Software-in-the-loop validation of a real-time receding-horizon enhanced dung beetle optimization algorithm for solar-connected smart-grid energy management

Huynh Tuyet Vy[1], Ho Pham Huy Anh[2,3,*]

[1]Faculty of Electronic Technology, Industrial University of Ho Chi Minh City, Ho Chi Minh City, Vietnam (e-mail: huynhtuyetvy@iuh.edu.vn)
[2]Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, Dien Hong Ward, Ho Chi Minh City, Viet Nam
[3]Vietnam National University Ho Chi Minh City (VNU-HCM), Dong Hoa Ward, Thu Duc District, Ho Chi Minh City, Viet Nam
[*]Corresponding author E-mail: hphanh@hcmut.edu.vn
DOI: https://doi.org/10.64032/mca.v30i4.459

Abstract

The Enhanced Dung Beetle Optimization (EDBO) algorithm, which augments the Dung Beetle Optimizer with the Harris Hawks Optimizer and the Nelder–Mead simplex, has demonstrated strong performance for offline energy-management optimization in solar-connected residential micro-grids. However, its behavior under real-time and communication-constrained operation has not been established, because the offline formulation solves a single 168-variable weekly problem with full future knowledge and a very large computational budget. This paper reformulates the EDBO-based energy management system (EMS) as a real-time receding-horizon controller and validates it in a software-in-the-loop (SIL) framework - a first step toward Hardware-in-the-Loop (HIL) deployment - in which the controller and a real-time smart-grid plant model run as separate processes on a development workstation and exchange data over a real Modbus TCP link (localhost). The controller re-optimizes a short look-ahead window each hour from the live battery state of charge, reducing the per-cycle dimensionality from 168 to 48 variables. Across a six-scenario matrix, the controller met the scaled real-time deadline on all 1,008 control cycles, reproduced the published no-storage baseline exactly (40.012 USD/week), achieved a 6.72% weekly cost reduction on the nominal scenario (mean over five seeds)-within 0.67% of the exact linear-programming optimum (37.07 USD/week) established under sustainable terminal constraints-and remained stable and robust under price spikes, load steps, 20% forecast error and 15% communication packet loss. A deterministic linear-programming baseline confirms that, for the convex base model, an exact solver is globally optimal and far faster; the gradient-free EDBO framework is retained because, unlike the LP, it extends without reformulation to the non-convex features common in practical battery energy-management systems, such as power-dependent converter efficiency. The results indicate that the receding-horizon reformulation recovers essentially all of the attainable economic benefit while operating in a real-time, communication-in-the-loop closed loop; embedded (Raspberry Pi) and Power-HIL validation are identified as the next steps.

Keywords: Dung Beetle Optimization; Software-in-the-Loop; receding-horizon optimization; energy management system; smart grid; Modbus TCP; real-time control.

1. Introduction

The transition toward distributed, renewable-integrated smart grids has made intelligent energy management a decisive factor for both economic and environmental performance [1][2]. In a solar-connected residential micro-grid, an energy management system (EMS) must continuously dispatch a battery energy storage system (BESS) against fluctuating photovoltaic (PV) generation, time-varying residential load and dynamic electricity prices, so that the total operating cost is minimized while all physical constraints are respected. Population-based meta-heuristics have become a popular tool for this non-convex, constrained problem because they require no gradient information and handle complex objective landscapes robustly [3][4][5].

The Enhanced Dung Beetle Optimization (EDBO) algorithm was recently proposed for this purpose [3]. EDBO augments the canonical Dung Beetle Optimizer (DBO) [4] with the fast-convergence exploitation of the Harris Hawks Optimizer (HHO) [5] and the deterministic local refinement of the Nelder–Mead (NM) simplex [6], hybridized selectively across the DBO sub-populations. On thirty CEC-2017 benchmark functions EDBO attained the best overall Friedman–Nemenyi ranking against DBO, HHO and three recent meta-heuristics (MH), and on a residential smart-grid case study it reduced the weekly operating cost from a 40.012 USD/week baseline to 36.981 USD/week, a 7.58% saving [3].

These results, however, were obtained entirely in offline simulation on a workstation: the entire week of load, PV and price data was assumed known in advance, the problem was solved once as a single 168-variable optimization, and a very large computational budget (population 35, 15,000 iterations) was used. A deployable EMS operates under fundamentally different conditions. Decisions must be produced online, within a strict control-cycle deadline, using only past and forecast information; the algorithm must run on constrained embedded hardware such as a low-cost single-board computer; and the controller must exchange measurements and commands with the plant through industrial communication channels that introduce latency, jitter and packet loss. None of these dimensions can be assessed by offline simulation, and a naive port of the offline algorithm to an embedded controller would violate any realistic real-time deadline. This disconnect between offline benchmark performance and field deployability is common to the entire DBO-family EMS literature [7][8][9], where - to the best of our knowledge - no work has yet reported an embedded, real-time, communication-in-the-loop validation.

Hardware-in-the-Loop (HIL) testing is the accepted bridge between simulation and field deployment [10][11]. As a first, hardware-independent step, this study adopts a software-in-the-loop (SIL) realization: the controller and a

Received: 13 July 2026; Revised: 14 October 2026; Accepted: 19 October 2026.

real-time plant model run as separate processes communicating over a real Modbus TCP link, so that the closed-loop control logic, the communication interface and the real-time timing behavior are all exercised before any physical hardware is introduced. To operate a many-evaluation MH within a real-time budget, EDBO is embedded inside a receding-horizon loop [12][13]. The main contributions are:

- A real-time formulation of the EDBO-based EMS based on receding-horizon optimization, reducing the per-cycle problem dimension from 168 to a short window (48 hours) and enabling warm-starting between cycles.
- A complete, reproducible SIL platform - the first stage of a HIL workflow - in which the controller and a real-time smart-grid plant model communicate over a real Modbus TCP link, with an explicit register data-map and a documented per-cycle message sequence, designed to run unchanged on physical hardware.
- A quantitative, six-scenario validation reporting optimization quality, per-cycle computation time, communication latency, real-time deadline compliance, reliability under packet loss, and sensitivity to forecast error - all against a single, consistent cost model that reproduces the published baseline exactly.
- Evidence that the real-time controller recovers essentially all of the attainable economic benefit (within 0.67% of the exact linear-programming optimum) while meeting a scaled real-time deadline over a real communication link - indicating, though not yet proving, embedded deployability.

