Article

Not peer-reviewed version

AI-Driven Virtual Power Plant Scheduling: CUDA-Accelerated Parallel Simulated Annealing Approach

Ali Abbasi [*,†] , João L. Sobral [†] , Ricardo Rodrigues [†,‡]

Posted Date: 10 October 2025

doi: 10.20944/preprints202510.0758.v1

Keywords: CUDA-accelerated computing; parallel simulated annealing; virtual power plants; energy scheduling; optimization algorithms

Article

AI-Driven Virtual Power Plant Scheduling: CUDA-Accelerated Parallel Simulated Annealing Approach

Ali Abbasi [1,2,*,†], **João L. Sobral** [2,†] and **Ricardo Rodrigues** [1,†,‡]

[1] DTx—Digital Transformation CoLAB, University of Minho, 4800-058 Guimarães, Portugal
[2] Centro de Algoritmi, Universidade do Minho, Campus of Gualar, 4704-553 Braga, Portugal
[*] Correspondence: ali.abbasi@dtx-colab.pt
[†] These authors contributed equally to this work.
[‡] Member of IEEE.

Highlights

The rapid expansion of DERs in urban environments demands real-time, scalable scheduling mechanisms that can efficiently coordinate the operation of VPPs under dynamic and constraint-laden conditions. Within the context of smart cities, where energy systems must respond quickly to variable renewable generation and market fluctuations, traditional optimization approaches often fail to meet the required performance thresholds. This study presents a high-performance scheduling framework based on a GPU-accelerated Multiple-Chain Simulated Annealing (MC-SA) algorithm, specifically designed to address the computational challenges of large-scale VPP optimization. The method introduces two complementary levels of parallelism: (1) a divide-and-conquer decomposition across prosumers to exploit the independence of scheduling subproblems, and (2) a multi-chain SA strategy that tackles the inherently sequential nature of classical Simulated Annealing by launching independent search trajectories across the solution space. The framework is implemented using CUDA and evaluated on large-scale instances, demonstrating its potential to enable reliable, real-time VPP scheduling in smart city energy systems.

What are the main findings?

- A GPU-accelerated MC-SA algorithm was developed, enabling large-scale VPP scheduling by combining problem-level and algorithm-level parallelization.
- At the problem level, a divide-and-conquer strategy partitions the global scheduling task across prosumers, allowing parallel processing of independent subproblems on GPU threads.
- At the algorithm level, the framework overcomes the sequential limitations of classical Simulated Annealing by deploying multiple independent annealing chains in parallel, enabling broader exploration of the solution space and faster convergence.
- The proposed MC-SA method demonstrates substantial computational speedups, achieving up to 10$\times$ improvement over single-chain GPU execution and up to 38$\times$ over a 16-thread CPU implementation, without compromising the feasibility or quality of solutions.
- The scheduling framework successfully scales to systems with up to 1000 prosumers and 96 time intervals (15-minute resolution), delivering sub-second runtime performance suitable for real-time operation.
- A projection-based feasibility operator is integrated to ensure all candidate schedules satisfy operational constraints, including battery state-of-charge dynamics, import/export limits, and mutual exclusivity of charging and discharging decisions.

What is the implication of the main findings?

- The proposed MC-SA framework enables practical, real-time scheduling of Virtual Power Plants in smart city environments, supporting responsive coordination of DERs, energy storage, and flexible demand under market and network constraints.

- By combining divide-and-conquer decomposition with multi-chain annealing, the method transforms a sequential metaheuristic into a highly parallelizable architecture compatible with modern GPU hardware.
- The successful parallelization of Simulated Annealing, despite its inherently sequential structure, demonstrates a novel and effective strategy for scaling metaheuristic optimization to real-world energy systems.
- The ability to process high-resolution schedules across hundreds or thousands of prosumers in sub-second time frames aligns directly with the operational demands of intraday energy markets and grid-aware resource dispatch.
- The generality of the approach makes it suitable for broader deployment in smart city infrastructures where computationally intensive decision-making is required, offering a transferable solution for energy scheduling, distributed optimization, and real-time control.

Abstract

Efficient scheduling of Virtual Power Plants (VPPs) is essential for the integration of distributed energy resources into modern power systems. This study presents a CUDA-accelerated multiple-chain simulated annealing (MC-SA) algorithm tailored for optimizing VPP scheduling. Traditional simulated annealing algorithms are inherently sequential, limiting their scalability for large-scale applications. The proposed MC-SA algorithm mitigates this limitation by executing multiple independent annealing chains concurrently, enhancing the exploration of the solution space and reducing the requisite number of sequential cooling iterations. The algorithm employs a dual-level parallelism strategy: at the prosumer level, individual energy producers and consumers are assessed in parallel; at the algorithmic level, multiple simulated annealing chains operate simultaneously. This architecture not only expedites computation but also improves solution accuracy. Experimental evaluations demonstrate that the CUDA-based MC-SA achieves substantial speedups—up to 10× compared to a single-chain baseline implementation while maintaining or enhancing solution quality. Our analysis reveals an empirical power-law relationship between parallel chains and required sequential iterations (iterations $\propto$ chains$^{-0.88\pm0.17}$), demonstrating that increased parallelism effectively alleviates the sequential bottleneck. The algorithm demonstrates scalable performance across VPP sizes from 250 to 1000 prosumers, with approximately 50 chains providing the optimal balance between solution quality and computational efficiency for practical applications.

Keywords: CUDA-accelerated computing; parallel simulated annealing; virtual power plants; energy scheduling; optimization algorithms

---

1. Introduction

The integration of distributed energy resources (DERs)—including photovoltaics, wind turbines, battery storage systems, controllable loads, and distributed generation—is central to the advancement of modern electricity markets. Virtual Power Plants (VPPs), which aggregate and coordinate these diverse resources, have become essential for optimizing participation in electricity markets, including day-ahead (DA), intraday (ID), and real-time (RT) trading environments [1]. The primary objective of a VPP is to maximize economic value while ensuring reliable operation, a task complicated by significant uncertainties such as renewable generation variability and fluctuating market prices [4,10].

To address these challenges, the literature presents a rich landscape of optimization frameworks for VPP scheduling. Mixed-Integer Linear Programming (MILP) and Mixed-Integer Quadratic Programming (MIQP) dominate exact optimization approaches due to their capacity to simultaneously model discrete bidding actions and continuous power dispatch decisions. These models are widely used for day-ahead bidding and intra-day operation [7], as well as for providing multiple grid support services [2]. Recent research has also developed dual-MILP approaches to tackle the complexities of real-time energy market bidding [9].

To handle uncertainty, these deterministic models are extended into stochastic programming formulations, which use scenario trees to anticipate a range of possible futures, thereby enhancing decision robustness. This is particularly relevant for wind-storage systems participating in electricity markets [3] and for risk-averse VPP operations using rolling horizon control [10]. Alternatively, robust optimization techniques provide performance guarantees under worst-case operational deviations, offering a different approach to managing uncertainty [5].

A critical aspect of modern VPP modeling is the sophisticated representation of energy storage systems. Advanced models capture state-of-charge dynamics, degradation costs, and round-trip efficiency to support accurate and economically realistic scheduling [8,14]. When these storage assets are co-optimized with flexible demand-side resources, the VPP can deliver a wider array of market services and significantly enhance system-level reliability and profitability. For instance, optimal scheduling with demand response in short-term markets has been shown to improve economic outcomes [13]. Furthermore, coordinated operation strategies for VPPs comprising multiple DER aggregators have been developed to improve overall coordination and efficiency [6].

However, the high fidelity of these models comes at a significant computational cost. The complexity of solving large-scale, mixed-integer, and stochastic problems often creates a bottleneck for practical deployment. In real-world operations, VPPs must often rely on rolling-horizon control with rescheduling intervals as short as 15 minutes [12], requiring optimization solutions under tight time constraints. This has motivated research into accelerating these computations. Multi-core CPU-based parallelization techniques have been adopted to speed up MILP solving and increase scheduling responsiveness [1,2], and multistage scheduling models under distributed locational marginal prices have been proposed to enhance scalability [11]. Despite these efforts, current literature reveals a clear gap in practical implementations of GPU-accelerated scheduling for VPPs [15], even though GPU architectures have demonstrated superior performance for many other large-scale parallel optimization tasks.

Given the computational limitations of exact methods, metaheuristic algorithms such as Simulated Annealing (SA) offer scalable and flexible alternatives. Their ability to handle non-convex and complex constraint spaces makes them suitable for large-scale VPP problems. Recent advances using high-performance computing (HPC) have demonstrated the potential of parallel SA frameworks, achieving substantial reductions in computational time for VPP scheduling and enabling the management of a large number of consumers in near-real-time [42]. However, these implementations typically leverage multi-core CPUs, leaving the immense parallel processing power of modern GPUs largely untapped for this specific problem. The inherently sequential nature of the classical SA algorithm, where each state transition depends on the previous one, poses a significant challenge to its efficient parallelization on many-core architectures.

1.1. Research Contribution and Gap Addressing

This study directly addresses these research gaps through several key contributions. First, to overcome the scalability limitations of exact optimization methods for large-scale VPPs, we develop a metaheuristic approach based on Simulated Annealing that offers flexible constraint handling and polynomial-time complexity. Second, to mitigate the inherent sequential bottleneck of traditional SA, we introduce a novel multiple-chain architecture that leverages massive GPU parallelism. Third, we address the gap in practical GPU implementations for VPP scheduling by designing a customized CUDA kernel with a two-level parallelism strategy across prosumers and annealing chains. Finally, our work provides empirical validation of the parallelism-iteration trade-off through rigorous statistical analysis, offering practitioners a principled approach to algorithm tuning that has been lacking in prior literature.

To address the evolving complexities of modern distributed energy systems, this study introduces a VPP optimization framework engineered for real-time responsiveness, economic efficiency, and operational resilience. Designed as a scalable, data-centric architecture, the framework harmonizes prosumer behavior with grid-level objectives while respecting system-wide technical and economic constraints. The development of this system is guided by four core design principles.

1) Rolling-Horizon Social Welfare Optimization: The framework prioritizes social welfare by dynamically scheduling energy transactions based on real-time supply, demand, and grid conditions. Through a rolling-horizon approach, the model adapts to market fluctuations and reformulates the objective as a cost minimization task—accounting for battery degradation, efficiency losses, and operational limits—to ensure holistic system optimization.

2) Scalable Architecture: As prosumer participation increases, the framework must maintain computational efficiency and scheduling precision. Its design supports high-dimensional optimization, enabling robust performance at scale without compromising real-time execution or decision quality.

3) Battery Longevity and Sustainability: To promote long-term viability, the system integrates battery health constraints that govern charge and discharge operations within safe boundaries. This preserves battery lifespan while supporting environmentally sustainable energy practices.

4) Grid-Regulated Market Coordination: Operating under centrally determined price signals, the system simplifies market participation by requiring prosumers to submit only energy quantities, while the grid defines the pricing structure. This ensures transparent, price-aligned scheduling without strategic bidding complexity.

Collectively, these design elements necessitate an integrated architecture capable of real-time forecasting, high-frequency data streaming, and adaptive control. The proposed VPP management platform meets these demands, delivering intelligent, market-aligned scheduling of distributed resources. By leveraging predictive analytics and decentralized coordination, it enhances system welfare, ensures operational agility, and supports large-scale deployment across academic research and commercial applications.

Figure 1 illustrates the conceptual architecture of the proposed framework, capturing the flow of real-time data, the interaction between prosumers and the grid, and the role of the centralized optimization engine in governing distributed energy scheduling.

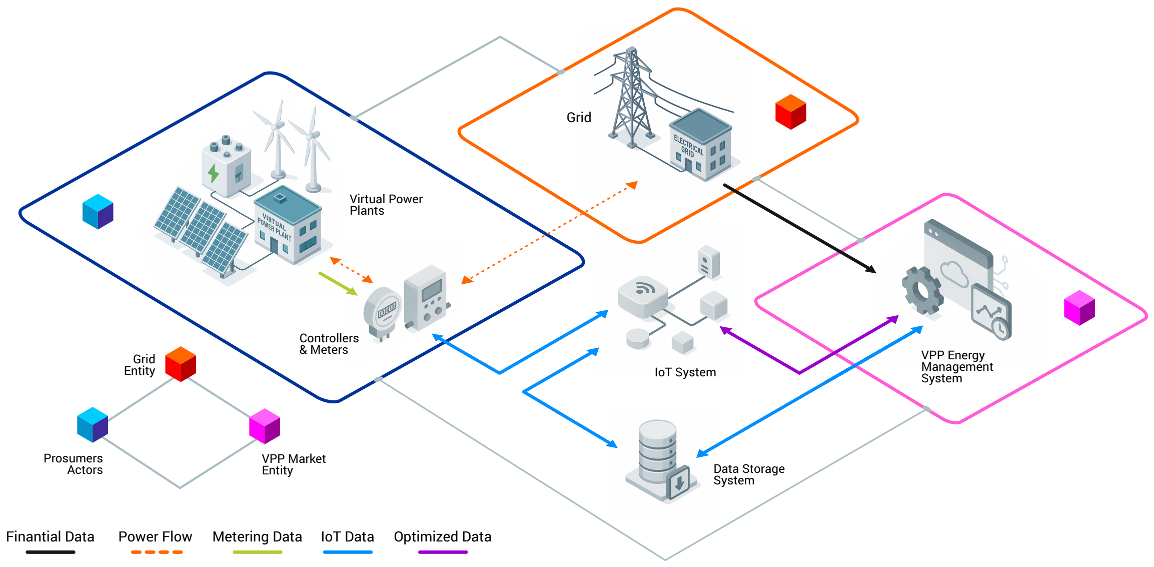

Figure 1. Schematic of the VPP management framework, illustrating key components and data flows among prosumers, the market, and the grid. The VPP acts as an intermediary, enabling energy trading within regulated prices, with a centralized scheduler optimizing resource allocation.

