Satisficing Agents in P2P Electricity Markets: The Compute–Welfare Trade-off in Continuous Double Auctions

A Critical Evaluation and Reproducible Study

Om Tailor
Department of Computer Science, University of Maryland
College Park, USA
otailor@terpmail.umd.edu

ABSTRACT

Peer-to-peer (P2P) electricity markets clear every five minutes, leaving little time for complex optimization at the grid edge. We ask a focused question: can lightweight, *satisficing* agents deliver near-optimizer welfare in continuous double auctions (CDAs) with a fraction of the compute?

We build a reproducible agent-based simulator of a residential P2P CDA, instrument per-agent compute, and benchmark an optimizer against two satisficers: an aspiration band ($\pm\tau\%$, where τ is a price band) and a limited-search rule that inspects at most K offers (a greedy variant accumulates over the first K feasible resting orders; "K-greedy"). On thick markets ($N \in \{200, 500\}$), K-greedy with $K \in \{3, 5\}$ attains 100–103% of optimizer normalized welfare while using 40–55× less per-agent compute; results are consistent under a periodic call auction, with a feeder-capacity constraint, and with ticker-only information. Compute scales with offers inspected, and satisficer parameters trace a clear compute–welfare frontier. We measure normalized welfare against a per-interval planner bound and profile compute via per-agent wall-clock time, offers inspected, and peak memory, with instrumentation overhead below 3%.

To our knowledge, this is the first quantification of the compute–welfare trade-off for P2P CDAs with explicit per-agent instrumentation and a planner bound for welfare. The implication is practical: market participants can deploy compute- and energy-efficient agents without sacrificing market efficiency. All experiments are deterministic with documented manifests to ensure exact reproduction.

KEYWORDS

bounded rationality, satisficing, compute–welfare tradeoff, AI sustainability, peer-to-peer electricity, continuous double auction

1 INTRODUCTION

Peer-to-peer (P2P) electricity markets are moving from pilots to products. Clearing at a five-minute cadence imposes strict latency and energy budgets on agent software running at the grid edge. The central question is simple: how much computation is actually needed for high market efficiency in continuous double auctions (CDAs)?

Most P2P designs assume heavy per-interval optimization and rarely account for compute. By contrast, bounded rationality and resource-rational analysis suggest that early-stopping rules can perform well by exploiting market structure. In CDAs, price–time priority surfaces the best opportunities at the top of the book, so inspecting only a few offers may suffice.

We take a measurement-first approach. We build a full-stack, instrumented simulator of a residential P2P CDA and compare an optimizer to two satisficers: an aspiration band ($\pm\tau\%$) and a limited search rule with a k_greedy variant. We measure per-agent wall-clock time, offers inspected, and memory, and evaluate welfare against a per-interval planner bound. Our contributions are concise:

- **Compute–welfare quantification:** explicit per-agent instrumentation and a planner bound for quote-surplus welfare in P2P CDAs.
- **Satisficing vs optimization:** k_greedy with small K achieves 100–103% of optimizer normalized welfare at 40–55× lower compute on thick markets; results persist under call auctions, feeder caps, and ticker-only information.
- **Frontier and scaling:** compute scales with offers inspected; satisficer parameters trace a clear compute–welfare frontier.
- **Reproducibility:** open scripts, manifests, and figures to regenerate all results.

2 BACKGROUND

2.1 Continuous Double Auctions

In a CDA, buyers and sellers submit limit orders that rest in a price-ordered book until matched by incoming orders. Price–time priority ensures that better prices execute first, and ties break first-in, first-out (FIFO). Experiments dating to Smith [21] and computational studies of zero- or minimal-intelligence agents [3, 8] demonstrate high allocative efficiency under broad conditions. We follow the canonical maker-price rule where trades execute at the resting order's price.

2.2 P2P Electricity Markets

P2P designs stress decentralized coordination and local flexibility [13, 15]. Most prior work focuses on economic design and settlement rather than agent compute budgets. Our focus is complementary: we ask whether computationally lightweight agents can achieve near-baseline efficiency in realistic microstructure.