The remainder of this paper is organized as follows. Section 2 reviews related work and positions the contribution. Section 3 formalizes the smart-grid model and operating cost. Section 4 summarizes the EDBO algorithm. Section 5 presents the real-time receding-horizon controller. Section 6 describes the SIL platform and communication architecture. Sections 7 and 8 report the experimental setup and results, and Section 9 concludes.

2. Related works

Population-based MH are widely used for micro-grid scheduling and battery dispatch because they require no convexity or differentiability and handle mixed, non-linear constraints [14]. Among them, the Dung Beetle Optimizer (DBO) [4] has attracted attention in the energy domain for its accuracy and convergence, and recent variants have been applied to micro-grid scheduling [15][16][17] and photovoltaic forecasting [18][19]. The Enhanced DBO (EDBO) [3] hybridizes DBO with the fast exploitation of the Harris Hawks Optimizer and the derivative-free local refinement of the Nelder–Mead simplex , activated selectively within the DBO sub-populations to mitigate premature convergence; it was validated offline on benchmark functions and on a residential smart-grid case [3][14][20]. Compared with the alternatives - rule-based controllers (real-time but sub-optimal), convex or mixed-integer programs (near-optimal and fast but restricted to a specific mathematical form) [21], and reinforcement learning (offline training, data-hungry) - MHs offer black-box flexibility at the price of many objective evaluations, which is the principal obstacle to their real-time deployment.

The standard remedy for the computational cost of optimization-based control is Model Predictive Control (MPC), or receding-horizon control, which repeatedly optimizes a short look-ahead window, applies only the first action, and rolls the window forward using fresh measurements [12][22]. Most MPC-based EMS formulations, however, rely on convex or mixed-integer solvers rather than MH, whereas the MH EMS literature almost invariably solves a single full-horizon problem offline. Validating such a controller before field deployment is the role of HIL testing, in which the controller runs on real or emulated hardware while the plant is reproduced on a real-time simulator, exchanging data over an industrial link such as Modbus TCP [11][23][24].

The literature therefore leaves a clear gap: DBO and its variants deliver strong offline optimization quality [3][14][20], while real-time control and HIL validation are mature but have not been brought to bear on hybrid MH EMS. EDBO in particular has never been exercised under embedded, real-time, communication-constrained conditions. This paper fills that gap by reformulating the EDBO-based EMS for real-time receding-horizon operation, validating it in a SIL framework over a real Modbus TCP link, and quantifying, against a single, consistent cost model that reproduces the published baseline exactly, the practical cost of moving from offline simulation to deployable operation.

3. Smart-grid system model

To make the paper self-contained, this section develops the smart-grid model and the operating-cost objective in full; it follows the formulation validated offline in [3] but is presented here from first principles so that the controller and results in the following sections can be read without reference to the earlier work.

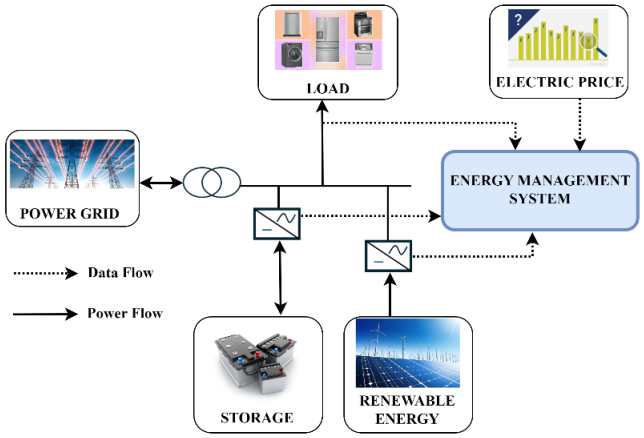


Figure 1: Home power grid with integrated renewable energy and storage.

The system, shown in Fig. 1, is a single-household solar-connected micro-grid with four power components: a rooftop photovoltaic array that injects generation, an aggregated residential load that draws demand, a lithium-ion battery energy storage system (BESS) that can both absorb and inject power, and a bidirectional connection to the utility grid that supplies any deficit or absorbs any surplus. An EMS - the

controller developed in this paper - observes the system state and the price signal and issues, each hour, a single decision: the battery charge or discharge power. Everything else follows from that decision through the instantaneous power balance at the bus.

3.1 Battery storage model

The battery state is represented by the state of charge (SoC), $S(t)$, round-trip efficiency η, and energy storage limits. At each time step, the optimization algorithm proposes a charging/discharging command $\Delta(t)$, while $a(t)$ denotes the command actually applied to the battery after enforcing operational constraints. Accounting for charging and discharging losses, the candidate SoC is computed as

$$S(t)=\begin{cases} S(t-1)+\eta\Delta(t), \Delta(t)\geq 0 \\ S(t-1)+\dfrac{\Delta(t)}{\eta}, \Delta(t)<0 \end{cases} \quad (1)$$

where $S(t)$ is the candidate SoC. The applied action $a(t)$ equals the proposed action $\Delta(t)$ whenever the resulting SoC remains within the admissible range; otherwise, the proposed action is discarded and the battery remains idle:

$$a(t)=\begin{cases} \Delta(t), S_{\min}<S(t)<S_{\max} \\ 0 \quad ,otherwise \end{cases} \quad (2)$$

Yielding the realized SoC as follows:

$$S(t)=\begin{cases} S(t), S_{\min}<S(t)<S_{\max} \\ S(t-1), otherwise \end{cases} \quad (3)$$

This feasibility projection ensures that the battery always operates within its prescribed SoC limits.