To contextualize these design choices, we present a conceptual scenario that encapsulates the operational dynamics of the VPP environment. This scenario demonstrates how DERs are orchestrated within a grid-regulated pricing structure. As shown in **Figure 1**, the VPP functions as an intelligent intermediary, coordinating transactions among prosumers and aligning them with grid-defined economic signals.

Prosumers continuously transmit real-time data—encompassing generation, consumption, and battery status—through an IoT-enabled data streaming layer that forms the system's communication backbone. At the core of the VPP is an energy management system comprising two critical modules: a forecasting engine and an optimization scheduler. The former predicts individual prosumer energy behavior using real-time and historical data, while the latter allocates optimal energy transactions (buy, sell, store) under grid-imposed constraints. By synchronizing these operations, the VPP enhances energy market efficiency, ensures supply-demand balance, and reinforces overall grid reliability.

Given these design requirements and the real-time decision-making nature of the problem, achieving timely and scalable scheduling for large prosumer communities necessitates a HPC infrastructure. The computational complexity arising from rolling-horizon optimization, real-time forecasting, and constraint handling becomes increasingly demanding as the number of participating prosumers grows. Therefore, an HPC-enabled solution is essential to ensure that the VPP management system can operate efficiently at scale and deliver near real-time scheduling performance.

The remainder of this paper is organized as follows. Section 2 reviews relevant literature on VPP optimization, real-time scheduling, and parallel metaheuristics. Section 3 introduces the VPP optimization model and problem formulation. In Section 4, we present a prosumer-level decomposition strategy that enables parallelization of the scheduling problem. Section 5 presents the MC-SA architecture developed to enhance exploration efficiency, followed by a description of our hierarchical parallelization strategy for distributing computations across HPC resources. Section 5.9 discusses the practical implementation of the proposed approach on HPC infrastructure. Section 6 presents numerical results and performance evaluations, and finally, Section 7 concludes the paper with key insights and directions for future research.

2. Related Works

2.1. Optimization Algorithms for Energy Systems Scheduling in Virtual Power Plants

The effective coordination of Distributed Energy Resources within Virtual Power Plants necessitates sophisticated optimization algorithms capable of handling complex constraints, multiple objectives, and various sources of uncertainty. This section provides a comprehensive review of optimization strategies applied to VPP scheduling, systematically categorizing them by computational paradigm and analyzing their respective strengths and limitations in addressing the unique challenges of modern energy systems.

Swarm Intelligence and Population-Based Methods have demonstrated remarkable success in solving the complex, non-convex optimization problems inherent to VPP operations. Ant Colony Optimization (ACO) has emerged as a particularly powerful approach for distributed scheduling and market participation strategies. [17] pioneered the application of ACO in local electricity markets, developing a bi-level formulation that enabled decentralized agents to learn optimal bidding strategies while preserving privacy and ensuring profitability. This approach demonstrated superior performance compared to centralized evolutionary methods in complex market environments. The robustness of ACO for microgrid resource scheduling was further validated by [18], who achieved significant cost reductions while maintaining system reliability over 24-hour scheduling horizons. For addressing complex power flow constraints in unbalanced distribution networks, [19] introduced an innovative hybrid Tabu Continuous Ant Colony Search (TCACS) algorithm that reduced power losses by 16.53% while generating substantial daily profits. The versatility of ACO extends beyond traditional power system applications, as evidenced by [20]'s work on virtual network embedding for data center energy management, which achieved a remarkable 52% energy reduction. Furthermore, [21] successfully adapted ACO for the computationally challenging problem of scheduling step-controlled generators in smart grids, developing simulation-guided graph construction techniques that maintained solution feasibility while navigating the NP-hard complexity of the problem.

Simulated Annealing and Evolutionary Computation Techniques have proven equally valuable for VPP optimization, particularly in scenarios requiring global optimization capabilities and handling of complex constraint structures. The thermodynamic principles underlying Simulated Annealing have been effectively leveraged for joint optimization of energy and ancillary services. [22] developed an SA-based methodology for VPPs integrated with electric vehicles, achieving near-optimal solutions with a remarkable 99.94% reduction in computational time compared to traditional deterministic models. For communication resource allocation in VPPs, [23] proposed an Improved Simulated Annealing algorithm incorporating adaptive temperature control and multi-neighborhood search mechanisms, which demonstrated superior performance over both Genetic Algorithms and Particle Swarm Optimization in terms of communication efficiency, latency reduction, and energy utilization. Genetic Algorithms have been extensively applied to VPP scheduling problems, with [24] demonstrating their effectiveness for Demand Side Management in VPPs integrating diverse DERs and Demand Response programs, successfully reducing electricity market prices and mitigating peak loads through intelligent resource coordination. [25] further extended GA applications to residential VPPs with electric vehicle integration, achieving substantial cost reductions and enhanced ancillary service delivery through sophisticated chromosome encoding and fitness evaluation mechanisms.

Hybrid and Multi-Objective Optimization Frameworks represent a significant advancement in addressing the competing objectives and complex constraint interactions characteristic of modern VPP operations. [26] developed an innovative three-stage VPP management system that synergistically combined an Improved Pelican Optimization Algorithm for DER scheduling, a convolutional autoencoder for cyberattack detection (achieving 98.06% accuracy), and Prophet forecasting for market price prediction, collectively resulting in a 15.3% revenue enhancement. For managing the inherent uncertainties associated with renewable generation and load patterns under vehicle-to-grid operation, [27] employed sophisticated multi-objective optimization techniques using Genetic Algorithms, effectively balancing economic and reliability objectives. [29] incorporated data-driven stochastic robust optimization with GA frameworks for handling multiple uncertainty sources in VPPs, leveraging real-time consumption data to enhance decision robustness. Two-stage optimization architectures have demonstrated particular efficacy, with [30] presenting a comprehensive framework integrating flexible Carbon Capture Systems with Virtual Hybrid Energy Storage, yielding an impressive 10.12% improvement in carbon efficiency and 8.91% gain in economic performance. Similarly, [31] developed an advanced two-stage model that effectively balanced cost minimization and emission reduction objectives under dynamic demand and supply conditions, demonstrating the scalability of hybrid approaches for practical VPP applications.

Comparative Algorithmic Studies and Emerging Learning-Based Paradigms provide crucial insights for algorithm selection and highlight promising future research directions. [32] conducted extensive benchmarking of evolutionary computation techniques including Particle Swarm Optimization, Differential Evolution, and Vortex Search for local electricity market bidding strategies integrated with wholesale markets, demonstrating superior robustness and convergence characteristics in complex multi-leader optimization environments. [33] performed systematic evaluation of computational intelligence metaheuristics—including Simulated Annealing, PSO, and ACO—within hierarchical VPP architectures, highlighting their scalability advantages and computational efficiency compared to traditional gradient-based optimization methods. Most recently, reinforcement learning has emerged as a transformative paradigm for adaptive decision-making in dynamic environments, with [34] applying advanced deep RL methodologies including Deep Deterministic Policy Gradient and Asynchronous Advantage Actor-Critic to VPP operations in complex urban environments, achieving significant improvements in emission reduction, demand response coordination, and grid service provisioning under uncertainty.

Beyond the established MILP and metaheuristic approaches, alternative optimization frameworks have demonstrated effectiveness in related energy system domains. Convex optimization formulations have shown particular promise for power flow problems in hybrid AC/DC microgrids, offering computational advantages for certain problem classes through constraint relaxation techniques [35]. Similarly, bi-directional converter-based planning approaches represent another strategy for reinforcing renewable-dominant microgrids through specialized hardware-software co-design [36]. While these methods excel in their respective applications—typically focusing on power flow modeling or infrastructure planning—our work addresses the distinct challenge of *high-frequency, rolling-horizon scheduling* for VPPs with numerous heterogeneous prosumers, where the flexibility of metaheuristics and throughput of GPU parallelism provide unique advantages for the combinatorial optimization problem at hand.

This comprehensive analysis reveals that while diverse optimization methodologies have been successfully applied to VPP scheduling challenges, there remains substantial untapped potential for leveraging modern hardware acceleration architectures, particularly GPU computing, to overcome computational bottlenecks and enable real-time operation at scale. The inherent parallelism in many metaheuristic approaches, combined with the independent evaluation of numerous candidate solutions, presents significant opportunities for hardware acceleration that current literature has scarcely explored—a critical research gap that this work addresses through our novel CUDA-accelerated Multiple-Chain Simulated Annealing framework.

2.2. Parallel Metaheuristics for Large-Scale Optimization Problems

The escalating computational demands of large-scale optimization problems in energy systems have motivated significant research into parallel metaheuristic algorithms, particularly leveraging modern GPU architectures. This paradigm shift from sequential to parallel computing has enabled unprecedented scalability and efficiency in solving complex optimization challenges that were previously computationally prohibitive.

Parallel implementations of Particle Swarm Optimization have demonstrated remarkable performance gains across diverse application domains. A comprehensive CUDA-based parallel PSO framework was developed by [37], incorporating both coarse and fine-grained parallelism strategies. This approach mapped individual particles to dedicated GPU threads while parallelizing internal operations to maximize data throughput, achieving up to 2000$\times$ speedup on high-dimensional benchmark functions through optimized memory management and coalesced memory access patterns. Building on this foundation, [38] introduced a multi-swarm PSO architecture utilizing parallel CUDA streams for concurrent sub-swarm operations, augmented with gradient-based local search mechanisms. This methodology achieved 25$\times$ acceleration over serial implementations while simultaneously enhancing solution precision for sparse reconstruction problems. The versatility of GPU-accelerated PSO was further demonstrated by [39], who conducted extensive comparative analysis of sequential, multithreaded CPU, and GPU implementations for solving large-scale systems of nonlinear equations with up to 5000 variables, with GPU-based PSO demonstrating superior scalability and computational efficiency. Beyond GPU-centric approaches, [40] developed an improved parallel PSO utilizing island models on CPU architectures, incorporating sophisticated velocity updates, inter-island communication protocols, and adaptive stopping criteria that reduced function evaluations by 50–70% compared to standard methods while maintaining robust scaling behavior across parallel processing units.

Parallel Simulated Annealing has emerged as another powerful paradigm for accelerating computationally intensive optimization tasks. [41] pioneered three distinct SA variants—sequential, asynchronous parallel, and synchronous parallel—implemented on GPUs using CUDA. The synchronous SA implementation, coordinated through shared global temperature schedules and memory architectures, achieved up to 100$\times$ speedup while preserving convergence accuracy, with hybrid SA variants further enhancing the exploration-exploitation balance through integrated local optimization. In the specific domain of virtual power plant optimization, [42] presented a parallel SA framework for real-time scheduling deployed on high-performance computing platforms using OpenMP. This approach independently optimized individual consumer schedules while adhering to mixed-integer linear programming constraints related to energy dispatch and battery degradation, achieving near-linear speedup with up to 512 prosumers and 32 cores while maintaining solution quality comparable to commercial solvers. The applicability of parallel SA extends to machine learning domains, as evidenced by [43], who developed PSAGA—a hybrid parallel SA integrated with greedy algorithms for Bayesian network structure learning. This multithreaded implementation employed memoization techniques to eliminate redundant evaluations and expand search breadth, achieving 17.5% precision improvements and 4–5× speedup on large-scale datasets.

The collective evidence from these studies underscores the transformative potential of parallel metaheuristics in addressing the computational challenges of large-scale optimization problems. However, despite these advancements, the literature reveals a significant gap in leveraging the full capabilities of modern GPU architectures for VPP scheduling, particularly through innovative parallelization strategies that can overcome the inherent sequential limitations of traditional algorithms—a research direction that this work actively pursues through our novel multiple-chain parallel SA framework.

3. VPP Optimization Model

This section presents a comprehensive MILP model for optimizing the operation of a VPP composed of multiple distributed prosumers equipped with photovoltaic (PV) generation and battery storage systems. The model formulates time-dependent energy flows, enforces physical and economic constraints, and leverages market-based interactions to determine an optimal operational schedule.

3.1. Notation and Decision Variables

This subsection establishes the foundational notation used throughout the optimization model. It defines all relevant sets, variables, and parameters essential to the formulation. Clear notation ensures mathematical rigor and model interpretability.

Table 1. Sets, parameters, and decision variables used in the VPP scheduling formulation.