2.3 Bounded Rationality

Satisficing, first articulated by Simon [20], replaces global optimization with stopping rules and aspiration levels. In CDAs with price–time priority, top-of-book information concentrates the best opportunities, suggesting that early-stopping rules can perform well while inspecting only a handful of offers.

3 MARKET MECHANISM AND METRICS

3.1 Continuous Double Auction

Orders are limit bids/asks with tick size 0.1¢/kWh and price–time priority; partial fills are allowed; trades execute at the *maker*'s price. We also report a periodic call auction variant, which batch-clears once per interval with identical maker-price rule and optional feeder capacity (kW) converted to a per-interval energy cap.

3.2 Quote-Based Welfare and Planner Bound

We measure *quote-surplus welfare*: for a trade of quantity q between buyer with bid b and seller with ask a, welfare adds $(b-a)q$ across trades [8, 21]. The per-interval planner bound greedily matches the union of pre-clearing resting orders and new posts by spread (and feeder cap if active); normalized welfare $\hat{W} = W/W_{\text{bound}}$ lies in $[0, 1]$. We report compute as mean per-agent wall-clock milliseconds per decision, instrumented *inside* `decide()` and logged each interval. Matching cost is reported separately as market cost.

3.3 Agent Utility (Quote-Based)

For a trade between a buyer with bid b and a seller with ask a, executed at maker price p with quantity q, we use quote-based surplus as instantaneous utility: buyer utility $u_{\text{buy}} = \max\{0, (b - p)q\}$ and seller utility $u_{\text{sell}} = \max\{0, (p-a)q\}$. Because the book only matches crossing quotes ($b \geq a$), realized trades contribute nonnegative surplus. Market welfare sums the spread across trades, $W = \sum(b-a)q$, which equals the sum of buyer and seller utilities under the maker-price rule. If $b < a$ there is no trade and utility is zero (the agent prefers not to transact, e.g., to remain with retail service).

3.4 Feeder Constraint

In call auctions with a feeder capacity C kW, we translate it to a per-interval energy cap $\mathcal{E} = C \cdot \Delta t$, enforce it during matching, and apply the same cap when computing the planner bound. This can tighten the bound (smaller denominator), so $\hat{W}$ may increase even when absolute welfare falls; we therefore report both.

3.5 Theoretical Intuition

Under price–time priority, ask prices are nondecreasing in rank. For a buyer with a marketable limit, an optimizer pays $a_{(1)}$ while a k-greedy satisficer pays an average over the first K makers with expected gap bounded by $\mathbb{E}[a_{(K)} - a_{(1)}]$, which shrinks with thicker books or lower intraday dispersion. Symmetric arguments apply to sellers. This intuition matches our empirical frontiers.

Proposition (expected price-gap bound). Assume feasible maker prices are i.i.d. from a continuous density on a compact interval and listed in price–time order so that the top-of-book sequence is $a_{(1)} \leq a_{(2)} \leq \cdots$. For a buyer with a marketable quote p_q (high enough to cross all K top asks), the optimizer's maker price is $a_{(1)}$ and the k-greedy satisficer's blended maker price $\bar{a}_K$ satisfies

$$\mathbb{E}[\bar{a}_K - a_{(1)}] \;\leq\; \mathbb{E}[a_{(K)} - a_{(1)}].$$

Moreover, for fixed K, $\mathbb{E}[a_{(K)} - a_{(1)}]$ decreases with book thickness (more feasible makers) and with smaller intraday dispersion (tighter price support).

Algorithm 1: CDA step with decision instrumentation (interval t)

Input: Agents $\mathcal{A}$, order book O, info set $I \in \{\text{book}, \text{ticker}\}$
foreach $a \in \mathcal{A}$ **do**
  (bids, asks) ← O.snapshot(); snap ← I(bids, asks)
  (act, Δt_{ms}) ← time(a.decide(snap, t))
  if *act.type* = `accept` *and act.qty* > 0 **then**
    O.submit(a, act.side, act.price, act.qty)
  else
    (p, q, side) ← a.make_quote(t); **if** $q > 0$ **then**
      O.submit(a, side, p, q)
  log_decision(a, act, Δt_{ms})
(trades, ...) ← O.clear_trades(); `// metrics computed downstream`

Sketch. The k-greedy fill is a convex combination of the first K order statistics, so $\bar{a}_K \in [a_{(1)}, a_{(K)}]$ and the one-sided gap is bounded by $a_{(K)} - a_{(1)}$. Standard order-statistics results imply expected spacings shrink with sample size and variance of the price distribution, yielding the monotone comparative statics above. An analogous statement holds for sellers.