3.2 Operating cost and constraints

Table 1: Smart-grid and battery model parameters

Parameter	Value
Horizon	168 h (24 h × 7 days)
BESS maximum / initial capacity	12 kWh / 6 kWh
BESS rated power	3 kW
Charge / discharge caps	10 kWh / 2 kWh
Charge/discharge efficiency	0.95
Battery depreciation cost	0.075 USD per kWh (per cycle term)
Sell price	90% of buy price
Load / solar / buy price	NREL and U.S. BLS, San Francisco [25], [26], [27]

The EMS minimizes the weekly operating cost, defined as the sum over the horizon of the grid transaction cost, including buy price $(e_{buy}(t))$, sell price $(e_{sell}(t))$, and the battery depreciation cost:

$$\min C=\sum_{t=1}^{168}\left(\begin{array}{l} \dfrac{e_{buy}(t)+e_{sell}(t)}{2}P_G(t) \\ +\dfrac{e_{buy}(t)-e_{sell}(t)}{2}\left|P_G(t)\right|+C_s\left|a(t)\right| \end{array}\right) \quad (4)$$

subject to the hourly load-balance equation $P_G(t)+a(t)+P_{PV}(t)=P_{load}(t)$ with $P_G(t), P_{PV}(t), and\ P_{load}(t)$ represent the grid energy, solar energy, and load energy, respectively at the t current time. Let C_s be depreciation coefficient. Additional constraints include the battery rated-power limit $|a(t)|\leq P_{rated}$, and the state-of-charge bounds $S_{\min}<S(t)<S_{\max}$. The model parameters, taken directly from [3], are listed in Table 1.

4. Real-time receding-horizon EDBO controller

The offline formulation optimizes the full decision vector $x=[\Delta_1,...,\Delta_{168}]$ at once. The proposed controller instead optimizes, at each hour t, only the next H hourly commands $x=[\Delta_t,...,\Delta_{t+h-1}]$ from the current state of charge and an H-hour forecast; it applies only the first command Δ_t, advances one hour, and re-optimizes the shifted window, warm-started from the previous solution. This reduces the per-cycle dimensionality from 168 to H and dramatically shrinks the search space.

Table 2: Controlled look-ahead-horizon study (nominal scenario, 200 iterations, population 30, mean of three seeds)

Window H (h)	Weekly cost (USD/week)	Saving vs baseline (%)	Mean solve time (ms)
12	38.34	4.18	207
24	37.43	6.45	295
48 (selected)	37.32	6.73	444
72	37.39	6.55	571

Algorithm 1: Receding-horizon EDBO control loop

```
for t = 0 … 167 do
    read (t, SoC) from plant over Modbus TCP
    build window forecast [t : t+H) of load, PV, price
    warm ← shift(previous_plan)
    Δ* ← EDBO.solve(window, SoC, warm) // H variables
    send Δ*[0] to plant over Modbus TCP
    previous_plan ← Δ*
end for
```

The window length H is a design parameter. Because a 12kWh battery cannot economically shift energy across a whole week, its useful planning horizon is on the order of one day. In a controlled study on the nominal scenario - every horizon evaluated under the same budget of 200 iterations and population 30, averaged over three random seeds - the mean weekly saving was 4.18% at $H=12h$, 6.45% at $H=24h$, 6.73% at $H=48h$ and 6.55% at $H=72h$, while the mean per-cycle solve time grew monotonically from 0.21 s to 0.57 s (Table 2). The saving peaks at $H=48h$ and declines for

longer horizons as near-term decisions are diluted; $H = 48\,h$ was therefore selected as the best cost/computation trade-off. Three refinements keep each solve within an embedded budget while preserving quality: warm-starting from the previous cycle's shifted plan; adaptive budgeting of population and iteration count with convergence-based early stopping; and clipping of infeasible commands before actuation so the plant never receives a state-violating set-point. The control loop is summarized in Algorithm 1.

The reformulation transforms the computational profile fundamentally. The offline solve performs population × iterations ≈ 35 × 15,000 ≈ 5.25 × 10^5 evaluations of a 168-dimensional objective, once. Each real-time cycle instead performs only 30 × 200 = 6,000 evaluations of a 48-dimensional objective, and the window objective is evaluated by a vectorized recursion over the horizon, so the per-cycle cost is small and, crucially, bounded and predictable - the property a real-time deadline requires. Warm-starting exploits the temporal coherence of successive windows: because consecutive problems differ only by one shifted hour, seeding the population with the previous plan places the search close to a good solution and reduces the iterations needed for convergence. The adaptive-budget rule sizes the population and iteration count to the measured per-evaluation time and the target cycle deadline, and stops early once the incumbent fitness stagnates, so the controller degrades gracefully on slower hardware rather than missing deadlines. Finally, clipping infeasible commands before actuation guarantees that the plant never receives a set-point that would violate the hard state-of-charge limits, decoupling constraint satisfaction from the stochastic search.

The per-cycle cost of the controller is O(P·I·H), where P is the population, I the iteration count and H the window length, since each of the P·I candidate evaluations performs an O(H) recursive pass over the window; with P = 30, I = 200 and H = 48 this is a small, fixed budget that does not grow with the week length. Convergence-based early stopping terminates a cycle once the incumbent has not improved for a fixed number of iterations, so easy cycles finish faster and free time for harder ones, while the warm start typically places the initial population within a few percent of the final cost. Because the window objective reuses the exact battery dynamics and cost accounting of Section 3, feasibility and cost are evaluated identically to the offline formulation.

5. Software-in-the-loop platform and path to HIL

The controller and the real-time plant are separate software processes exchanging data over a real Modbus TCP link, as shown in Fig. 2. The validation workflow is staged: Stage-1 SIL, the stage executed and reported in this paper, runs both processes on a single development workstation through exactly the software interfaces (including the Modbus binding, here over localhost) that the later stages will use; Stage-2 Controller-HIL (CHIL) [24] would place the controller on an embedded target such as a Raspberry Pi 4/5 and the plant on a dedicated real-time simulator across a physical network; and Stage-3 Power-HIL would add real inverters and a physical battery. Stages 2 and 3 are the deployment path and are left to future work; no physical hardware was used in the present study. Crucially, even Stage-1 SIL is not equivalent to the offline simulation: it executes the full closed-loop controller with 168 sequential solves and live state feedback across the communication interface, whereas the offline study is a single open-loop optimization.