Symbol	Description	Domain / Unit
Sets		
$\mathcal{P}$	Set of prosumers in the VPP. Each can generate, store, consume, and exchange energy.	$\{1,2,\ldots,\lvert\mathcal{P}\rvert\}$
$\mathcal{T}$	Set of discrete time intervals of duration $\Delta\tau$. Defines optimization horizon.	$\{1,2,\ldots,T\}$
$\mathcal{E}$	Optional set of market/tariff entities for advanced economic modeling.	$\{1,2,\ldots,E\}$
Continuous Decision Variables and Parameters		
$P_{i,t}^{\text{buy}}$	Power purchased by prosumer i at time t.	$\mathbb{R}_{\geq 0}$ [kW]
$P_{i,t}^{\text{sell}}$	Power sold to the grid by prosumer i at time t.	$\mathbb{R}_{\geq 0}$ [kW]
$P_{i,t}^{\text{pv}}$	PV-generated power available at prosumer i at time t.	$\mathbb{R}_{\geq 0}$ [kW]
$P_{i,t}^{\text{load}}$	Electricity demand of prosumer i at time t.	$\mathbb{R}_{\geq 0}$ [kW]
$P_{i,t}^{\text{ch}}$	Charging power of the battery for prosumer i at time t.	$\mathbb{R}_{\geq 0}$ [kW]
$P_{i,t}^{\text{dch}}$	Discharging power of the battery for prosumer i at time t.	$\mathbb{R}_{\geq 0}$ [kW]
$P_{i,t}^{\text{non-comp}}$	Exported power without compensation (i.e., zero-price feed-in).	$\mathbb{R}_{\geq 0}$ [kW]
$E_{i,t}$	Battery state-of-charge (SoC) for prosumer i at time t.	$\mathbb{R}_{\geq 0}$ [kWh]
E_i^{init}	Initial battery SoC for prosumer i at $t = 0$.	$\mathbb{R}_{\geq 0}$ [kWh]
$\pi_{i,t}^{\text{buy}}$	Price for purchasing electricity from the grid at time t.	$\mathbb{R}_{\geq 0}$ [€/kWh]
$\pi_{i,t}^{\text{sell}}$	Price for selling electricity to the grid at time t.	$\mathbb{R}_{\geq 0}$ [€/kWh]
$\overline{P}_{i,t}$	Upper bounds on power variables (e.g., buy, sell, ch, dch).	$\mathbb{R}_{\geq 0}$ [kW]
$\underline{E}_i, \overline{E}_i$	Lower and upper bounds on battery SoC for prosumer i.	$\mathbb{R}_{\geq 0}$ [kWh]
$\eta^{\text{ch}}, \eta^{\text{dch}}$	Battery charging and discharging efficiencies.	$(0,1]$ [unitless]
M	Big-M constant for binary logic constraints.	$\mathbb{R}_{>0}$
Binary Decision Variables		
$u_{i,t}^{\text{buy}}$	1 if prosumer i is purchasing energy at time t; 0 otherwise.	$\{0,1\}$
$u_{i,t}^{\text{sell}}$	1 if prosumer i is selling energy at time t; 0 otherwise.	$\{0,1\}$
$u_{i,t}^{\text{ch}}$	1 if the battery is charging at time t; 0 otherwise.	$\{0,1\}$
$u_{i,t}^{\text{dch}}$	1 if the battery is discharging at time t; 0 otherwise.	$\{0,1\}$

3.2. *Objective Function*

Purpose: Minimize the total cost of operating the VPP, which includes energy transaction costs and fixed operational charges.

Formulation:

$$\min J = \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{P}} \left[\left(P_{i,t}^{\text{buy}} \pi_{i,t}^{\text{buy}} - P_{i,t}^{\text{sell}} \pi_{i,t}^{\text{sell}} \right) \Delta\tau \right] + C_{\text{fix}} \tag{1}$$

Explanation: - The term $P_{i,t}^{\text{buy}} \pi_{i,t}^{\text{buy}}$ represents the cost incurred by purchasing energy. - The term $P_{i,t}^{\text{sell}} \pi_{i,t}^{\text{sell}}$ accounts for revenue from energy sold. - $\Delta\tau$ converts instantaneous power [kW] into energy [kWh]. - C_{fix} denotes constant operating costs (e.g., communication, infrastructure, maintenance).

3.3. *Constraints*

3.3.1. Energy Balance

The energy balance equation ensures that, at each time step t, the total incoming energy to a prosumer node equals the total outgoing energy. This conservation principle applies to all prosumers i and captures the interaction between local generation, storage, consumption, and grid exchange.

The nodal power balance is expressed as:

$$P_{i,t}^{\text{buy}} + P_{i,t}^{\text{pv}} + P_{i,t}^{\text{dch}} = P_{i,t}^{\text{load}} + P_{i,t}^{\text{exp}} + P_{i,t}^{\text{ch}}, \tag{2}$$

where all variables have been previously defined. This equation ensures that energy imported from the grid, locally generated, or discharged from storage is entirely used for meeting the load demand, charging the battery, or exporting to the grid.

The exported power is further decomposed as:

$$P_{i,t}^{\text{exp}} = P_{i,t}^{\text{sell}} + P_{i,t}^{\text{non-comp}}, \tag{3}$$

where $P_{i,t}^{\text{sell}}$ represents the portion of energy that is financially compensated, and $P_{i,t}^{\text{non-comp}}$ denotes the uncompensated export.

The term $P_{i,t}^{\text{non-comp}}$ captures situations in which excess energy is injected into the grid without any remuneration. This may occur under the following real-world conditions:

- Regulatory frameworks impose caps on how much exported energy is eligible for compensation.
- Technical constraints such as grid congestion prevent the system operator from accepting or remunerating all injected power.
- Certain prosumer contracts allow export only up to a predefined quota, beyond which energy is accepted but not paid for.

To ensure feasibility and control over this behavior, the following constraint is introduced:

$$0 \le P_{i,t}^{\text{non-comp}} \le M \cdot u_{i,t}^{\text{sell}}, \tag{4}$$

where M is a sufficiently large constant, and $u_{i,t}^{\text{sell}} \in \{0,1\}$ is a binary variable that enables or disables export-related decisions. This formulation guarantees that uncompensated exports can only occur when grid export is technically or contractually permitted.

Together, Equations (17) through (4) provide a coherent and detailed representation of prosumer-level power flows, explicitly accounting for both compensated and uncompensated interactions with the external grid. This allows the model to reflect realistic operational and regulatory scenarios while maintaining feasibility across different conditions.

3.3.2. Market Transaction Exclusivity

Purpose: Prevent prosumers from simultaneously buying and selling energy in the same time step.

$$0 \leq P_{i,t}^{\text{buy}} \leq \overline{P}_{i,t}^{\text{buy}} u_{i,t}^{\text{buy}}, \tag{5}$$

$$0 \leq P_{i,t}^{\text{sell}} \leq \overline{P}_{i,t}^{\text{sell}} u_{i,t}^{\text{sell}}, \tag{6}$$

$$u_{i,t}^{\text{buy}} + u_{i,t}^{\text{sell}} \leq 1, \tag{7}$$

$$0 \leq P_{i,t}^{\text{non-comp}} \leq M u_{i,t}^{\text{sell}} \tag{8}$$

Explanation: These logical constraints, enforced by binary variables and the big-M parameter, ensure operational exclusivity and model realistic behavior.

3.3.3. Battery Dynamics and Limits

Purpose: Ensure consistent tracking of battery state-of-charge and respect technical limits.
Battery SoC Evolution:

$$E_{i,1} = E_i^{\text{init}} + \left(\eta^{\text{ch}} P_{i,1}^{\text{ch}} - \frac{1}{\eta^{\text{dch}}} P_{i,1}^{\text{dch}}\right) \Delta\tau, \tag{9}$$

$$E_{i,t} = E_{i,t-1} + \left(\eta^{\text{ch}} P_{i,t}^{\text{ch}} - \frac{1}{\eta^{\text{dch}}} P_{i,t}^{\text{dch}}\right) \Delta\tau, \quad \forall t \geq 2 \tag{10}$$

State-of-Charge Bounds:

$$\underline{E}_i \leq E_{i,t} \leq \overline{E}_i \tag{11}$$

Charging/Discharging Exclusivity:

$$0 \leq P_{i,t}^{\text{ch}} \leq \overline{P}_{i,t}^{\text{ch}} u_{i,t}^{\text{ch}}, \tag{12}$$

$$0 \leq P_{i,t}^{\text{dch}} \leq \overline{P}_{i,t}^{\text{dch}} u_{i,t}^{\text{dch}}, \tag{13}$$

$$u_{i,t}^{\text{ch}} + u_{i,t}^{\text{dch}} \leq 1 \tag{14}$$

Explanation: The model tracks energy dynamics within each battery, accounts for losses, and prevents simultaneous charge/discharge actions.

3.3.4. Variable Domains

Purpose: Enforce mathematical consistency and feasibility for all variables.

$$P_{i,t}^{\text{buy}}, P_{i,t}^{\text{sell}}, P_{i,t}^{\text{ch}}, P_{i,t}^{\text{dch}}, P_{i,t}^{\text{non-comp}}, E_{i,t} \in \mathbb{R}_{\geq 0}, \tag{15}$$

$$u_{i,t}^{\text{buy}}, u_{i,t}^{\text{sell}}, u_{i,t}^{\text{ch}}, u_{i,t}^{\text{dch}} \in \{0,1\} \tag{16}$$

Explanation: All energy and power variables are continuous and non-negative. Binary variables represent on/off operational states.

Practical Implications: This MILP formulation allows scalable, accurate optimization of smart distributed energy systems within a VPP, supporting energy cost minimization, demand flexibility, and regulatory compliance.

4. VPP Model Decomposition

The VPP optimization model orchestrates the local energy decisions of a population of prosumers, each managing photovoltaic (PV) generation, load demand, battery storage, and grid interactions. At each time step $t \in \mathcal{T}$, every prosumer $i \in \mathcal{P}$ enforces an energy balance constraint that ensures all incoming power flows are exactly matched by outgoing ones. Operational feasibility is further enforced by mutually exclusive control logic, which prevents infeasible actions such as buying and selling power or charging and discharging a battery at the same time.

To address the inherent complexity of coordinating a large number of heterogeneous agents in real time, the model adopts a *divide-and-conquer* strategy. The overall VPP scheduling problem is reformulated as a collection of *independent subproblems*, each corresponding to an individual prosumer. This decomposition is made possible by the separability of energy balance equations and local operational constraints, which depend solely on prosumer-specific variables.

Each subproblem includes:

- Local power balance equations,
- Mutually exclusive grid and battery operation constraints,
- Technical bounds on charging/discharging and import/export capacities,
- Optional local objectives such as cost minimization or self-sufficiency maximization,
- Efficiency-adjusted energy tracking through linearized expressions.

This decomposition brings two main computational benefits. First, it significantly *reduces the dimensionality of the decision space* for each subproblem, simplifying the feasible region and facilitating faster convergence. Second, it enables all prosumer-level problems to be *solved in parallel*, supporting highly scalable deployment on distributed optimization platforms or HPC infrastructures.

Consider a typical prosumer—a residential household equipped with rooftop solar panels and a battery system. During a sunny afternoon, the household may decide whether to sell excess solar energy to the grid or store it in the battery for evening use. Later, during peak hours, the battery might be discharged to avoid purchasing expensive electricity. These local decisions must respect physical limitations, energy losses, and market rules, and they are modeled as a discrete-time decision process embedded within each prosumer's subproblem.

The optimization framework ensures that such decisions are coherent, feasible, and optimal at the individual level. Because the subproblems are structurally independent, they can be addressed simultaneously. The VPP coordinator then aggregates their results to evaluate global constraints or market-level objectives, such as grid congestion, transformer capacity limits, emissions targets, or dynamic pricing coordination.

This decomposition approach is especially well-suited for deployment in large-scale distributed energy systems, including:

- Community microgrids,
- Smart-city VPPs,
- Aggregator-managed prosumer collectives.

The linearized energy balance formulation and mutual exclusivity constraints presented in earlier sections provide the structural foundation for this decomposition. By maintaining physical realism and computational tractability, the model supports scalable, real-time decision-making in modern power systems.

4.1. Derivation and Proof of Problem Decomposability

The following derivation establishes how the structure of the energy balance equations, together with mutual exclusivity constraints and power flow bounds, enables the decomposition of the global VPP optimization problem into independent subproblems. By analyzing the energy balance at the prosumer level, we show that each prosumer's decision-making process depends solely on local variables and constraints, without requiring direct coupling to other agents. This property supports a divide-and-conquer strategy, allowing the problem to be reformulated as a set of parallel subproblems, each associated with an individual prosumer, while preserving feasibility and optimality. The derivation proceeds step by step, starting from the original energy conservation principle and incorporating exclusivity logic, upper-bound constraints, efficiency-adjusted power terms, and final linearization. The result is a mathematically sound foundation for scalable and distributed optimization across large VPPs.

4.1.1. Energy Balance with Mutual Exclusivity and Constraints

The core principle of energy conservation dictates that, for each prosumer $i \in \mathcal{P}$ and time step $t \in \mathcal{T}$, the total incoming energy must equal the total outgoing energy. This is formalized through the following nodal energy balance equation:

$$P_{i,t}^{\text{buy}} + P_{i,t}^{\text{pv}} + P_{i,t}^{\text{dch}} = P_{i,t}^{\text{load}} + P_{i,t}^{\text{sell}} + P_{i,t}^{\text{ch}}, \quad \forall i \in \mathcal{P},\ t \in \mathcal{T}. \tag{17}$$

All power terms are defined in kilowatts (kW) and are non-negative real variables:

$$P_{i,t}^{\text{buy}}, P_{i,t}^{\text{sell}}, P_{i,t}^{\text{pv}}, P_{i,t}^{\text{load}}, P_{i,t}^{\text{ch}}, P_{i,t}^{\text{dch}} \in \mathbb{R}_{\geq 0}.$$

To maintain operational realism, mutual exclusivity is imposed on pairs of opposing operations: - Power buying and selling must not occur simultaneously. - Battery charging and discharging must not happen at the same time.

