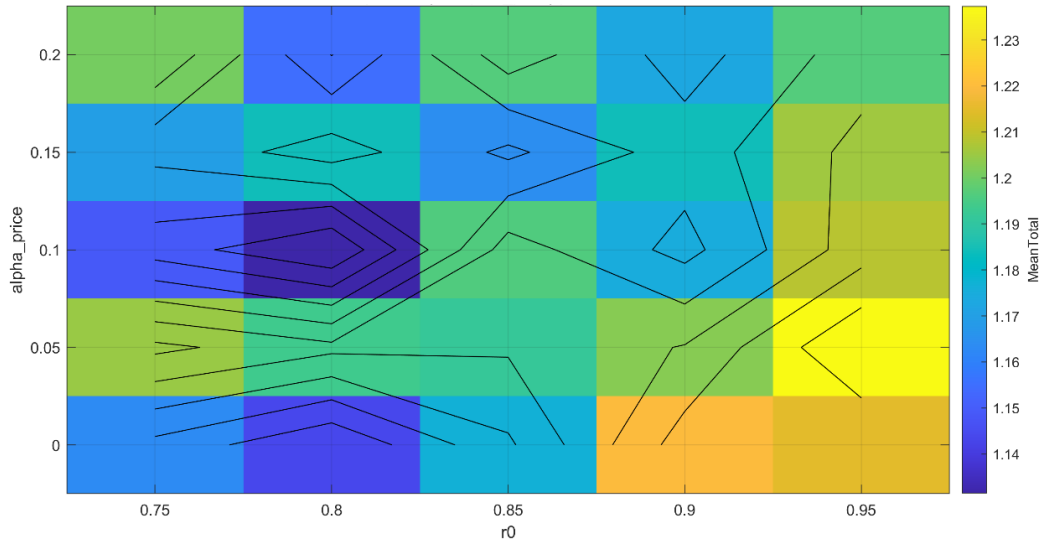


Figure 11 Heatmap over (r_0 , α_{price})

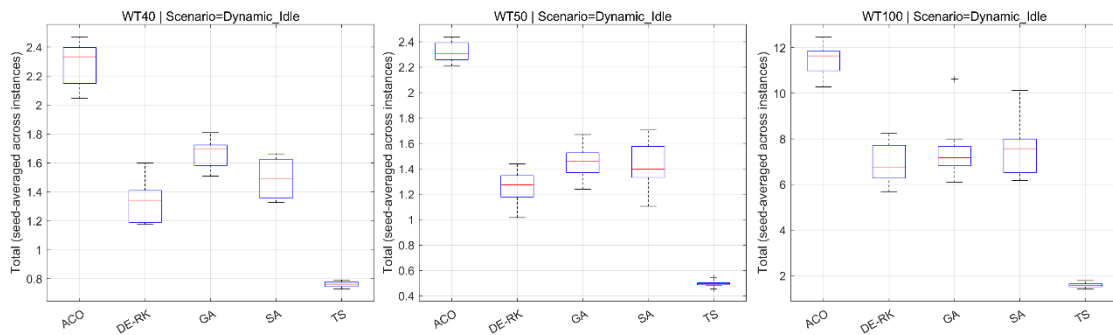


Figure 12 Robustness across random seeds under Dynamic_Idle (TOU)

4.8 Industrial Applicability and Implementation Scenario

Industrial implementation requires mapping the abstract scheduling variables to observable shop-floor signals. The framework is most suitable for bottleneck

machines whose operating age or condition index is available and whose electricity cost varies materially across the day. **Table 16** gives representative deployment settings without changing the numerical experiments reported above.