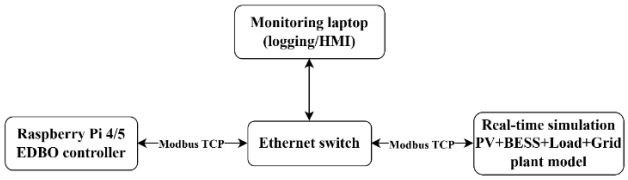


Figure 2: Hardware – in – the loop connection diagram

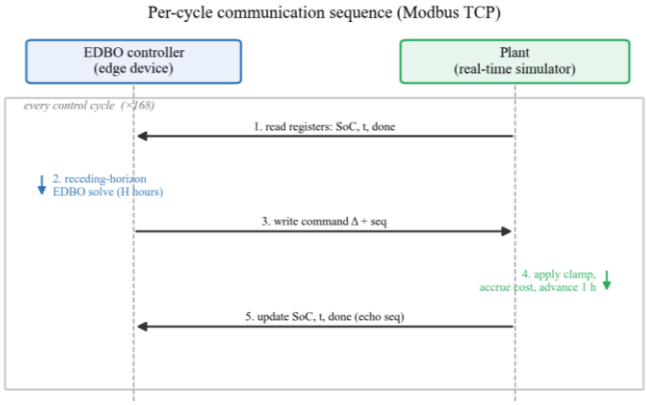


Figure 3: Per-cycle communication sequence over Modbus TCP

The real-time plant reproduces the four components of the micro-grid. The PV source is a time-series power input scaled from NREL solar-resource data [25], optionally perturbed by fast irradiance transients to emulate passing clouds; the load is an aggregated residential demand profile from NREL data [26], with the ability to inject step changes; the BESS is modelled by the discrete state-of-charge dynamics of (1)-(3), including efficiency and hard energy/power limits, so that the plant enforces the same physical constraints the controller optimizes against; and the grid is an ideal bidirectional connection priced by the time-varying buy/sell series [27]. Each cycle, the plant exposes the measured state of charge, timestep and completion flag, applies the received command through the same feasibility projection used in optimization, accrues the realized operating cost, and advances one hour.

Table 3: Modbus TCP holding – register data - map

Register	Type	Direction	Meaning
HR0–1	float32	plant → controller	Battery state of charge (kWh)
HR2	uint16	plant → controller	Timestep t (hour of week)
HR3	uint16	plant → controller	Done flag (week complete)
HR4	uint16	plant → controller	Processed sequence (echo)
HR10–11	float32	controller → plant	Battery power command Δ (kW)
HR12	uint16	controller → plant	Command sequence counter

The Modbus data-map is given in Table 3. Live state (SoC, timestep, completion flag) flows from plant to controller; the battery command and a sequence counter flow from controller to plant. Floating-point quantities occupy two consecutive 16-

bit holding registers (big-endian float32), and the sequence counter forms the handshake by which the controller confirms each command has been applied. The resulting per-cycle message sequence is shown in Fig. 3.

6. Experimental setup

The controller used a 48-hour window, 200 EDBO iterations and a population of 30 per cycle, against a 2-second per-cycle real-time deadline; each scenario was driven for a full 168-hour week over the Modbus TCP link. Six scenarios stress both optimization and real-time behavior: S1 nominal (identical to the offline case study); S2 low-PV (solar ×0.25 on days 3–5); S3 PV surplus (solar ×1.8); S4 evening price spike (buy price ×2, hours 17–20); S5 sustained +50% load on day 4; and S6 communication dropout (15% packet loss, handled by a safe-default idle command). Performance was measured by realized weekly cost and saving versus a no-storage baseline; optimality gap to the exact optimum of the convex model (Section 7.1.1); per-cycle solve time; Modbus round-trip latency; deadline-miss rate; dropped commands; and constraint violations. The no-storage baseline (40.012 USD/week) reproduces the published benchmark in [3] exactly. The earlier offline study reported a weekly cost of 36.981 USD/week under an unconstrained terminal condition; however, when enforcing sustainable cyclic operation with terminal energy neutrality $\left(SoC(T) \geq SoC(0) = 6kWh\right)$, the exact lower bound computed by linear programming is 37.07 USD/week (Table 4).

The scenarios are designed to probe distinct stress axes rather than to sample a distribution: S2 and S3 bracket the generation extremes (scarce and abundant PV); S4 and S5 introduce sharp temporal events in price and demand that test the controller's ability to react within its window; and S6 targets the communication layer directly. S1 is retained as the reference because it is identical to the offline case study, enabling a like-for-like comparison. Each scenario is executed over a full 168-hour week so that end-of-horizon effects and the accumulation of small per-cycle sub-optimalities are captured, rather than being hidden by a short run. The evaluation metrics are defined as follows. The optimality gap of the real-time controller relative to the exact linear-programming optimum (Section 7.1.1) F_{off}, the real-time deadline-miss rate over the N control cycles, and the mean round-trip communication latency over the M Modbus transactions are as follows:

$$g = 100\frac{F_{RHO} - F_{off}}{F_{off}}[\%] \tag{5}$$

$$\rho = \frac{1}{N}\sum_{k=1}^{N} 1\left[\tau_{cyc}(k) > T_d\right] \tag{6}$$

$$\overline{\tau_c} = \frac{1}{M}\sum_{m=1}^{M} \tau_c(m) \tag{7}$$

where T_d is the per-cycle deadline, τ_{cyc} is the measured cycle time, $1[\cdot]$ the indicator function, and $\tau_c(m)$ is the round-trip time of the m^{th} transaction. The per-cycle solve time τ_{solve} is reported by its mean, 99th percentile and maximum.

7. Results and discussion

7.1 Convexity analysis and solver selection

7.1.1 Convex base model

The EMS optimization problem defined by Eqs. (1)-(4) is convex: the operating cost is piecewise linear in the battery power, the battery efficiency is constant ($\eta = 0.95$), and all constraints are linear. The problem can therefore be reformulated as a linear program (LP) by splitting the battery power into non-negative charge and discharge variables and the grid power into import and export variables. This transformation linearizes the piecewise buy/sell pricing structure, yielding a 4T-variable LP that is solved to global optimality using a standard solver. For a convex problem of this form, an exact LP solver is generally preferable to a gradient-free meta-heuristic because it guarantees the global optimum with significantly lower computational effort.