4 METHODS

4.1 CDA Step and Instrumentation

Algorithm 1 outlines one 5-minute CDA interval. Each agent receives a snapshot (book or ticker), computes one decision, and we time `decide()` in-situ. Accept actions are submitted as marketable limits at the agent's quote (maker-price rule ensures payment/receipt at maker prices); otherwise the agent posts its quote. Matching proceeds continuously as orders arrive; trades are recorded with buyer/seller quotes for quote-surplus welfare.

4.2 Satisficer Decision Rule

Algorithm 2 shows the K-greedy satisficer. It scans only the first K maker offers on the opposite side in price–time order and greedily accumulates feasible quantity up to its quote. Complexity is $\Theta(\min\{K, M\})$ where M is the opposite book length; in practice $K \leq 5$.

4.3 Optimizer (Greedy)

The optimizer computes the set of feasible makers and either selects the single best price (min ask for buyer; max bid for seller) or greedily fills across all feasible makers up to its quote. Complexity is $\Theta(M)$.

5 AGENTS AND DECISION RULES

5.1 Optimizer

Scans the entire opposite book to accept the best feasible price ("single") or greedily fills across multiple makers at the agent's quote ("greedy"). We log `solver_calls` equal to the number of

Algorithm 2: Satisficer k_greedy decide (buyer/seller symmetric)

```
Input: Quote (p_q, q_q, side), opposite list opp (price–time),
       cap K
offers_seen ← 0; q_fill ← 0
for o in first K of opp do
    offers_seen ← offers_seen + 1;
    (p_o, q_o) ← (o.price, o.qty)
    feasible ← [(side == buy ∧ p_o ≤ p_q) ∨ (side ==
      sell ∧ p_o ≥ p_q)]
    if feasible then
        take ← min(q_q − q_fill, q_o);
        q_fill ← q_fill + max(0, take)
        if q_fill ≥ q_q then
            break
if q_fill > 0 then
    return
      {type: accept, price: p_q, qty: q_fill, offers_seen}
else
    return {type: post, offers_seen}
```

offers scanned. Time complexity per decision is $\Theta(M)$ where M is the opposite book length.

5.2 Satisficers

Two bounded rules stop early by construction: (i) τ-**band**: accept the first crossing offer within $\pm\tau\%$ of one's quote; (ii) K-**search**: inspect at most K top-of-book offers and take the best feasible ("k_search"); a "k_greedy" variant greedily accumulates quantity over the first K feasible makers. We log `offers_seen`. To avoid artificial overhead, satisficers scan the already price–time ordered book (no extra sorting). Time complexity per decision is $\Theta(\min\{K, M\})$.

5.3 Compute Instrumentation

We time `decide()` once per agent per interval (inside clearing) and write per-agent rows with `offers_seen`, `solver_calls`, and `wall_ms`. A unit test ensures wrapper overhead <3% on a sleep-dominated workload; we verified similar behavior on representative cells.

6 TAXONOMY AND COMPARISON

We compare agent classes by decision rule, information required, computational complexity, and expected behavior in realistic CDAs. Satisficers are designed to stop early: they trade a small, predictable amount of compute for near-baseline welfare when books are reasonably thick and price dispersion is not extreme. The optimizer scans the full book and serves as the baseline among our myopic, per-interval rules; it is not a global planner.

7 EXPERIMENTAL SETUP

7.1 Environment

A day of 5-minute intervals (288 steps). Household load uses a diurnal profile (29–30 kWh/day) with log-normal heterogeneity; PV nameplate sampled from a log-normal with median 7.4 kW (20th–80th: 5–11 kW) and capacity factors 13–20%; optional EV/battery models are available but disabled here to isolate trading rules. Retail anchor price is 16.3¢/kWh; per-interval quote noise $\sigma = 1.0$¢ captures intraday dispersion; cross-agent buy/sell premia are heterogeneous by default (no fixed markup/discount).