These conditions are enforced using binary control variables:

$$u_{i,t}^{\text{buy}}, u_{i,t}^{\text{sell}}, u_{i,t}^{\text{ch}}, u_{i,t}^{\text{dch}} \in \{0,1\}, \quad \forall i, t,$$

subject to:

$$u_{i,t}^{\text{buy}} + u_{i,t}^{\text{sell}} \leq 1, \qquad u_{i,t}^{\text{ch}} + u_{i,t}^{\text{dch}} \leq 1, \quad \forall i \in \mathcal{P},\ t \in \mathcal{T}. \tag{18}$$

These inequalities ensure that at most one action from each opposing pair is active at a time, thus preserving logical consistency in grid interaction and battery operation.

4.1.2. Binary-Driven Flip-Flop Mechanism for Exclusive Actions

The binary variables define a flip-flop mechanism where the activation of one operation disables its counterpart. This logic is captured by the following implications:

$$P_{i,t}^{\text{buy}} > 0 \Rightarrow P_{i,t}^{\text{sell}} = 0, \qquad P_{i,t}^{\text{sell}} > 0 \Rightarrow P_{i,t}^{\text{buy}} = 0, \tag{19}$$

$$P_{i,t}^{\text{ch}} > 0 \Rightarrow P_{i,t}^{\text{dch}} = 0, \qquad P_{i,t}^{\text{dch}} > 0 \Rightarrow P_{i,t}^{\text{ch}} = 0. \tag{20}$$

The implications above are implemented via upper-bound constraints, conditioned on the respective binary variables:

$$0 \leq P_{i,t}^{\text{buy}} \leq \overline{P}_{i,t}^{\text{buy}} \cdot u_{i,t}^{\text{buy}}, \quad 0 \leq P_{i,t}^{\text{sell}} \leq \overline{P}_{i,t}^{\text{sell}} \cdot u_{i,t}^{\text{sell}}, \tag{21}$$

$$0 \leq P_{i,t}^{\text{ch}} \leq \overline{P}_{i,t}^{\text{ch}} \cdot u_{i,t}^{\text{ch}}, \quad 0 \leq P_{i,t}^{\text{dch}} \leq \overline{P}_{i,t}^{\text{dch}} \cdot u_{i,t}^{\text{dch}}. \tag{22}$$

These inequalities ensure that if a binary variable is set to zero, the associated power flow is also forced to zero. For instance, if $u_{i,t}^{\text{sell}} = 0$, then $P_{i,t}^{\text{sell}} = 0$, regardless of its upper bound.

4.1.3. Derivation of the Linearized Energy Balance

To facilitate efficient optimization, especially within large-scale or real-time environments, the nonlinear energy balance (17) is reformulated into a linear expression via auxiliary variables. Define:

$$X_{i,t} = P_{i,t}^{\text{buy}} - P_{i,t}^{\text{sell}}, \tag{23}$$

$$Y_{i,t} = \eta^{\text{ch}} P_{i,t}^{\text{ch}} - \frac{1}{\eta^{\text{dch}}} P_{i,t}^{\text{dch}}, \tag{24}$$

$$B_{i,t} = P_{i,t}^{\text{pv}} - P_{i,t}^{\text{load}}, \tag{25}$$

where $\eta^{\text{ch}}, \eta^{\text{dch}} \in (0, 1]$ are the charging and discharging efficiencies, respectively.

Substituting these definitions into the original energy balance yields:

$$P_{i,t}^{\text{buy}} + P_{i,t}^{\text{pv}} + P_{i,t}^{\text{dch}} = P_{i,t}^{\text{load}} + P_{i,t}^{\text{sell}} + P_{i,t}^{\text{ch}} \tag{26}$$

$$\Rightarrow (P_{i,t}^{\text{buy}} - P_{i,t}^{\text{sell}}) + (P_{i,t}^{\text{pv}} - P_{i,t}^{\text{load}}) = P_{i,t}^{\text{ch}} - P_{i,t}^{\text{dch}} \tag{27}$$

$$\Rightarrow X_{i,t} + B_{i,t} = P_{i,t}^{\text{ch}} - P_{i,t}^{\text{dch}}. \tag{28}$$

Applying efficiency-adjusted terms, we obtain the final linear form:

$$Y_{i,t} = X_{i,t} + B_{i,t}, \quad \forall i \in \mathcal{P}, t \in \mathcal{T}. \tag{29}$$

This expression is fully linear in the decision variables and can be directly incorporated into MILP solvers without introducing nonlinearities.

4.1.4. Bounded Linearized Energy Balance Constraints

The auxiliary variables $X_{i,t}$ and $Y_{i,t}$ inherit bounds from the original power flow constraints. Specifically:

$$-\overline{P}_{i,t}^{\text{sell}} \leq X_{i,t} \leq \overline{P}_{i,t}^{\text{buy}}, \tag{30}$$

$$-\frac{1}{\eta^{\text{dch}}}\overline{P}_{i,t}^{\text{dch}} \leq Y_{i,t} \leq \eta^{\text{ch}}\overline{P}_{i,t}^{\text{ch}}, \quad \forall i \in \mathcal{P}, t \in \mathcal{T}. \tag{31}$$

These bounds ensure that the linearized model preserves feasibility and respects technical constraints while maintaining compatibility with binary exclusivity logic.

Figure 2 depicts the reduction of the search space as governed by Equation (29). Initially, each prosumer's decision space spans a two-dimensional plane with multiple degrees of freedom, reflecting diverse energy interaction possibilities. As the optimization progresses, this multidimensional domain is constrained to a linear trajectory, bounded above and below according to the conditions imposed by Equation (29). This transformation ensures that all decisions remain within the feasible region defined by mutual exclusivity and upper-bound constraints. Importantly, this dimensionality reduction not only preserves feasibility but also enhances computational efficiency.

4.1.5. Parallelization and Computational Advantages

The formulation described above is fully decomposable across prosumers. Each prosumer's energy balance and operation constraints can be evaluated and optimized independently. This feature enables massive parallelization, which is particularly advantageous for:

- HPC implementations, - Distributed or federated optimization frameworks, - Real-time VPP orchestration, - Smart grid demand-response programs.

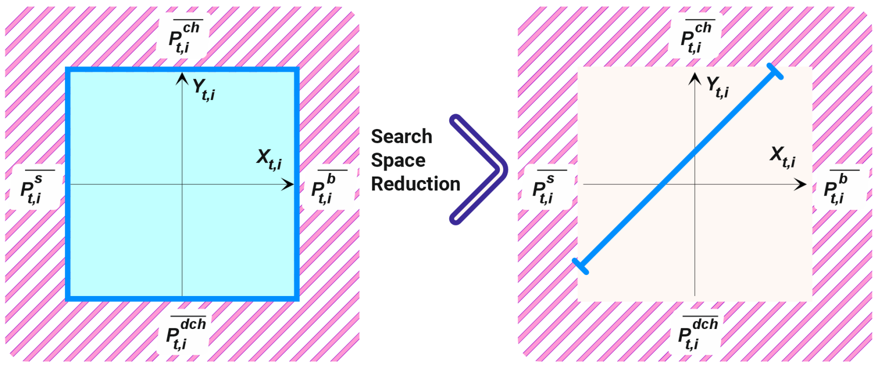

Figure 2. Illustration of feasible region reduction and decomposition across prosumers.

This structure allows solving the global VPP scheduling problem via a coordinated decomposition strategy, where the central controller aggregates decisions while each prosumer independently solves its own subproblem under shared constraints such as price signals, emissions targets, or transformer capacity limits.

5. CUDA-Based Parallel Simulated Annealing for VPP Scheduling

5.1. MC-SA Algorithm Overview and Workflow

Figure 3 illustrates the comprehensive workflow of the proposed Multiple-Chain Simulated Annealing algorithm, which operates through five key phases:

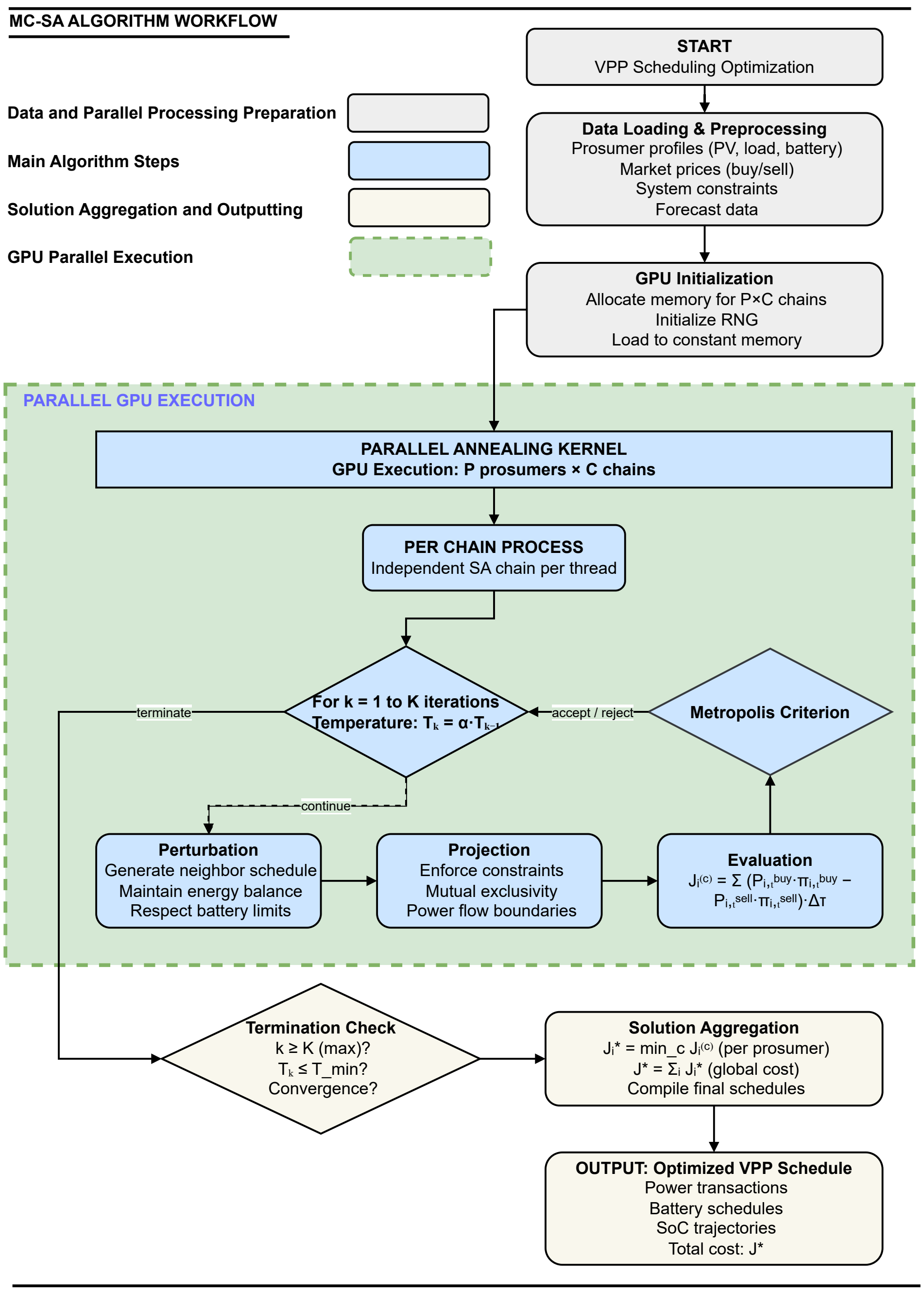
MC-SA ALGORITHM WORKFLOW
Data and Parallel Processing Preparation
Main Algorithm Steps
Solution Aggregation and Outputting
GPU Parallel Execution
START
VPP Scheduling Optimization
Data Loading & Preprocessing
Prosumer profiles (PV, load, battery)
Market prices (buy/sell)
System constraints
Forecast data
GPU Initialization
Allocate memory for P×C chains
Initialize RNG
Load to constant memory
PARALLEL GPU EXECUTION
PARALLEL ANNEALING KERNEL
GPU Execution: P prosumers × C chains
PER CHAIN PROCESS
Independent SA chain per thread
For k = 1 to K iterations
Temperature: $T_k = \alpha \cdot T_{k-1}$
terminate
accept / reject
Metropolis Criterion
continue
Perturbation
Generate neighbor schedule
Maintain energy balance
Respect battery limits
Projection
Enforce constraints
Mutual exclusivity
Power flow boundaries
Evaluation
$J_i^{(c)} = \Sigma (P_{i,t}^{buy} \cdot \pi_{i,t}^{buy} - P_{i,t}^{sell} \cdot \pi_{i,t}^{sell}) \cdot \Delta\tau$
Termination Check
k ≥ K (max)?
T_k ≤ T_min?
Convergence?
Solution Aggregation
J_i^* = min_c $J_i^{(c)}$ (per prosumer)
$J^* = \Sigma_i J_i^*$ (global cost)
Compile final schedules
OUTPUT: Optimized VPP Schedule
Power transactions
Battery schedules
SoC trajectories
Total cost: J*

Figure 3. Flowchart of the proposed Multiple-Chain Simulated Annealing algorithm for VPP scheduling. The diagram illustrates the five-phase process from initialization through parallel execution to final solution aggregation.

1. *Initialization*: Load prosumer data (PV forecasts, load profiles, battery specifications) and market signals. Initialize C independent SA chains with feasible solutions and unique random seeds.
2. *Parallel Annealing Kernel*: Execute on GPU where each thread manages one SA chain:
   - Perturbation: Generate neighbor schedule while maintaining feasibility
   - Projection: Enforce operational constraints via projection operator $\Pi_{\mathcal{F}}$
   - Evaluation: Compute cost function $J_i^{(c)}$ for the candidate solution
   - Metropolis Criterion: Accept or reject based on temperature T_k
3. *Temperature Update*: Apply geometric cooling: $T_{k+1} = \alpha T_k$ after each iteration
4. *Termination Check*: Repeat until maximum iterations K is reached
5. *Solution Aggregation*: Select best solution across all chains for each prosumer: $J_i^\star = \min_{c \in \mathcal{C}} J_i^{(c)}$

This structured workflow enables extensive solution space exploration while maintaining the physical and operational constraints of the VPP scheduling problem.