Table 4 compares the LP optimum with the RHO-EDBO controller across the scenario matrix. The LP reaches the global optimum in approximately 20–25 ms and provides a lower bound on the operating cost. Under the sustainable boundary constraint $\left(SoC(T+H) \geq SoC(0) = 6kWh\right)$, the exact global lower bound for the nominal case (S1) is 37.07 USD/week. Note that the earlier offline figure of 36.981 USD/week reported in [3] resulted from an open-terminal formulation where the battery was allowed to deplete fully from 6 kWh to 2 kWh by the end of the week, artificially deflating electricity purchases. By enforcing cyclic energy neutrality, the present formulation guarantees continuous, sustainable operation. The RHO-EDBO controller reaches 37.32 USD/week on S1—within 0.67% of this exact bound—and remains within 0.6–2.2% across all scenarios. The absence of simultaneous charging and discharging in the LP solution further confirms that the formulation is exact rather than a relaxation.

7.1.2 Implications for practical EMS applications

The conclusion above is specific to the convex base model. In practical battery energy management systems, nonlinear characteristics such as power-dependent charge/discharge efficiency, for example: $\left(\eta = 0.898 - \frac{0.173 \cdot |\Delta(t)|}{P_{rated}}\right)$ [28], degradation-aware cost functions, or discrete operating modes are often considered. These features introduce nonlinearities and may render the optimization problem non-convex, preventing an exact LP reformulation. In such cases, gradient-free meta-heuristics remain attractive because they can optimize the original model directly without requiring problem-specific linearization or reformulation. Therefore, while LP is preferred for the convex benchmark considered in this work, the EDBO framework provides a more general optimization engine that can be extended to future non-convex EMS formulations.

7.2 Software-in-the-loop validation of the RHO-EDBO controller

This session evaluates the performance of the proposed RHO-EDBO controller in the SIL environment. Table 5 summarizes the results across all test scenarios, while Fig. 4 compares the corresponding weekly operating costs.

Table 4: Deterministic LP global optimum versus RHO - EDBO on the convex base model

Scenario	Base (USD/wk)	LP optimum (USD/wk)	RHO-EDBO (USD/wk)	Gap to LP (%)	LP Solve (ms)
S1 nominal	40.01	37.07	37.32	0.67	24.6
S2 low-PV	50.29	47.21	47.51	0.64	22.1
S3 PV surplus	16.01	13.41	13.70	2.16	24.5
S4 price spike	56.24	39.03	39.79	1.95	23.8
S5 load step	45.14	42.14	42.41	0.64	20.8
S6 dropout	40.01	37.07	37.82	2.02	21.0

Table 5: SIL results over Modbus TCP across the scenario matrix

Scenario	Base (USD/wk)	RHO-EDBO (USD/wk)	Save (%)	Solve mean (ms)	Comm mean (ms)	Miss / drop / viol.
S1 nominal	40.01	37.32 ± 0.10	6.72 ± 0.25	449	0.47	0% / 0 / 0
S2 low-PV	50.29	47.51 ± 0.04	5.54 ± 0.09	446	0.48	0% / 0 / 0
S3 PV surplus	16.01	13.70 ± 0.04	14.42 ± 0.25	452	0.49	0% / 0 / 0
S4 price spike	56.24	39.79 ± 0.22	29.25 ± 0.40	447	0.51	0% / 0 / 0
S5 load step	45.14	42.41 ± 0.09	6.05 ± 0.20	445	0.48	0% / 0 / 0
S6 dropout	40.01	37.82 ± 0.19	5.48 ± 0.49	446	0.52	0% / 20 / 0

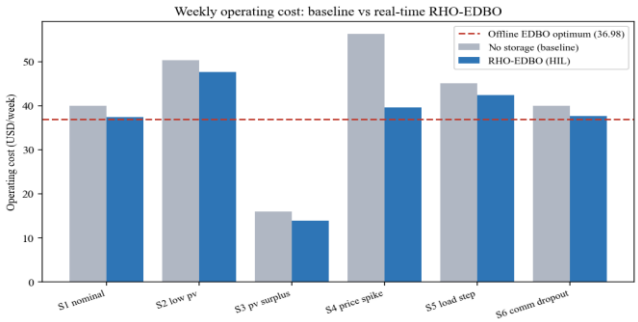


Figure 4: Weekly operating cost: no-storage baseline vs real-time RHO-EDBO

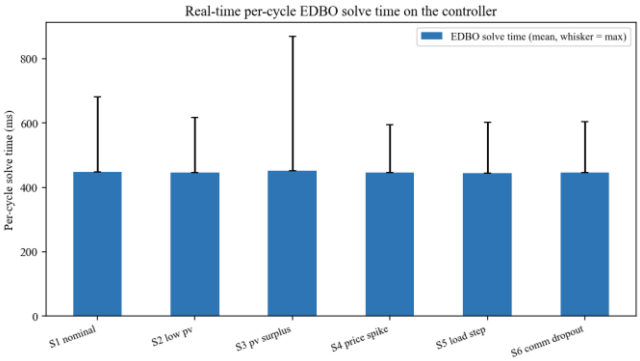


Figure 5: Per-cycle EDBO solve time across scenarios (mean, whisker to maximum)

7.2.1 Real-time feasibility

The mean per-cycle solve time was about 0.45 s (mean) (Table 6, Fig. 5), with a worst case of 0.869 s (Table 7), still under 1 s and well within the 2-second deadline, and a 0% deadline-miss rate in every run (Table 7), and mean round-trip communication latency below 1ms (Table 5). By contrast, a single offline solve of the full 168-variable problem took roughly 105 s - three orders of magnitude beyond any real-time cycle - confirming that the receding-horizon reformulation, not merely the choice of hardware, is what makes deployment feasible. Table 6 contrasts the two computational profiles. Although the receding-horizon controller performs more objective evaluations in aggregate over the week (about 1.0×10^6 versus 5.25×10^5), each individual solve is small, low-dimensional and bounded, so it fits comfortably within a control cycle; the offline approach concentrates all its effort into one large, high-dimensional solve that cannot be interrupted or bounded to a deadline.