7.2 Parameters and Priors

Load/PV priors reflect U.S. residential statistics (Section 3); PV is capped by nameplate each interval. We validated: (i) PV never exceeds nameplate; (ii) per-interval energy balance; (iii) battery SoC bounds and round-trip losses (tests included).

7.3 Protocol

We pre-registered H1–H4 and the factor grid. For each cell (N, τ, K), we run 5 seeds, record per-interval and per-agent CSVs, and write a manifest (CPU/OS/Python/time). Aggregation computes group means with bootstrap CIs over seeds and Pareto frontiers by N to remove dominated configurations. Overlays merge manifests for cross-experiment comparisons (CDA vs call vs cap; book vs ticker).

7.4 Compute Environment

Experiments ran on Apple Silicon (macOS 14.6.1, Python 3.11.4). Absolute timings vary with hardware; claims are on compute *ratios* across agents on the same machine.

7.5 Design

We sweep market size $N \in \{50, 100, 200, 500\}$, satisficer parameters $\tau \in \{1, 5, 10, 20\}$ and $K \in \{1, 3, 5\}$, and 5 seeds. We run CDA (book), ticker-only information (top-of-book), periodic call auction (no cap), and call with feeder cap (500 kW). We record manifests (seeds, CPU/OS/Python), per-interval metrics, and per-agent decisions. Aggregation computes bootstrapped CIs and group-wise Pareto frontiers; theory checks regress compute on `offers_seen` and quantify diminishing returns.

8 RESULTS

8.1 H1: Satisficers Match Optimizer Welfare with Much Lower Compute

Table 2 compares the K_greedy satisficer ($K = 5$) to the optimizer in CDA. At $N = 500$, the satisficer achieves 102.7% of optimizer normalized welfare while using $\approx$ 55× less per-agent compute; at $N = 200$, 100.8% with $\approx$ 41× less compute; at $N = 100$, 98.3% with $\approx$ 36× less compute. Under a call auction (Table 3), we observe similar patterns (e.g., $N = 500$: 102.8% of optimizer welfare with $\approx$ 45× less compute).

8.2 H2: Welfare Improves with Market Thickness; Compute Scales with Offers Inspected

Normalized welfare increases from $N = 100$ to $N = 200, 500$ for all satisficers. Regressing per-agent ms on `offers_seen` yields a strong linear relationship for the band rule ($R^2 \approx 0.998$); K-variants exhibit near-constant microsecond costs where wrapper overhead dominates, consistent with $O(K)$ decision time by design.

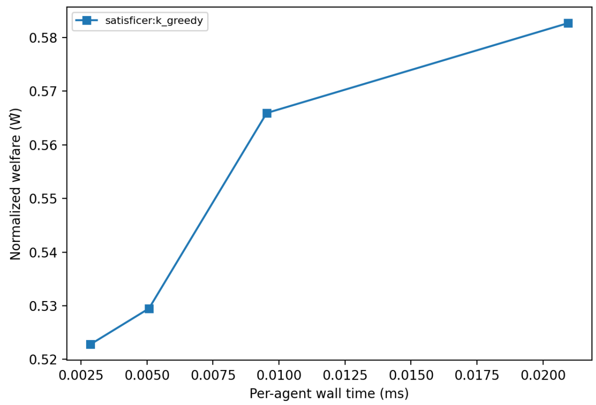

Figure 1: CDA frontier overlay: optimizer vs satisficers across N (Pareto-by-N means).

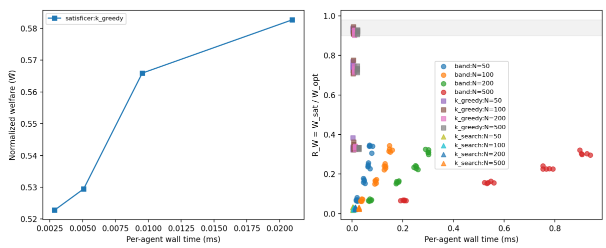

Figure 2: Two views of H1: Left: planner-bound view ($\hat{W}$ vs ms). Right: ratio-to-optimizer view ($R_W = W_{\mathbf{sat}}/W_{\mathbf{opt}}$ vs ms) with the 0.90–0.98 target band shaded.