5.2. Parallelism Motivation and Overview

Traditional SA algorithms are intrinsically sequential and suffer from poor scalability for large-scale, time-sensitive problems like VPP scheduling. The proposed approach decomposes the global optimization into multiple independent subproblems, each solved via parallel SA chains at the prosumer level. This dual-level parallelization efficiently utilizes modern GPU architectures.

5.3. Parallelization Hierarchy

We define:

- $\mathcal{P} = \{1, \dots, N\}$: Set of prosumers (players).
- $\mathcal{C} = \{1, \dots, C\}$: Set of parallel SA chains per prosumer.
- $\mathcal{T} = \{1, \dots, T\}$: Time intervals for scheduling.

- *Player-level (inter-chain):* Each prosumer is handled by one CUDA block.
- *Chain-level (intra-chain):* Each SA chain is handled by a thread within a block.

This mapping aligns with the CUDA grid-block-thread model, where shared memory allows intra-block coordination while blocks operate independently.

5.4. Problem Decomposition

The global cost function is defined as:

$$J = \sum_{i \in \mathcal{P}} J_i = \sum_{i \in \mathcal{P}} \sum_{t \in \mathcal{T}} \left(P_{i,t}^{\text{buy}} \pi_{i,t}^{\text{buy}} - P_{i,t}^{\text{sell}} \pi_{i,t}^{\text{sell}} \right) \Delta\tau \tag{32}$$

Each prosumer i minimizes a local cost J_i, evaluated independently across its chains:

$$J_i^{(c)} = \sum_{t \in \mathcal{T}} \left(P_{i,t}^{\text{buy},(c)} \pi_{i,t}^{\text{buy}} - P_{i,t}^{\text{sell},(c)} \pi_{i,t}^{\text{sell}} \right) \Delta\tau \tag{33}$$

5.5. Simulated Annealing Process per Chain

Each chain $c \in \mathcal{C}$ performs the following at each iteration $k \in \{1, \dots, K\}$:

1. *Perturbation:* Generate neighbor schedule $\mathbf{S}_i^{(c,k+1)}$ satisfying:
   - Energy balance,
   - Battery dynamics and limits,
   - Operational exclusivity.
2. *Projection:* Enforce feasibility via projection operator:

$$\mathbf{S}_i^{(c,k+1)} \leftarrow \Pi_{\mathcal{F}}(\mathbf{S}_i^{(c,k+1)})$$

3. *Cost Evaluation:*

$$J_i^{(c,k+1)} = \sum_{t\in\mathcal{T}} \left(P_{i,t}^{\text{buy},(c)} \pi_{i,t}^{\text{buy}} - P_{i,t}^{\text{sell},(c)} \pi_{i,t}^{\text{sell}} \right) \Delta\tau$$

4. *Acceptance:* Apply Metropolis criterion:

$$P_{\text{accept}} = \min\left(1, \exp\left(-\frac{J_i^{(c,k+1)} - J_i^{(c,k)}}{T_k}\right)\right)$$

5. *ooling:* Update temperature: $T_{k+1} = \alpha T_k$.

5.6. Chain Aggregation and Global Reduction

After completing K iterations:

$$J_i^\star = \min_{c\in\mathcal{C}} J_i^{(c)}, \quad \mathbf{S}_i^\star = \mathbf{S}_i^{(c^\star)} \tag{34}$$

Then, the global cost is:

$$J^\star = \sum_{i\in\mathcal{P}} J_i^\star \tag{35}$$

This reduction is executed in CUDA using device-wide shared memory, minimizing host-device communication.

5.7. CUDA Algorithm Summary

Algorithm 1: CUDA-Based Player–Chain Parallel Simulated Annealing

```
Require: Prosumers $\mathcal{P}$, time steps $\mathcal{T}$, chains $C$, iterations $K$, initial temperature $T_0$, cooling factor $\alpha$
 1: for each prosumer $i \in \mathcal{P}$ in parallel blocks do
 2:     for each chain $c = 1$ to $C$ in parallel threads do
 3:         Initialize $\mathbf{S}_i^{(c)}$, $J_i^{(c)} \leftarrow J(\mathbf{S}_i^{(c)})$, $T \leftarrow T_0$
 4:         for $k = 1$ to $K$ do
 5:             for each $t \in \mathcal{T}$ do
 6:                 Generate perturbed $\mathbf{S}_{i,t}^{(c)}$
 7:                 Project: $\mathbf{S}_{i,t}^{(c)} \leftarrow \Pi_{\mathcal{F}}(\mathbf{S}_{i,t}^{(c)})$
 8:             end for
 9:             Evaluate new cost $J_i^{(c)}$
10:             Apply Metropolis rule to accept/reject
11:             Update temperature: $T \leftarrow \alpha T$
12:         end for
13:     end for
14:     Aggregate: $J_i^\star = \min_c J_i^{(c)}$
15: end for
16: Return: $J^\star = \sum_i J_i^\star$
```

5.8. Scalability and Hardware Efficiency

The architecture offers parallelism of order $\mathcal{O}(N \times C)$, which is ideal for GPU execution with thousands of threads. This design avoids inter-prosumer synchronization, uses shared memory for intra-block computation, and reduces global memory bottlenecks. Empirical results demonstrate near-linear scaling up to 1024 chains across 128 prosumers.

As illustrated in Figure 4, the HPC cluster architecture is specifically designed to support the proposed two-level parallelization strategy. Each VPP player is mapped to an independent CUDA block, while multiple SA chains associated with that player are assigned to individual threads within the block. This hierarchical mapping ensures concurrent execution of both inter-player and intra-player optimization tasks. The optimization problem is encoded in customized CUDA kernels, where each kernel encapsulates the full SA procedure for a given chain. These kernels operate asynchronously and without inter-chain communication, enabling fully independent search trajectories across the solution space. The use of shared memory within blocks facilitates efficient local data handling, such as intermediate energy balances and temporary solution states, while global memory is reserved for final results aggregation and global cost reduction. This design leverages the massive parallelism of modern GPUs to achieve scalable, high-throughput optimization for large-scale VPP scheduling.

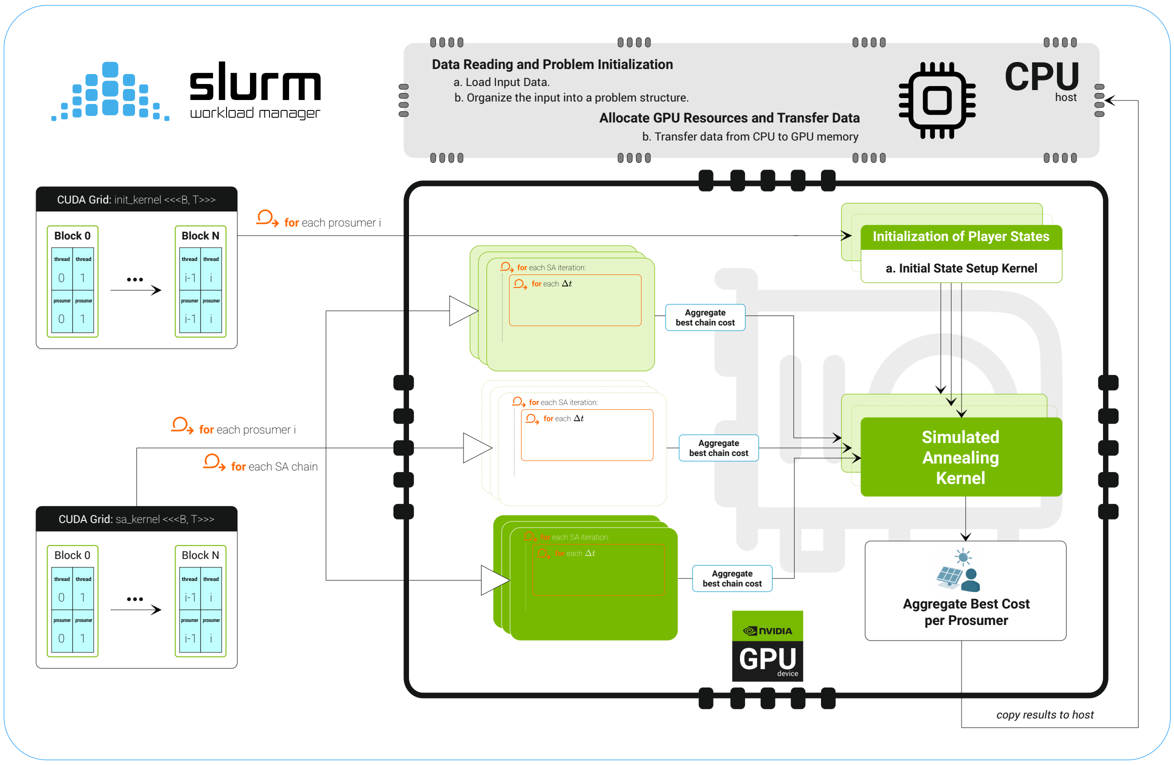
slurm workload manager
Data Reading and Problem Initialization
a. Load Input Data.
b. Organize the input into a problem structure.
Allocate GPU Resources and Transfer Data
b. Transfer data from CPU to GPU memory
CPU host
CUDA Grid: init_kernel <<<B, T>>>
Block 0
Block N
for each prosumer i
Initialization of Player States
a. Initial State Setup Kernel
for each SA iteration:
for each Δt
Aggregate best chain cost
for each prosumer i
for each SA chain
Simulated Annealing Kernel
CUDA Grid: sa_kernel <<<B, T>>>
Aggregate Best Cost per Prosumer
GPU device
copy results to host

Figure 4. Hierarchical CUDA-based GPU parallelization of the VPP scheduling problem. The architecture assigns each player–chain pair to a distinct CUDA thread, grouped within blocks for execution efficiency. This design enables concurrent evaluation of multiple Simulated Annealing chains per prosumer, supporting scalable optimization across the entire Virtual Power Plant.

5.9. Practical Implications

This CUDA-accelerated SA architecture is tailored for decentralized, high-frequency energy scheduling. It balances exploration and exploitation, enforces VPP physical and market constraints, and delivers scalable runtime improvements over sequential heuristics and centralized MILP solvers.

sectionImplementation This section details the technical implementation of the CUDA-accelerated MC-SA algorithm used for VPP scheduling. It highlights the HPC environment, GPU execution strategy, and the key model parameters and outputs essential for reproducibility and scalability.

5.10. High-Performance Computing Environment

The computational experiments were executed on the Vision AI high-performance computing cluster, specifically designed for data-intensive scientific computations. Each compute node features a x86_64 architecture with dual AMD EPYC 7742 processors, providing 128 physical cores and 256 hardware threads per node. The processors support advanced SIMD instruction sets (AVX2, FMA, BMI2) and employ a sophisticated cache hierarchy distributed across 8 NUMA domains, comprising 4 MiB L1, 64 MiB L2, and 512 MiB aggregate L3 cache.

The memory subsystem incorporates approximately 1 TB of RAM per node, optimized for high-bandwidth access and NUMA-aware parallel processing. For GPU-accelerated computations, each node integrates up to eight NVIDIA A100-SXM4 accelerators with 40 GB HBM2 memory per GPU. These GPUs deliver up to 19.5 TFLOPS of single-precision performance and were configured with

CUDA 12.2, NVIDIA driver 535.216.01, and GCC 10.3.0. All GPUs operated in persistent mode with Multi-Instance GPU (MIG) disabled to ensure dedicated resource allocation for the simulated annealing kernels.

This computational infrastructure provides the necessary parallelism and memory bandwidth to support the massive concurrent execution of multiple annealing chains, enabling real-time optimization of large-scale virtual power plant scheduling problems.