Table 16 Representative industrial implementation scenarios

Industrial setting	Bottleneck machine	Jobs / product type	TOU relevance	Degradation indicator	Decision supported
CNC machining	Spindle or machining center	Machining orders, high-mix parts	Peak-valley electricity for machining load	Spindle operating time or condition index	PM timing and short waiting before jobs
Heat treatment	Furnace	Batches or customer orders	High energy intensity during heating cycles	Accumulated heating hours or thermal cycles	Avoiding peak processing and planning PM
Coating line	Coating or curing line	Product lots requiring controlled processing	Time-varying electricity during curing/operation	Operating age, coating-head wear, or line condition	Threshold-based PM and wait/no-wait decision
Testing station	High-load test bench	Product qualification tests	Long test runs exposed to peak tariffs	Accumulated test hours or sensor-based condition index	Sequencing, PM timing, and limited waiting

5. Conclusion

This paper investigated a degradation-aware single-machine scheduling problem under time-varying electricity tariffs. Compared with prior TOU scheduling studies that mainly treat power as fixed or exogenous, the proposed framework explicitly couples reliability-driven degradation, a reliability-power mapping, PM restoration, waiting decisions, and weighted tardiness. This positioning clarifies how the study extends current knowledge on TOU scheduling, maintenance-integrated scheduling, and idle-time control.

The core methodological contribution is an interpretable, tariff-responsive reliability-threshold policy $r_{th}(t)$ bounded by r_{min} and r_{max} , supplemented by a conservative waiting trigger. The threshold increases in high-price windows to reduce exposure to low-reliability processing when electricity is expensive, and it relaxes in low-price windows to mitigate due-date pressure. Under FLAT pricing, the policy degenerates to a static baseline, which provides a direct legitimacy check.

Comprehensive experiments on OR-Library WT40/WT50/WT100 provide three layers of evidence. Degeneration verification confirms behavioral consistency under FLAT and rational responsiveness under TOU. Enumeration on small instances establishes auditable optima for approximation assessment. Medium-to-large benchmark results show that TS provides the best overall balance of total cost, stability, and runtime under the tested settings, while mechanism and ablation analyses reveal that adaptiveness and waiting interact non-additively under strong due-date pressure.

From a managerial perspective, the proposed policy offers transparent decision rules rather than black-box prescriptions. It is most relevant for bottleneck machines in machining, heat treatment, pressing, coating, or testing operations where electricity prices vary over time and machine condition affects power demand. When tariffs are high and due-date slack is sufficient, conservative

thresholds and selective waiting can reduce marginal energy exposure. When due-date pressure dominates, waiting should be restricted and thresholds should be relaxed to avoid excessive tardiness penalties.

Several limitations motivate future research. First, the policy-induced approach restricts the decision space; future work can jointly optimize sequence, PM timing, and waiting duration under richer constraints such as release dates, sequence-dependent setup, stochastic processing, and condition-monitoring uncertainty. Second, the degradation and power models can be calibrated with machine-level sensor data, including idle-state, startup, shutdown, and thermal-cycling effects. Third, extending the framework to parallel-machine and flow-shop environments would improve industrial applicability. Future work may also include stricter evaluation-budget-matched algorithm comparisons and instance-level statistical validation.

6. Statements & Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Funding

Not applicable.

Availability of data

All the information used is publicly available in the article. The codes are available on reasonable request.

Acknowledgements

Not applicable.

7. References

- [1] M. Ghorbanzadeh and M. Ranjbar, "Energy-aware production scheduling in the flow shop environment under sequence-dependent setup times, group scheduling and renewable energy constraints," *European Journal of Operational*