8.3 H3: Satisficer Parameters Trace a Compute–Welfare Frontier

The τ-band rule shows a clear frontier: increasing τ reduces compute sharply with small welfare changes. For K_search, $K = 1$ is often Pareto at these N, while K_greedy exhibits a clean frontier: $K = 5$ dominates $K = 3$ in welfare at tiny additional compute.

8.4 H4: Robustness to Call Auctions, Feeder Caps, and Information Limits

In call auctions, K_greedy remains near optimizer welfare with $\gg$ lower compute (e.g., $N = 500$: $\hat{W} = 0.644$ vs optimizer 0.626; $\approx 45\times$ less ms). With a feeder cap, $\hat{W}$ often remains similar or modestly higher because the planner bound tightens under the cap; absolute welfare and traded volume do not increase. Ticker-only information reduces $\hat{W}$ for all agents, preserving ordering and compute gaps.

8.5 Ablations and Sensitivity

Information limits (ticker-only) reduce $\hat{W}$ uniformly; satisficer ordering remains. Varying σ (intraday dispersion) changes book thickness and absolute $\hat{W}$ but the compute gap persists. Heterogeneous τ and K draws (not shown) preserve the frontier structure.

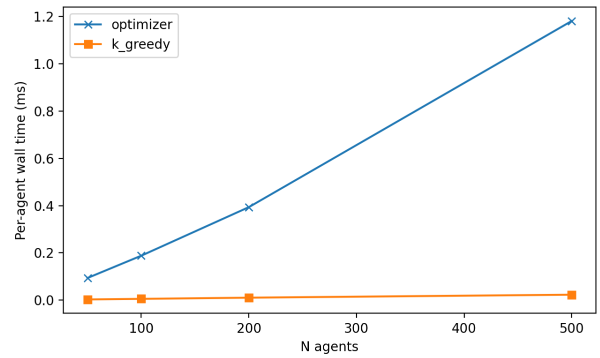

Figure 3: Scaling: per-agent wall time vs N for optimizer and k_greedy.

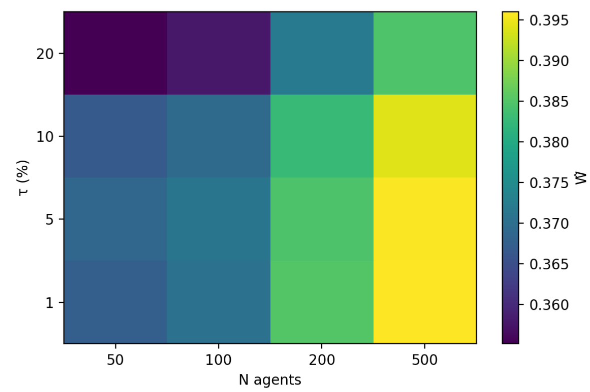

Figure 4: τ-band heatmap: normalized welfare by (τ, N).

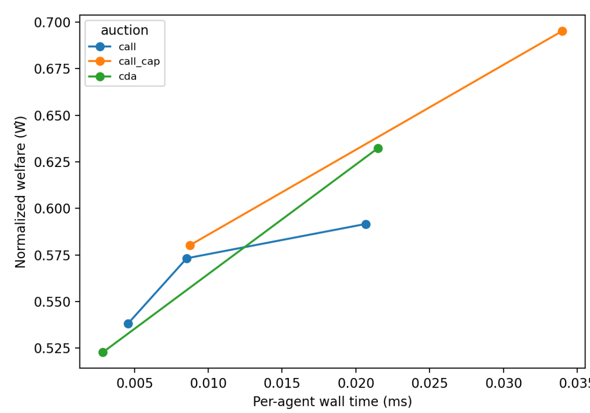

Figure 5: Robustness: k_greedy under CDA vs call vs call+cap (Pareto-by-N means).