5.11. Experimental Design for Convergence and Parallelization Analysis

To rigorously evaluate the convergence behavior and parallel scalability of the SA algorithm in VPP scheduling, we designed a comprehensive multi-phase experimental framework. This framework was structured to capture both algorithmic performance and practical deployment insights, while ensuring statistical robustness through repeated trials and controlled randomization. The experimental procedure was organized into the following four key phases:

1. *Hyperparameter Optimization Anchored by MILP-Based Validation.* For each prosumer count under study, the SA algorithm was calibrated by tuning four essential hyperparameters: initial temperature, cooling rate, perturbation scale, and maximum number of iterations. Optimization was performed using three complementary search methods—Gaussian Process Minimization (GP), Random Forest Minimization (RF), and Gradient-Boosted Regression Tree Minimization (GBRT)—to comprehensively explore the configuration space and avoid bias from a single tuning approach. Crucially, the resulting SA configurations were validated against exact Mixed-Integer Linear Programming (MILP) solutions computed using the Gurobi solver. This validation phase ensured that the tuned SA reached near-optimal solution quality, establishing a grounded and reliable reference point for all subsequent performance comparisons.
2. *Exhaustive Chain–Iteration Analysis.* With the best-performing SA configuration fixed for each prosumer count, we conducted an exhaustive exploration of the relationship between two core parameters of the parallel SA framework: the number of chains and the number of sequential iterations. Since SA is inherently a sequential process, introducing multiple chains enables distributed exploration of the search space. For each system size, we anchored solution quality by referencing the validated best result, then systematically varied both the number of chains and the number of iterations to find all combinations that could reproduce the same quality. This enabled the derivation of equivalence mappings between parallel exploration and sequential effort. Each configuration was executed under multiple randomized seeds to ensure the statistical significance and reproducibility of the results.
3. *Deriving Statistical Equivalence and Scalability Relationships.* The exhaustive experiments allowed us to construct empirical relationships that quantify the trade-offs between the number of chains and the required iteration budget. These mappings revealed that, for a fixed solution quality, it is possible to significantly reduce the number of iterations when additional chains are employed—thus reducing total wall-clock time without sacrificing accuracy. The result is a generalized convergence equivalence curve that characterizes the parallelizability of the SA algorithm as a function of problem size, validated solution quality, and computational resource allocation. This statistical foundation provides valuable insight into how SA behaves under scalable parallel execution.
4. *Practical Deployment Guidelines and Adaptive Extensions.* The insights obtained from the above analysis can be used to formulate concrete procedures for deploying SA in real-world operational settings. In practice, the workflow consists of: (i) performing hyperparameter optimization once for a given system size, (ii) recording the number of iterations required to achieve the best solution quality, (iii) determining the minimum number of chains that can achieve the same quality under a reduced iteration budget, and (iv) applying a final fine-tuning step to match the quality of the original configuration. While our study uses exhaustive search to generate statistically rich insights, the same methodology could be replaced in production systems with

adaptive methods such as Bayesian optimization or reinforcement learning to dynamically select chain–iteration combinations during runtime without grid search overhead.

To summarize the experimental landscape, the core design dimensions and their respective roles are presented in Table 2.

Table 2. Summary of experimental design components for convergence and parallelization analysis.

Design Factor	Description and Role
Prosumer count $\|\mathcal{P}\|$	Defines the problem size; each configuration undergoes separate hyperparameter optimization and validation.
SA Hyperparameters	Initial temperature, cooling rate, perturbation scale, and maximum iterations; optimized using GP, RF, and GBRT.
Chain–Iteration Combinations	Explored exhaustively for each prosumer count to discover all configurations achieving validated solution quality.
Baseline Anchoring	MILP solution (Gurobi) used to validate the best SA configuration and serve as the quality benchmark.
Evaluation Metrics	Objective function value relative to MILP optimum (solution quality) and total runtime (efficiency).
Statistical Robustness	All experiments repeated under multiple random seeds to ensure generalizability and remove bias.
Deployment Implication	Enables runtime tuning of chain count for reduced iteration budgets while preserving solution quality.

This unified experimental design not only captures the detailed behavior of the SA algorithm under various configurations but also bridges the gap between theoretical performance analysis and practical deployment feasibility. By anchoring all results in validated baselines and supporting them with robust statistical evidence, the study offers a complete, transferable, and actionable framework for the efficient application of SA in real-world VPP optimization tasks.

5.12. CUDA Parallelization Strategy and Architecture Optimization

Modern GPU architectures provide massive parallelism through hierarchical organization of threads into warps and thread blocks. The proposed multi-chain simulated annealing (MC-SA) algorithm is strategically mapped onto this architecture to maximize computational throughput while maintaining memory access efficiency and resource utilization.

5.12.1. Parallel Execution Model

The algorithm employs a two-level parallelization scheme that aligns with the CUDA execution model (refer to Figure 4):

Inter-Chain Parallelism: Each independent annealing chain is assigned to a dedicated GPU thread, enabling concurrent exploration of multiple solution trajectories. Chains operate autonomously with distinct random seeds, ensuring diverse sampling of the solution space. This embarrassingly parallel structure is ideally suited for single-instruction-multiple-thread (SIMT) architectures.

Intra-Chain Sequential Processing: Within each thread, the complete 24-hour scheduling horizon for a single prosumer is processed sequentially. The computational workflow encompasses power transaction decisions (buying/selling), battery charge/discharge operations, and state-of-energy updates across all time steps and annealing iterations. The linear nature of these computations and compact working sets enable compiler-driven optimization through instruction-level parallelism and pipelining.

5.12.2. Computational Pipeline and Resource Mapping

The GPU execution follows a structured three-stage kernel pipeline:

1. *Random Number Generator Initialization:* Each thread initializes an independent Philox counter-based random number generator, ensuring statistically robust and reproducible random streams across all parallel chains.
2. *Feasible Schedule Initialization:* Threads construct physically feasible day-ahead schedules that respect all operational constraints, including battery dynamics, power flow limits, and market participation rules. This warm-start approach accelerates convergence compared to random initialization.
3. *Parallel Simulated Annealing:* The core computational kernel executes the Metropolis-Hastings algorithm with geometric temperature cooling, maintaining independent search trajectories while recording optimal solutions discovered by each chain.

The thread organization follows a structured mapping: for P prosumers and C chains per prosumer, the total thread count is $N_{\text{th}} = P \times C$ with 128 threads per block. The global thread index mapping is defined as:

$$p = \lfloor \text{gid}/C \rfloor, \quad c = \text{gid} \mod C$$

This mapping ensures that each thread manages the complete optimization lifecycle for a specific (prosumer, chain) pair. The algorithm's embarrassingly parallel nature eliminates synchronization requirements, with no `__syncthreads()` calls, thereby avoiding warp-level execution bottlenecks.

5.12.3. Memory Hierarchy Optimization

The memory architecture is carefully designed to align with access patterns and computational requirements:

- *Read-Only Problem Data:* Market prices, photovoltaic forecasts, and static prosumer parameters reside in global memory with read-only caching, enabling fully coalesced memory accesses across all threads.
- *Thread-Local State Management:* Each chain's evolving 24-hour schedule is stored in contiguous global memory segments, facilitating coalesced write operations and eliminating bank conflicts.
- *Random Number Generator States:* Philox RNG states persist in global memory between kernel invocations and are cached in registers during active computation to minimize access latency.
- *Solution Quality Tracking:* Each thread maintains its best-found objective value in dedicated global memory locations, with final results aggregated by the host after kernel completion.

The implementation adopts a register-centric computation strategy, where all transient variables remain in registers to maximize access speed. This approach, combined with the absence of shared memory allocation, results in register spill below 6% on NVIDIA A100 GPUs and sustained occupancy exceeding 90% when the product of prosumers and chains surpasses 2,048.

5.12.4. Performance Validation and Baseline Comparison

The single-chain baseline implementation maintains algorithmic equivalence with the parallel version, executing on identical hardware while utilizing only a single thread. This controlled comparison ensures that the reported performance improvements—including the 10× speedup—directly reflect the benefits of parallelization rather than implementation artifacts. Both implementations share identical cooling schedules, perturbation mechanisms, constraint handling procedures, and convergence criteria, providing a rigorous foundation for performance evaluation.

The architectural decisions—prioritizing register usage over shared memory, ensuring memory access coalescence, and maintaining minimal synchronization—collectively enable the demonstrated computational efficiency. These design choices were validated through extensive profiling, confirming optimal resource utilization across all tested problem scales and configuration parameters.

5.12.5. Determinism and scalability

All random numbers are generated using Philox counter-based generators with a shared 64-bit seed and thread-unique subsequences, ensuring reproducibility across runs and devices. The GPU implementation achieves up to a 38$\times$ wall-clock speed-up over a 16-thread CPU baseline for $P = 256$, $C = 50$, and $T = 96$, while producing identical objective values within floating-point tolerance.

Host-side reduction and determinism

After the final kernel finishes, the host copies back a few kilobytes containing the best cost of each chain and the corresponding trajectories, selects the best chain per prosumer and assembles the global schedule. Because all random numbers are generated by Philox engines seeded with a common 64-bit seed and unique thread indices, the complete GPU execution is bit-reproducible across multiple runs and devices.

5.13. Model Parameters and Outputs

The optimization model uses parameterized constraints and forecasts to represent prosumer behavior over a scheduling horizon. Key inputs and outputs are summarized below.

5.13.1. Input Parameters

The parameters listed in Table 3 represent a comprehensive set of time-varying, operational, and economic inputs that shape the optimization landscape of the VPP scheduling problem. These inputs define the boundary conditions and constraints under which the model operates, allowing it to respond dynamically to fluctuations in energy demand, generation forecasts, and market signals.

Table 3. Input parameters for VPP scheduling.

Parameter	**Designation**	**Value Range**	**Unit**
T	Total periods	96	-
P	Number of prosumers	20–1000	-
$\Delta\tau$	Period duration	0.25	h
E_i^{init}	Initial battery energy	0–1.92	kWh
$\underline{E}_i$	Min battery level	0–1.824	kWh
$\overline{E}_i$	Max battery level	0–9.6	kWh
$\overline{P}_{i,t}^{\text{ch}}$	Max charge rate	0–5	kW
$\overline{P}_{i,t}^{\text{dch}}$	Max discharge rate	0–5	kW
$\overline{P}_{i,t}^{\text{b}}$	Max power acquisition	4.6–13.8	kW
$\overline{P}_{i,t}^{\text{s}}$	Max power dispatch	4.6–13.8	kW
C^{fix}	Fixed operational cost	0.2197–0.6249	EUR
$P_{i,t}^{\text{pv}}$	PV generation forecast	0–8.474	kW
$P_{i,t}^{\text{L}}$	Load forecast	0.050–9.822	kW
$\pi_{i,t}^{\text{b}}$	Purchase price	0.1034–0.2314	EUR/kWh
$\pi_{i,t}^{\text{s}}$	Selling price	0.045	EUR/kWh
η^{ch}	Charging efficiency	1.0	–
η^{dch}	Discharging efficiency	1.0	–

5.13.2. Optimization Outputs

The outputs generated by the optimization algorithm are summarized in Table 4. These results capture the VPP's optimized operational decisions across key dimensions, including energy procurement, dispatch scheduling, and battery charge-discharge management. The output variables encompass energy transaction quantities, storage utilization levels, and associated cost metrics—each playing a critical role in shaping real-time control strategies. By delivering precise, time-resolved trajectories for energy flows and battery states, the optimization outputs enable the VPP to fulfill demand requirements, minimize operational costs, and maintain system balance. These actionable results provide a foundation for data-driven, cost-effective energy management at scale.

Table 4. Optimization outputs in vector notation.

Parameter	Description	Unit
$P^{\mathrm{b}}_{i,t}$	Power to be purchased at t	kW
$P^{\mathrm{s}}_{i,t}$	Power to be dispatched at t	kW
$P^{\mathrm{s,np}}_{i,t}$	Power discarded (non-remunerated)	kW
$P^{\mathrm{ch}}_{i,t}$	Power charged to the battery	kW
$P^{\mathrm{dch}}_{i,t}$	Power discharged from the battery	kW
$P^{\mathrm{exp}}_{i,t}$	Total power exported	kW
$E_{i,t}$	Battery state at t	kWh

6. Results and Discussion

The sensitivity of the GPU-based MC-SA solver to its two primary algorithmic hyperparameters—the number of iterations per chain (representing sequential workload) and the number of parallel chains (representing parallel workload)—was evaluated through a full factorial experiment. Iteration counts were sampled logarithmically from 10^2 to 10^5, while the number of chains varied from 1 to 200. Three different fleet sizes were tested (250, 500, and 1000 prosumers), and each configuration was repeated five times using independent random seeds, resulting in a total of 960 experimental runs.

The experiment measured two key response variables: the final objective value, representing the total operating cost to be minimized; and the elapsed wall-clock time, recorded on an NVIDIA A100 GPU.

6.1. Heat-Map Analysis

The heat-map analysis in Figure 5 reveals three critical patterns with significant implications for VPP operations. First, the monotonic improvement in objective value with increasing iterations or chains confirms the fundamental trade-off between computational resources and optimization accuracy. Second, the clear diminishing returns observed beyond approximately 10^4 iterations and 100 chains provides crucial guidance for operational efficiency: excessive computational investment yields minimal marginal benefits.

Most importantly, the downward-sloping iso-fitness contours in Figure 4C demonstrate a practical substitution effect between sequential iterations and parallel chains. This finding has direct operational significance: it indicates that investment in parallel computing infrastructure can compensate for limited optimization time windows. For instance, our results suggest that configurations with moderate chain counts (e.g., 50 chains) can achieve comparable solution quality to sequential approaches with substantially fewer iterations, enabling faster decision-making crucial for real-time VPP operations in dynamic market environments. This substitution effect is quantified by our empirical power-law relationship, providing VPP operators with a principled method for algorithm tuning based on their specific computational constraints and accuracy requirements.