Table 6: Computational profile: offline solve versus per-cycle real-time solve

Quantity	Offline	RHO - EDBO (per cycle)
Decision dimension	168	48
Population × iterations	35 × 15,000	30 × 200
Evaluations per solve	5.25×10^5	6.0×10^3
Number of solves per week	1	168
Wall time per solve	≈ 105 s	≈ 0.45 s
Bounded to a deadline?	no	yes

7.2.2 Optimization quality

On the nominal week the controller reached 37.32 ± 0.10 USD/week over five seeds, a 6.72% reduction against the baseline (Table 5). This is within 0.67% of the exact linear-programming global optimum of 37.07 USD/week (Table 4, Section 7.1.1), confirming that the real-time controller recovers essentially all of the economic benefit while

operating online, under an embedded budget and without full future knowledge, the small residual gap being the expected price of real-time operation. The gap is small for two reasons. The receding-horizon controller replaces one 168-dimensional optimization with a sequence of easier 48-dimensional problems, each anchored by the true measured state of charge; and the daily periodicity of the data means a 48-hour window already captures the exploitable structure. The temporal decomposition thus trades negligible global optimality for large gains in per-solve tractability and, through feedback, for immunity to open-loop error accumulation.[3]

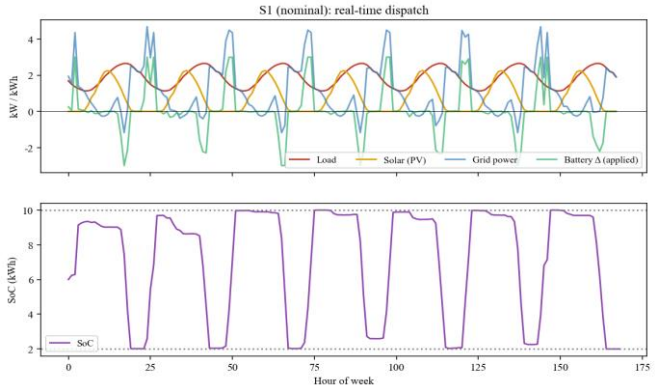


Figure 6: Real-time dispatch (S1): load, PV, grid power and applied battery command (top); battery SoC (bottom).

The per-scenario results (Table 5, with weekly costs compared in Fig. 4) are economically sensible. The largest savings occurred under the evening price spike (29.25%), where arbitrage is most valuable, and under PV surplus (14.42%). The corresponding dispatch (Fig. 6) shows the battery charging during the midday solar peak and discharging into the evening load/price peak, with the state of charge respecting its [2, 10] kWh limits throughout. Savings are smallest under low PV (S2, 5.54%), where little surplus is available to arbitrage, and largest under the evening price spike (S4, 29.25%), where the controller pre-charges and discharges through the peak, exactly the behavior the objective rewards; PV surplus (S3, 14.42%) and the load step (S5, 6.05%) fall in between, and the communication-dropout scenario (S6) retains a 5.48% saving despite twenty lost commands, about 1.2 points below nominal, with zero constraint violations throughout. Across all scenarios the mean per-cycle solve time stays in a narrow 0.44–0.45 s band, so the computational load is essentially independent of operating condition, important for guaranteeing the real-time deadline in the field.

7.2.3 Statistical characterization and the role of warm-starting

Because EDBO is stochastic, each scenario was repeated over five independent random seeds; Table 5 reports the mean and standard deviation of the realized weekly cost and saving, and Table 7 the corresponding per-cycle solve-time statistics (mean, 99th percentile and maximum). The standard deviation of the weekly cost is at most 0.22 USD/week, corresponding to less than 0.55% of the mean cost across all scenarios (the worst case of S4 scenario in Table 5), so the controller's performance is reproducible rather than fortuitous.

The one per-cycle acceleration actually employed, warm-starting from the previous window's shifted plan, was ablated on the nominal scenario over the same five seeds (Table 8). Disabling it raises the mean weekly cost from 37.32 to 39.12 USD/week (a 4.6% relative increase, the saving falling from 6.7% to 2.2%) at essentially unchanged solve time, because a cold-started population rarely converges to a comparable plan within the fixed 200-iteration budget. Warm-starting is therefore the single most consequential real-time provision. Feasibility pruning enforces the battery constraints by rejecting infeasible commands. Consistent with this design, no constraint violations were observed in any scenario (Table 5).

Table 7: Per-cycle solve-time statistics of the receding-horizon EDBO controller

Scenario	Solve mean (ms)	Solve p99 (ms)	Solve max (ms)	Deadline miss
S1 nominal	449	656	682	0%
S2 low-PV	446	550	618	0%
S3 PV surplus	452	775	869	0%
S4 price spike	447	544	595	0%
S5 load step	445	539	603	0%
S6 dropout	446	548	604	0%

Table 8: Warm-start ablation on the nominal scenario (mean ± standard deviation, five seeds)

Warm-start	Cost (USD/wk)	Saving (%)	Solve mean (ms)
Enabled	37.32 ± 0.10	6.72	442
Disabled	39.12 ± 0.07	2.23	440

7.2.4 Robustness

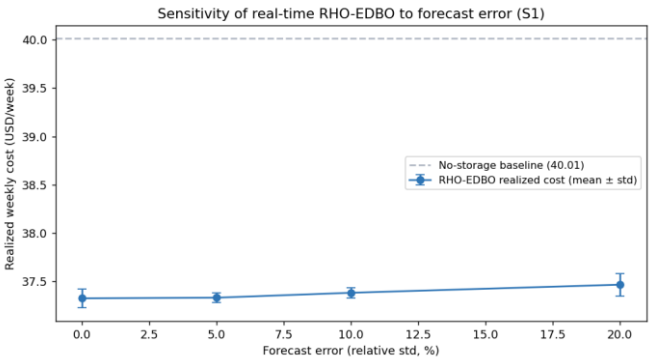


Figure 7: Sensitivity of realized weekly cost to forecast error (S1)