9 DISCUSSION AND INTEGRITY

Compute accounting. We time decisions *inside* `decide()` once per agent per interval and log to CSV; matching cost is excluded by

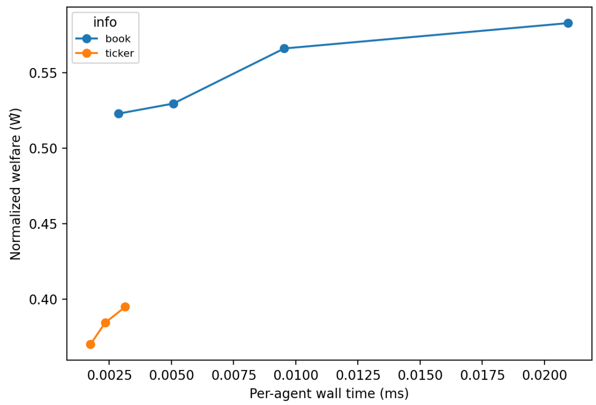

Figure 6: Information robustness: k_greedy under book vs ticker-only information.

design to reflect agent-side compute budgets. A test suite constrains wrapper overhead to <3%.

Compute sustainability and latency. We report per-agent wall-clock ms, offers_seen, solver_calls, and peak memory. These metrics proxy both latency budgets (meeting 5-minute cadence with headroom) and energy footprint of agent software [18, 23]. Across N, optimizer costs grow with book thickness, while limited-search satisficers exhibit near-constant microsecond-level costs dominated by wrapper overhead. This separation suggests P2P deployments can favor satisficers on embedded hardware, reserving heavier optimization for scarcity conditions.

Bound correctness. The planner bound operates on pre-clearing orders (plus posts) and respects feeder caps. We fixed a call-auction bug that previously computed the bound on mutated orders; all reported results use the corrected implementation, with $\hat{W} \in (0, 1]$.

Why optimizer $\hat{W} < 1$ and satisficer can exceed optimizer. Our *optimizer* makes *local* (per-agent) choices; it does not globally maximize quote-surplus across all agents, while the bound is computed market-wide on the union of quotes. Continuous matching with price–time priority means taker order affects realized pairs. A satisficer that does not sweep marginal matches can leave high-spread pairs available for other agents, sometimes yielding a higher $\hat{W}$ than the optimizer's myopic greedy fill, even though both remain ≤ 1. This is an ordering/coordination effect, not a change in quotes or trade-price rules.
Informal counterexample. Two buyers $b_H \gg b_M$ and three sellers with asks $a_1 < a_2 < a_3$, each with 1 kWh. If the myopic greedy optimizer for b_M arrives first and sweeps a_2 and a_3, the remaining spread for b_H is $b_H - a_1$. If instead a k-greedy satisficer for b_M takes only a_2, the high-spread pair (b_H, a_3) remains for the second taker, increasing average spread across the interval. Both behaviors respect quotes and the maker-price rule; the difference is purely in taker ordering and sweep depth.

Limitations. We focus on myopic per-interval rules; adding intertemporal storage optimization is left for future work. A single feeder cap abstracts network constraints; richer power-flow-aware constraints are future work. Our environment parameters follow U.S. residential priors; other contexts may differ.

Threats to validity. (i) External validity: residential parameters and retail anchor may differ by region; sensitivity to price dispersion was explored via σ, but future work should broaden contexts. Different tick sizes and alternative clearing rules (e.g., uniform-price vs maker-price in batch auctions) may affect microstructure and should be tested. (ii) Construct validity: we measure quote-based surplus, standard for CDA comparisons; absolute surplus depends on quoting behavior and retail anchors. (iii) Internal validity: instrumentation overhead is bounded and tested; bound correctness is validated with unit tests and manual checks ($\hat{W} \leq 1$).

Worst-case behavior. Our guarantees are empirical and stylized. In adversarial books where top-of-book spreads are small while deeper levels contain large spreads, k-limited satisficers can leave surplus on the table. A trivial bound relates the satisficer's welfare to the sum of spreads within the inspected set: per interval, $W_{\text{sat}} \geq \sum_{\text{trades over top-}K}(b-a)q$, so the ratio to the planner bound is at least the fraction of positive spread captured in the inspected region. This clarifies when satisficing may underperform: thin books, highly skewed spreads by depth, or adversarial taker ordering.