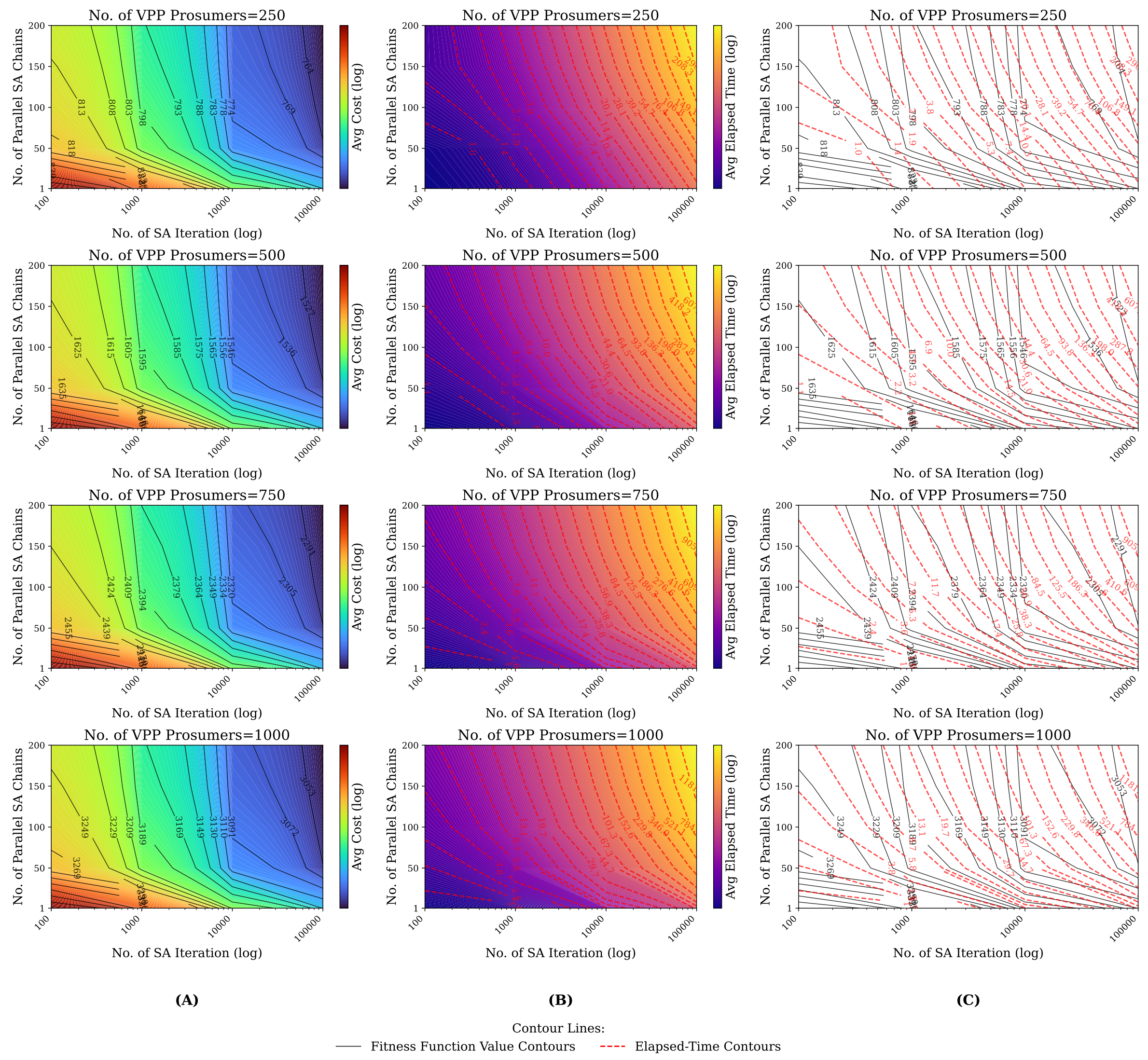

Figure 5. Heatmap visualization of MC-SA solver performance across three VPP fleet sizes (250, 500, and 1000 prosumers). Column A presents the final objective values (fitness function), highlighting solution quality across combinations of iteration counts and parallel chain counts. Column B shows the corresponding elapsed wall-clock time (in seconds), reflecting computational cost. Column C overlays iso-fitness contours on the time surface, illustrating trade-offs and substitution effects between iterations and chains in achieving equivalent solution quality with different computational budgets.

6.2. Time–Quality Pareto fronts

To translate the heat-map insights into practitioner guidelines, the raw data were replotted as Pareto curves of *objective value versus run-time* (Figure 6). Each coloured curve corresponds to a fixed chain count; tick marks annotate selected iteration budgets. Key observations are:

- A knee appears consistently around *50 chains*, where the solver reaches the 1 % optimality gap in <250 ms for 250 prosumers and <1 s for 1000 prosumers.
- Increasing the chain count beyond roughly 120 provides little benefit—and can even increase run-time for the largest fleet as the kernel spills registers and contends for memory bandwidth.
- At a *fixed* iteration budget, raising the number of chains always improves accuracy; the average gain is 8–13 % when going from 1 to 50 chains.

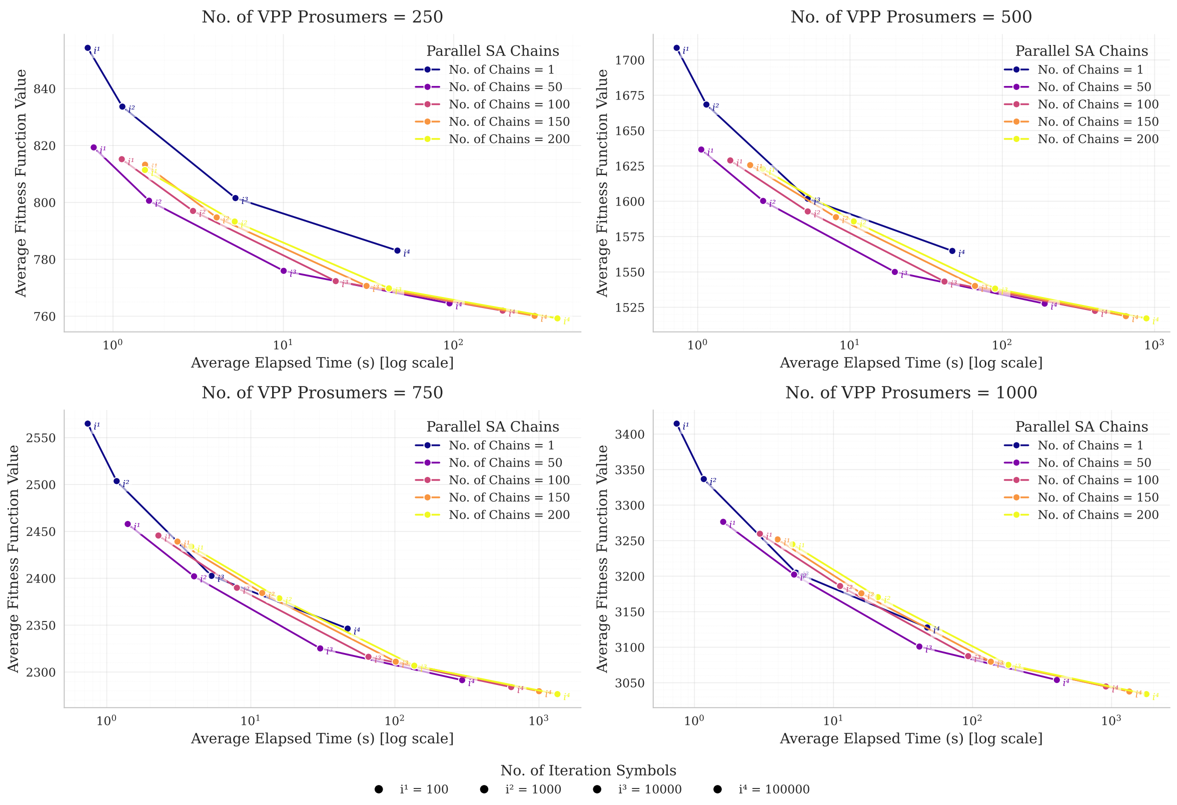

Figure 6. Pareto curves illustrating the trade-off between solution quality and execution time for different chain counts and iteration budgets. The plots highlight diminishing returns and show how parallel chains improve accuracy or reduce runtime, enabling efficient tuning of MC-SA for real-time VPP scheduling.

6.3. Empirical Chain–Iteration Law

Focusing on the iso-contour that corresponds to a 5 % optimality gap, a log–log least-squares fit of *iterations* versus *chains*[1] yields

$$\text{iterations} \;=\; \kappa\,(\text{chains})^{-0.88\pm0.17}, \qquad \kappa \simeq 1.22\times10^{5},$$

with a coefficient of determination $R^2 = 0.87$. Hence doubling the number of chains allows the sequential iteration budget to be reduced by a factor of $2^{-0.88} \approx 1.85$ while maintaining the same solution quality. This empirical power law corroborates the hypothesis that parallel chains effectively mitigate the cooling-schedule bottleneck of classical SA.

6.4. Practical Tuning Rule

Although a formal convergence monitor (e.g. the Gelman–Rubin diagnostic) could be employed to auto-terminate chains, the present data indicate that a simple heuristic suffices: *launch about 50 chains and set the iteration budget according to the above power-law*. This single parameter delivers near-optimal performance across all problem sizes tested.

6.5. Implications for Real-Time VPP Control

With the recommended setting the GPU solver re-optimises a 1000-prosumer VPP over a 15-minute rolling horizon in under one second—fast enough to react to intraday price updates or short-term PV forecast errors. Because the chain–iteration relationship is architecture-agnostic, the methodology can be transferred to other agent-based energy applications that exhibit many-agent symmetry.

[1] For every chain count, the smallest iteration budget whose average cost is within 5 % of the best value observed for that fleet size was selected.

6.6. Limitations and Future Work

The performance figures presented here were obtained on an NVIDIA A100; GPUs with fewer registers or lower memory bandwidth may shift the optimal chain count or iteration budget. In addition, a systematic benchmark against MILP ground truth for the largest instances remains outstanding. Future work will therefore proceed along three complementary lines:

(i) *Adaptive termination*: Integrate on-line convergence diagnostics (e.g. Gelman–Rubin or effective-sample-size estimates) so that each chain stops as soon as statistical equilibrium is detected.

(ii) *Hybrid refinement*: Couple MC-SA with a fast local optimiser—such as sequential quadratic programming or a MILP warm-start—to close the residual optimality gap without compromising real-time execution.

(iii) *Precision–performance co-design*: Extend the experimental matrix to include *numerical precision* as an additional factor. Concretely, we will explore mixed-precision arithmetic, fixed-point quantisation, and reduced accuracy in both *problem data* (price signals, forecasts) and *algorithmic state variables* (temperature, objective increments). The goal is to determine, jointly with the hyper-parameters studied here, the *minimal bit-width* that preserves solution quality while further reducing run-time and energy consumption.

By coupling algorithmic hyper-parameter tuning with precision-aware optimisation, we expect to push the time-to-solution envelope well below the sub-second mark, even on mid-range GPU hardware.

7. Conclusions

This study presented a scalable, GPU-accelerated implementation of MC-SA for efficient VPP scheduling. By exploiting CUDA-based dual-level parallelism—across independent annealing chains and within prosumer-level decision variables—the proposed solver enables faster and more flexible exploration of the solution space.

Through a comprehensive set of experiments, we demonstrated that increasing the number of parallel chains can effectively substitute for a portion of the sequential annealing iterations, reducing the impact of the inherent serial nature of classical simulated annealing. An empirical power-law relationship between chain count and required iterations was observed and quantified, providing a principled approach to tuning the algorithm's parallelism for a given accuracy target.

The results consistently show that launching approximately 50 chains provides a practical balance between execution time and solution quality across varying VPP sizes. In particular, the solver achieves sub-second execution times for fleet sizes up to 1000 prosumers under a 15-minute rolling horizon, demonstrating feasibility for real-time applications.

Future work will focus on incorporating adaptive convergence diagnostics, hybrid post-processing methods, and precision-aware algorithmic variants to further improve performance, robustness, and energy efficiency. Overall, the proposed MC-SA framework offers a promising and practical direction for real-time, distributed energy scheduling in large-scale smart grid environments.

Author Contributions: Conceptualization, A.A; software, A.A; investigation, A.A; writing—original draft preparation, A.A; writing—review and editing, A.A; visualization, A.A.; supervision, L. S and R. R; funding acquisition, R.R.

Funding: This research was funded by the European Union through the Recovery and Resilience Plan (PRR) as part of the Innovation Pact "NGS—New Generation Storage" (reference 58). This initiative is co-financed by NextGenerationEU under the Incentive System "Agendas para a Inovação Empresarial" ("Agendas for Business Innovation").

Data Availability Statement: The data and code used in this study are proprietary and cannot be shared due to confidentiality agreements with the private company involved. As a result, they are not publicly available

Acknowledgments: The authors gratefully acknowledge the support of the high-performance computing infrastructure provided through the project "High-Performance Computing for Enhanced Virtual Power Plant Efficiency" (reference 2024.00036.CPCA.A1). This infrastructure has been instrumental in facilitating the computational tasks required for this research. The project has been assigned the DOI https://doi.org/10.54499/2024.00036.CPCA.A1.

Conflicts of Interest: All authors have read and agreed to the published version of the manuscript.