Table 9: Realized weekly cost under increasing forecast error (S1, mean ± std over five seeds)

Forecast error	Realized cost (USD/week)	Saving vs baseline (%)
0% (perfect foresight)	37.32 ± 0.10	6.72
5%	37.33 ± 0.05	6.70
10%	37.38 ± 0.05	6.58
20%	37.46 ± 0.12	6.37

To assess forecast robustness, relative noise was added to the load and PV forecasts supplied to the controller, while the plant operated with the original profiles. As summarized in Table 10 and illustrated in Fig. 7, increasing the forecast error from 0% to 20% raises the weekly operating cost only from 37.32 to 37.46 USD/week, reducing the saving from 6.72% to 6.37%. This limited performance degradation indicates that the proposed controller is robust to forecast uncertainty. The main reason is the receding-horizon strategy, which re-optimizes every hour using updated measurements, preventing forecast errors from accumulating over time.

7.2.5 Comparison with the offline-study optimizers as the real-time solver

To position the proposed controller against the optimizers evaluated in the original offline study, DBO, HHO, RUN, SMA, and AO were used as the receding-horizon solver under identical settings (H = 48 h, population = 30, iterations = 200, warm-start enabled), each over the same five random seeds as Table 5. The corresponding results are summarized in Table 10 and illustrated in Fig. 8. EDBO achieved the lowest weekly operating cost (37.32 ± 0.10 USD/week), followed by DBO (37.47 ± 0.04 USD/week) and RUN (37.62 ± 0.21 USD/week), reproducing the ranking observed in the offline study.

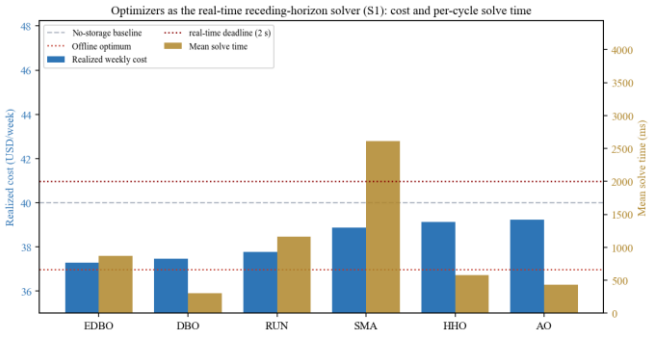


Figure 8: The offline-study optimizers as the real-time receding-horizon solver (S1)

Table 10: The offline-study optimizers evaluated as the real-time receding-horizon solver

Optimizer	Cost (USD/wk)	Saving (%)	Solve mean (ms)	Solve max (ms)	Deadline miss (%)
EDBO (proposed)	37.32 ± 0.10	6.72 ± 0.25	449 ± 6	682	0
DBO	37.47 ± 0.04	6.35 ± 0.10	281 ± 1	411	0
RUN	37.62 ± 0.21	5.98 ± 0.54	1084 ± 19	1372	0
SMA	38.55 ± 0.15	3.67 ± 0.37	2525 ± 25	3158	81
HHO	39.11 ± 0.06	2.27 ± 0.14	528 ± 10	692	0
AO	39.23 ± 0.03	1.95 ± 0.07	417 ± 2	522	0

Beyond solution quality, the real-time setting introduces a computational constraint that is absent from offline optimization. As shown in Fig. 8, most optimizers satisfied the 2 s deadline, whereas SMA exceeded it on 81% of control cycles (mean solve time 2.53 s) despite achieving a lower cost than HHO and AO. Consequently, SMA cannot be considered real-time deployable under the tested conditions.

8. Conclusion and future work

This paper presented a real-time receding-horizon implementation of the Enhanced Dung Beetle Optimization algorithm for residential smart-grid energy management and validated it in a software-in-the-loop environment using a real Modbus TCP communication link. The study demonstrated that the proposed controller can operate reliably within a real-time control framework while preserving the economic benefits previously achieved in offline optimization. The results showed that the controller consistently met the real-time deadline, achieved near-optimal operating costs, and remained robust under communication disturbances and forecast uncertainty. A solver-selection analysis further showed that linear programming is the preferred approach for the convex EMS formulation considered in this work, whereas meta-heuristic optimization becomes attractive when practical nonlinear characteristics are introduced. Future work will extend the proposed framework to embedded Controller-HIL implementation, Power-HIL validation with physical hardware, and more realistic non-convex EMS formulations incorporating battery and converter nonlinearities.

ACKNOWLEDGMENTS

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number MDT 107.01-2025.12.

REFERENCES

[1] E. H. Houssein, M. K. Saeed, G. Hu, and M. M. Al-Sayed, "Metaheuristics for Solving Global and Engineering Optimization Problems: Review, Applications, Open Issues and Challenges," *Archives of Computational Methods in Engineering*, vol. 31, no. 8, pp. 4485–4519, 2024, doi: 10.1007/s11831-024-10168-6.

[2] A. Das, R. Rai, B. Sasmal, K. G. Dhal, R. A. Khurma, and R. Saha, "Metaheuristic Algorithms Since 2020: Development, Taxonomy, Analysis, and Applications," *Archives of Computational Methods in Engineering*, 2025, doi: 10.1007/s11831-025-10408-3.

[3] H. T. Vy and H. P. H. Anh, "Enhanced Dung Beetle Optimization Algorithm Combined Harris Hawks Optimizer and Nelder Mead Simplex Method for Solar-Connected Smart-grid Application," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, 2025, [Online]. Available: https://api.semanticscholar.org/CorpusID:284498245

[4] J.-K. Xue and B. Shen, "Dung beetle optimizer: a new meta-heuristic algorithm for global optimization," *J. Supercomput.*, vol. 79, Nov. 2022, doi: 10.1007/s11227-022-04959-6.

[5] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications,"

Future Generation Computer Systems, vol. 97, pp. 849–872, 2019, doi: https://doi.org/10.1016/j.future.2019.02.028.