- Research*, vol. 307, no. 2, pp. 519–537, 2023, doi: 10.1016/j.ejor.2022.09.034.
- [2] P. Wu, N. Li, J. Cheng, and C. Chu, “Energy-efficient single-machine scheduling with release dates under time-of-use electricity tariffs,” *Journal of Cleaner Production*, vol. 393, 2023, Art. no. 136228, doi: 10.1016/j.jclepro.2023.136228.
- [3] S. Lee, M. Issabakhsh, H. W. Jeon, S. W. Hwang, and B. Do Chung, “Idle time and capacity control for a single machine scheduling problem with dynamic electricity pricing,” *Operations Management Research*, vol. 13, no. 3–4, pp. 197–217, 2020, doi: 10.1007/s12063-020-00156-x.
- [4] M. -J. Park and A. Ham, “Energy-aware flexible job shop scheduling under time-of-use pricing,” *International Journal of Production Economics*, vol. 248, 2022, Art. no. 108507, doi: 10.1016/j.ijpe.2022.108507.
- [5] J. Friederich and S. Lazarova-Molnar, “Reliability assessment of manufacturing systems: A comprehensive overview, challenges and opportunities,” *Journal of Manufacturing Systems*, vol. 72, pp. 38–58, 2024, doi: 10.1016/j.jmsy.2023.11.001.
- [6] M. A. Sellitto and B. Pinho, “Maintenance Strategy Choice Supported by the Failure Rate Function: Application in a Serial Manufacturing Line,” *Periodica Polytechnica Social and Management Sciences*, vol. 31, no. 1, pp. 38–51, 2023, doi: 10.3311/PPso.18627.
- [7] X. He, Z. Wang, Y. Li, S. Khazhina, W. Du, J. Wang, and W. Wang, “Joint decision-making of parallel machine scheduling restricted in job-machine release time and preventive maintenance with remaining useful life constraints,” *Reliability Engineering & System Safety*, vol. 222, 2022, Art. no. 108429, doi: 10.1016/j.ress.2022.108429.
- [8] J. Duan and J. Wang, “Energy-efficient scheduling for a flexible job shop with machine breakdowns considering machine idle time arrangement and machine speed level selection,” *Computers & Industrial Engineering*, vol. 161, 2021, Art. no. 107677, doi: 10.1016/j.cie.2021.107677.
- [9] J. Cheng, F. Chu, M. Liu, P. Wu, and W. Xia, “Bi-criteria single-machine batch scheduling with machine on/off switching under time-of-use tariffs,” *Computers & Industrial Engineering*, vol. 112, pp. 721–734, 2017, doi: 10.1016/j.cie.2017.04.026.
- [10] Z. Tian and L. Zheng, “Single machine parallel-batch scheduling under time-of-use electricity prices: New formulations and optimisation approaches,” *European Journal of Operational Research*, vol. 312, no. 2, pp. 512–524, 2024, doi: 10.1016/j.ejor.2023.07.012.
- [11] S. Yuan, B. Wang, Y. Pei, and T. Li, “Energy-efficient single-machine scheduling with group processing features under time-of-use electricity tariffs,” *Computers & Operations Research*, vol. 181, 2025, Art. no. 107100, doi: 10.1016/j.cor.2025.107100.
- [12] G. Gong, R. Chiong, Q. Deng, W. Han, L. Zhang, and D. Huang, “Energy-efficient production scheduling through machine on/off control during preventive maintenance,” *Engineering Applications of Artificial Intelligence*, vol. 104, 2021, Art. no. 104359, doi: 10.1016/j.engappai.2021.104359.
- [13] G. Gong, R. Chiong, Q. Deng, X. Gong, W. Lin, W. Han, and L. Zhang, “A two-stage memetic algorithm for energy-efficient flexible job shop scheduling by means of decreasing the total number of machine restarts,” *Swarm and Evolutionary Computation*, vol. 75, 2022, Art. no. 101131, doi: 10.1016/j.swevo.2022.101131.
- [14] H. -J. Kim, E. -S. Kim, J. -H. Lee, L. Tang, and Y. Yang, “Single-machine scheduling with energy generation and storage systems,” *International Journal of Production Research*, vol. 60, no. 23, pp. 7033–7052, 2022, doi: 10.1080/00207543.2021.2000655.
- [15] J. Chen, T. Ning, G. Xu, and Y. Liu, “A memetic algorithm for energy-efficient scheduling of integrated production and shipping,” *International Journal of Computer Integrated Manufacturing*, vol. 35, no. 10–11, pp. 1246–1268, 2022, doi: 10.1080/0951192X.2022.2025618.
- [16] D. B. M. M. Fontes, S. M. Homayouni, and J. C. Fernandes, “Energy-efficient job shop scheduling problem with transport resources considering speed adjustable resources,” *International Journal of Production Research*, vol. 62, no. 3, pp. 867–890, 2024, doi: 10.1080/00207543.2023.2175172.
- [17] M. Ghorbanzadeh, M. Davari, and M. Ranjbar, “Energy-aware flow shop scheduling with uncertain renewable energy,” *Computers & Operations Research*, vol. 170, 2024, Art. no. 106741, doi: 10.1016/j.cor.2024.106741.