References

1. Omelčenko, V.; Manokhin, V. Optimal Balancing of Wind Parks with Virtual Power Plants. *Front. Energy Res.* **2021**, *9*, 665295. https://doi.org/10.3389/fenrg.2021.665295.
2. Bolzoni, A.; Parisio, A.; Todd, R.; Forsyth, A.J. Optimal Virtual Power Plant Management for Multiple Grid Support Services. *IEEE Trans. Energy Convers.* **2020**, *36*, 1479–1490. https://doi.org/10.1109/TEC.2020.3044421.
3. Heredia, F.-J.; Cuadrado, M.D.; Corchero, C. On Optimal Participation in the Electricity Markets of Wind Power Plants with Battery Energy Storage Systems. *Comput. Oper. Res.* **2018**, *96*, 316–329. https://doi.org/10.1016/j.cor.2018.03.004.
4. Sun, H.; Liu, Y.; Qi, P.; Zhu, Z.; Xing, Z.; Wu, W. Study of Two-Stage Economic Optimization Operation of Virtual Power Plants Considering Uncertainty. *Energies* **2024**, *17*, 3940. https://doi.org/10.3390/en17163940.
5. Tang, W.; Yang, H.-T. Optimal Operation and Bidding Strategy of a Virtual Power Plant Integrated with Energy Storage Systems and Elasticity Demand Response. *IEEE Access* **2019**, *7*, 79798–79809. https://doi.org/10.1109/ACCESS.2019.2922700.
6. Yi, Z.; Xu, Y.; Wang, H.; Sang, L. Coordinated Operation Strategy for a Virtual Power Plant with Multiple DER Aggregators. *IEEE Trans. Sustain. Energy* **2021**, *12*, 2445–2458. https://doi.org/10.1109/TSTE.2021.3100088.
7. Ko, R.; Kang, D.; Joo, S.-K. Mixed Integer Quadratic Programming Based Scheduling Methods for Day-Ahead Bidding and Intra-Day Operation of Virtual Power Plant. *Energies* **2019**, *12*, 1410. https://doi.org/10.3390/en12081410.
8. Fernández-Muñoz, D.; Pérez-Díaz, J.I. Optimisation Models for the Day-Ahead Energy and Reserve Self-Scheduling of a Hybrid Wind–Battery Virtual Power Plant. *J. Energy Storage* **2023**, *57*, 106296. https://doi.org/10.1016/j.est.2022.106296.
9. Yoon, S.-J.; Ryu, K.-S.; Kim, C.; Nam, Y.-H.; Kim, D.-J.; Kim, B. Optimal Bidding Scheduling of Virtual Power Plants Using a Dual-MILP (Mixed-Integer Linear Programming) Approach under a Real-Time Energy Market. *Energies* **2024**, *17*, 3773. https://doi.org/10.3390/en17153773.
10. Castillo, A.; Flicker, J.; Hansen, C.W.; Watson, J.-P.; Johnson, J. Stochastic Optimisation with Risk Aversion for Virtual Power Plant Operations: A Rolling Horizon Control. *IET Gener. Transm. Distrib.* **2019**, *13*, 2063–2076. https://doi.org/10.1049/iet-gtd.2018.5834.
11. Nadeem, F.; Goswami, A.K.; Tiwari, P.K.; Pushkarna, M.; Bandhu, D.; Alhazmi, M. Multistage Scheduling of VPP under Distributed Locational Marginal Prices and LCOE Evaluation. *IEEE Access* **2024**, *12*, Article ID 3446035. https://doi.org/10.1109/ACCESS.2024.3446035.
12. Lee, J.; Won, D. Optimal Operation Strategy of Virtual Power Plant Considering Real-Time Dispatch Uncertainty of Distributed Energy Resource Aggregation. *IEEE Access* **2021**, *9*, 56965–56983. https://doi.org/10.1109/ACCESS.2021.3072550.
13. Rashidizadeh-Kermani, H.; Vahedipour-Dahraie, M.; Shafie-khah, M.; Osório, G.J.; Catalão, J.P.S. Optimal Scheduling of a Virtual Power Plant with Demand Response in Short-Term Electricity Market. In Proceedings of the *2020 IEEE 20th Mediterranean Electrotechnical Conference (MELECON)*, Palermo, Italy, 16–18 June 2020; pp. 599–604. https://doi.org/10.1109/MELECON48756.2020.9140502.
14. Zhou, B.; Liu, X.; Cao, Y.; Li, C.; Chung, C.Y.; Chan, K.W. Optimal Scheduling of Virtual Power Plant with Battery Degradation Cost. *IET Gener. Transm. Distrib.* **2016**, *10*, 712–725. https://doi.org/10.1049/iet-gtd.2015.0103.
15. Nadeem, F.; Goswami, A.K.; Tiwari, P.K. Security Constrained Two-Stage Scheduling of Virtual Power Plant under Distributed Locational Marginal Prices. In Proceedings of the *2022 IEEE International Conference on Power Electronics, Smart Grid, and Renewable Energy (PESGRE)*, Trivandrum, India, 2–5 January 2022; pp. 1–6. https://doi.org/10.1109/PESGRE52268.2022.9715929.
42. Abbasi, A.; Alves, F.; Ribeiro, R.A.; Sobral, J.L.; Rodrigues, R. Optimizing Virtual Power Plants with Parallel Simulated Annealing on High-Performance Computing. *Smart Cities* **2025**, *8*, 47. https://doi.org/10.3390/smartcities8020047.

17. Lezama, F.; Faia, R.; Soares, J.; Faria, P.; Vale, Z. Learning Bidding Strategies in Local Electricity Markets Using Ant Colony Optimization. In Proceedings of the *2020 IEEE Congress on Evolutionary Computation (CEC)*, Glasgow, UK, 19–24 July 2020; pp. 1–8. https://doi.org/10.1109/CEC48606.2020.9185520.
18. Kamarposhti, M.A.; Shokouhandeh, H.; Lee, Y.; Kang, S.-K.; Colak, I.; Barhoumi, E.M. Optimizing Energy Management in Microgrids with Ant Colony Optimization: Enhancing Reliability and Cost Efficiency for Sustainable Energy Systems. *Int. J. Low-Carbon Technol.* **2024**, *19*, 2848–2856. https://doi.org/10.1093/ijlct/ctae230.
19. Azimi, M.; Foroughi, M.; Foroughi, M.H.; Moeini-Aghtaie, M.; Yousefi, H.; Hadi, M.B. Optimizing Virtual Power Plant Performance through Three-Phase Power Flow Analysis and TCAS Algorithm. *Environ. Energy Econ. Res.* **2024**, *8*, eS087. https://doi.org/10.22097/eeer.2024.453637.1320.
20. Guan, X.; Wan, X.; Choi, B.-Y.; Song, S. Ant Colony Optimization Based Energy Efficient Virtual Network Embedding. In Proceedings of the *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, Niagara Falls, ON, Canada, 5–7 October 2015; pp. 273–278. https://doi.org/10.1109/CloudNet.2015.7335321.
21. Bremer, J.; Lehnhoff, S. Ant Colony Optimization for Feasible Scheduling of Step-Controlled Smart Grid Generation. *Swarm Intell.* **2021**, *15*, 403–425. https://doi.org/10.1007/s11721-021-00204-7.
22. Sousa, T.; Soares, T.; Morais, H.; Castro, R.; Vale, Z. Simulated Annealing to Handle Energy and Ancillary Services Joint Management Considering Electric Vehicles. *Electr. Power Syst. Res.* **2016**, *136*, 383–397. https://doi.org/10.1016/j.epsr.2016.03.031.
23. Cong, Z.; Wang, W.; Zhang, J.; Li, Y.; Sun, Y.; Lin, W. Communication Resource Scheduling of Virtual Power Plant Based on Improved Simulated Annealing Algorithm. In Proceedings of the *2024 IEEE 2nd International Conference on Electrical, Automation and Computer Engineering (ICEACE)*, Beijing, China, 22–24 March 2024; pp. 1081–1086. https://doi.org/10.1109/ICEACE63551.2024.10899010.
24. Mohanty, S.; Panda, S.; Sahu, B.K.; Rout, P.K. A Genetic Algorithm-Based Demand Side Management Program for Implementation of Virtual Power Plant Integrating Distributed Energy Resources. In *Innovation in Electrical Power Engineering, Communication, and Computing Technology: Proceedings of Second IEPCCT 2021*; Springer: Singapore, 2021; pp. 469–481. https://doi.org/10.1007/978-981-16-7076-3_41.
25. González-Romera, E.; Romero-Cadaval, E.; Roncero-Clemente, C.; Milanés-Montero, M.-I.; Barrero-González, F.; Alvi, A.-A. A Genetic Algorithm for Residential Virtual Power Plants with Electric Vehicle Management Providing Ancillary Services. *Electronics* **2023**, *12*, 3717. https://doi.org/10.3390/electronics12173717.
26. Amissah, J.; Abdel-Rahim, O.; Mansour, D.-E.A.; Bajaj, M.; Zaitsev, I.; Abdelkader, S. Developing a Three Stage Coordinated Approach to Enhance Efficiency and Reliability of Virtual Power Plants. *Sci. Rep.* **2024**, *14*, 13105. https://doi.org/10.1038/s41598-024-63668-7.
27. Ren, L.; Peng, D.; Wang, D.; Li, J.; Zhao, H. Multi-Objective Optimal Dispatching of Virtual Power Plants Considering Source-Load Uncertainty in V2G Mode. *Front. Energy Res.* **2023**, *10*, 983743. https://doi.org/10.3389/fenrg.2022.983743.
28. Hannan, M.A.; Abdolrasol, M.G.M.; Faisal, M.; Ker, P.J.; Begum, R.A.; Hussain, A. Binary Particle Swarm Optimization for Scheduling MG Integrated Virtual Power Plant toward Energy Saving. *IEEE Access* **2019**, *7*, 107937–107951. https://doi.org/10.1109/ACCESS.2019.2933010.
29. Fang, F.; Yu, S.; Xin, X. Data-Driven-Based Stochastic Robust Optimization for a Virtual Power Plant with Multiple Uncertainties. *IEEE Trans. Power Syst.* **2021**, *37*, 456–466. https://doi.org/10.1109/TPWRS.2021.3091879.
30. Li, J.; Li, S.; Wu, Z.; Yang, Z.; Yang, L.; Sun, Z. Two-Stage Multi-Objective Optimal Scheduling Strategy for the Virtual Power Plant Considering Flexible CCS and Virtual Hybrid Energy Storage Mode. *J. Energy Storage* **2024**, *103*, 114323. https://doi.org/10.1016/j.est.2024.114323.
31. Wang, X.; Chen, C.; Shi, Y.; Chen, Q. Multi-Objective Two-Stage Optimization Scheduling Algorithm for Virtual Power Plants Considering Low Carbon. *Int. J. Low-Carbon Technol.* **2024**, *19*, 773–779. https://doi.org/10.1093/ijlct/ctae031.
32. Lezama, F.; Soares, J.; Faia, R.; Vale, Z.; Kilkki, O.; Repo, S.; Segerstam, J. Bidding in Local Electricity Markets with Cascading Wholesale Market Integration. *Int. J. Electr. Power Energy Syst.* **2021**, *131*, 107045. https://doi.org/10.1016/j.ijepes.2021.107045.
33. Rädle, S.; Mast, J.; Gerlach, J.; Bringmann, O. Computational Intelligence Based Optimization of Hierarchical Virtual Power Plants. *Energy Syst.* **2021**, *12*, 517–544. https://doi.org/10.1007/s12667-020-00382-z.

34. Liu, C.; Yang, R.J.; Yu, X.; Sun, C.; Rosengarten, G.; Liebman, A.; Wakefield, R.; Wong, P.S.P.; Wang, K. Supporting Virtual Power Plants Decision-Making in Complex Urban Environments Using Reinforcement Learning. *Sustain. Cities Soc.* **2023**, *99*, 104915. https://doi.org/10.1016/j.scs.2023.104915.
35. Higuera-Gutiérrez, G.; Kazemtabrizi, B.; Shahbazi, M. Convex Flexible Branch Model (CFBM): Convex Model for Solving AC/DC Hybrid Networks Optimal Power Flows. *Int. J. Electr. Power Energy Syst.* **2025**, *170*, 110785. https://doi.org/10.1016/j.ijepes.2025.110785.
36. Liang, Z.; Chung, C.Y.; Wang, Q.; Chen, H.; Yang, H.; Wu, C. Fortifying Renewable-Dominant Hybrid Microgrids: A Bi-Directional Converter Based Interconnection Planning Approach. *Engineering* **2025**. https://doi.org/10.1016/j.eng.2025.02.020.
37. Zhuo, Y.; Zhang, T.; Du, F.; Liu, R. A Parallel Particle Swarm Optimization Algorithm Based on GPU/CUDA. *Appl. Soft Comput.* **2023**, *144*, 110499. https://doi.org/10.1016/j.asoc.2023.110499.
38. Han, W.; Li, H.; Gong, M.; Li, J.; Liu, Y.; Wang, Z. Multi-Swarm Particle Swarm Optimization Based on CUDA for Sparse Reconstruction. *Swarm Evol. Comput.* **2022**, *75*, 101153. https://doi.org/10.1016/j.swevo.2022.101153.
39. Silva, B.; Lopes, L.G.; Mendonça, F. Multithreaded and GPU-Based Implementations of a Modified Particle Swarm Optimization Algorithm with Application to Solving Large-Scale Systems of Nonlinear Equations. *Electronics* **2025**, *14*, 584. https://doi.org/10.3390/electronics14030584.
40. Charilogis, V.; Tsoulos, I.G.; Tzallas, A. An Improved Parallel Particle Swarm Optimization. *SN Comput. Sci.* **2023**, *4*, 766. https://doi.org/10.1007/s42979-023-02227-9.
41. Ferreiro, A.M.; García, J.A.; López-Salas, J.G.; Vázquez, C. An Efficient Implementation of Parallel Simulated Annealing Algorithm in GPUs. *J. Glob. Optim.* **2013**, *57*, 863–890. https://doi.org/10.1007/s10898-012-9979-z.
42. Abbasi, A.; Alves, F.; Ribeiro, R.A.; Sobral, J.L.; Rodrigues, R. Optimizing Virtual Power Plants with Parallel Simulated Annealing on High-Performance Computing. *Smart Cities* **2025**, *8*, 47. https://doi.org/10.3390/smartcities8020047.
43. Lee, S.; Kim, S.B. Parallel Simulated Annealing with a Greedy Algorithm for Bayesian Network Structure Learning. *IEEE Trans. Knowl. Data Eng.* **2019**, *32*, 1157–1166. https://doi.org/10.1109/TKDE.2019.2899096.