[6] J. E. Dennis and W. J. E., "Optimization on Microcomputers: The Nelder-Mead Simplex Algorithm," 1985. [Online]. Available: https://api.semanticscholar.org/CorpusID:61018648

[7] X. Liu, J.-S. Wang, S.-B. Zhang, X.-Y. Guan, and Y.-Z. Gao, "Optimization Scheduling of Off-grid Hybrid Renewable Energy Systems Based on Dung Beetle Optimizer with Convergence Factor and Mathematical Spiral," *Renew. Energy*, vol. 237, p. 121874, Nov. 2024, doi: 10.1016/j.renene.2024.121874.

[8] F. Zhu, G. Li, H. Tang, Y. Li, X. Lv, and X. Wang, "Dung beetle optimization algorithm based on quantum computing and multi-strategy fusion for solving engineering problems," *Expert Syst. Appl.*, vol. 236, p. 121219, 2024, doi: https://doi.org/10.1016/j.eswa.2023.121219.

[9] W. Tang, H. Qin, and S. Kang, "Dung beetle optimizer based on mean fitness distance balance and multi-strategy fusion for solving practical engineering problems," *Sci. Rep.*, vol. 15, no. 1, p. 26389, 2025, doi: 10.1038/s41598-025-02937-5.

[10] M. O. Faruque *et al.*, "Real-Time Simulation Technologies for Power Systems Design, Testing, and Analysis," *IEEE Power and Energy Technology Systems Journal*, vol. 2, pp. 63–73, Sep. 2015, doi: 10.1109/JPETS.2015.2427370.

[11] C. Dufour and J. Bélanger, *On the Use of Real-Time Simulation Technology in Smart Grid Research and Development*, vol. 50. 2013. doi: 10.1109/ECCE.2013.6647090.

[12] J. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. 2017.

[13] W. Cai, A. B. Kordabad, and S. Gros, "Energy management in residential microgrid using model predictive control-based reinforcement learning and Shapley value," *Eng. Appl. Artif. Intell.*, vol. 119, p. 105793, 2023, doi: https://doi.org/10.1016/j.engappai.2022.105793.

[14] H. T. Vy and H. P. H. Anh, "Application of Dung Beetle Optimizer Algorithm for Solving Optimization Problem in Smart Residential Power Grid," in *Computational Intelligence Methods for Green Technology and Sustainable Development*, Y.-P. Huang, W.-J. Wang, H.-G. Le, and A.-Q. Hoang, Eds., Cham: Springer Nature Switzerland, 2024, pp. 148–159.

[15] Y. Yue, H. Ren, D. Liu, and L. Zhang, "Optimal Scheduling of Microgrids Based on an Improved Dung Beetle Optimization Algorithm," *Applied Sciences*, vol. 15, no. 2, 2025, doi: 10.3390/app15020975.

[16] Y. Chen, W. Ning, and W. Du, "A study of scheduling strategies for microgrids based on the non-dominated sorting dung beetle optimization algorithm," *Sci. Rep.*, vol. 15, May 2025, doi: 10.1038/s41598-025-02446-5.

[17] W. Gu and F. Wang, "A multi-strategy improved dung beetle optimisation algorithm and its application," *Cluster Comput.*, vol. 28, no. 1, p. 49, 2024, doi: 10.1007/s10586-024-04704-z.

[18] Y. Zhang, T. Li, T. Ma, D. Yang, and X. Sun, "Short-Term Photovoltaic Power Prediction Based on Extreme Learning Machine with Improved Dung Beetle Optimization Algorithm," *Energies (Basel).*, vol. 17, no. 4, 2024, doi: 10.3390/en17040960.

[19] R. Fang, T. Zhou, B. Yu, Z. Li, L. Ma, and Y. Zhang, "Dung Beetle Optimization Algorithm Based on Improved Multi-Strategy Fusion," *Electronics (Basel).*, vol. 14, no. 1, 2025, doi: 10.3390/electronics14010197.

[20] H. P. H. Vy Huynh Tuyet and Anh, "Performance Evaluation of Dung Beetle Optimizer on Smart Home Energy Management," in *Advanced Signal Processing Systems*, T. Le Tien, Ed., Cham: Springer Nature Switzerland, 2026, pp. 261–272.

[21] A. Grimaldi, F. D. Minuto, J. Brouwer, and A. Lanzini, "Profitability of energy arbitrage net profit for grid-scale battery energy storage considering dynamic efficiency and degradation using a linear, mixed-integer linear, and mixed-integer non-linear optimization approach," *J. Energy Storage*, vol. 95C, p. 112380, Aug. 2024, doi: 10.1016/j.est.2024.112380.

[22] I. Toure, A. Payman, M.-B. Camara, and B. Dakyo, "Energy Management in a Renewable-Based Microgrid Using a Model Predictive Control Method for Electrical Energy Storage Devices," *Electronics (Basel).*, vol. 13, no. 23, 2024, doi: 10.3390/electronics13234651.

[23] Modbus Organization, "MODBUS Application Protocol Specification V1.1b3," online: https://www.modbus.org/file/secure/modbusprotocolspecification.pdf.

[24] A. Boukaibat, N. Krami, Y. Rochdi, Y. El Bakkali, M. Laamim, and A. Rochd, "Real-Time Energy Management in Microgrids: Integrating T-Cell Optimization, Droop Control, and HIL Validation with OPAL-RT," *Energies (Basel).*, vol. 18, no. 15, 2025, doi: 10.3390/en18154035.

[25] NREL, "Data of solar energy from National Renewable Energy Laboratory (NREL), USA," [Online] Available at: https://www.nrel.gov/grid/solar-resource/renewable-resource-data.html.

[26] NREL, "Data of load demand from National Renewable Energy Laboratory (NREL), USA," [Online] Available at: https://data.openei.org/submissions/153 [Accessed 2023].

[27] U.S.Bureau of Labor Statistics, "Data for buying electricity price ," [Online] https://www.bls.gov/regions/midwest/data/averageenergyprices_selectedareas_table.htm.

[28] H. T. Vy and H. P. H. Anh, "Optimal Renewable Energy Management System Using Hybrid Evolutionary Neural Q-Learning Technique," *Journal of Control, Automation and Electrical Systems*, 2025, doi: 10.1007/s40313-025-01192-3.