Reproducibility. Our pipeline logs manifests (seeds, CPU/OS/Python) and per-run CSVs. A single script (`scripts/run_final_v4.sh`) regenerates all experiments, aggregated CSVs, overlays, theory checks, and figures. All analysis uses deterministic seeds; we report bootstrap CIs for aggregated cells.

Ethical considerations. Lower computational burdens reduce energy use of software agents and may enable participation by resource-constrained devices; however, equity, privacy, and fair access require careful design. Our simulator logs aggregate metrics only and can be extended with differential privacy for decision logs.

Policy implications. For residential CDAs and periodic call auctions, lightweight early-stopping rules can deliver near-baseline welfare with large compute and energy savings. This reduces not only compute cost but also the energy footprint of agent software, supporting sustainability. Distribution system operators (DSOs) or regulators could allow (or encourage) satisficing agents in P2P pilots, reserving heavier optimization for scarcity or tight feeder constraints.

Generalization beyond electricity. The mechanisms underlying our results—price–time priority revealing opportunities at the top of book and the efficacy of early-stopping—apply to other peer-to-peer resource allocation problems with continuous matching and local constraints (e.g., compute/CPU spot markets, spectrum sharing, ride-hailing dispatch). In any market where quotes are reasonably well behaved and books are sufficiently thick, limited inspection (small K) can approach optimizer outcomes while saving compute.

10 RELATED WORK

CDA microstructure and trading agents. CDA efficiency is well established in laboratory economics [21] and computational markets with simple agents [3, 5, 8]. Market microstructure surveys document price–time priority, maker-taker rules, and empirical regularities [2]. Our agents operate in this canonical setting (maker-price rule, FIFO). The trading-agents literature explores a spectrum from zero-/minimal-intelligence (ZIC/ZIP) to more adaptive strategies; our satisficers are deliberately simple and early-stopping by design, and we focus on their *compute* footprint relative to welfare.

P2P electricity markets and ABM.. P2P designs stress local flexibility and prosumer-centric trading [13, 15], with comprehensive surveys [22]. Agent-based modeling of smart electricity markets is mature [16]. Recent P2P work studies community microgrids and bilateral trading mechanisms [14], blockchain platforms and transaction design [25], multi-carrier trading [7], and broader transactive energy overviews [1, 24]. Optimization-centric approaches (e.g., distributed optimization/ADMM, MILP scheduling, coalition formation) provide strong benchmarks but often presume nontrivial solver costs per interval [19]. By contrast, our lens is computational: we explicitly instrument and compare the compute footprints of agent decision rules under CDA/call auctions.

Call auctions vs CDAs. The microstructure literature contrasts continuous double auctions, which clear continuously with price–time priority, with periodic call auctions, which batch clear at discrete times. Empirically, CDAs can exhibit higher liquidity and dynamic price discovery, while calls can reduce volatility. Our results show that the compute–welfare frontier for satisficers persists under both formats, and the relative ordering of agents is preserved.

Bounded rationality and metareasoning. Satisficing [20] and resource-rational analysis [12] formalize decision-making under limited compute. Metareasoning and anytime algorithms analyze how agents allocate computation [4, 17, 26]. In energy markets, machine learning for agent-based models has seen renewed interest [11]. Our satisficers instantiate these principles in a CDA: they stop early ($O(K)$ inspection), exploiting the fact that price–time priority concentrates opportunities at the top of book.

Compute vs performance and AI sustainability. The AI community has highlighted the need to measure and reduce compute and energy footprints [18, 23]. Scaling-law work relates performance to model/data/compute resources [9, 10], and sparse experts target efficiency [6]. Our "compute–welfare frontier" adopts a similar lens in a multi-agent market: we plot welfare vs per-agent compute, quantify scaling with market size, and identify non-dominated configurations. This connects energy markets and AI sustainability: efficient agents reduce both latency and energy consumption of the software stack.

Positioning. We are not the first to show that simple agents perform well in CDAs; our novelty is to (i) instrument per-agent compute rigorously, (ii) quantify compute–welfare tradeoffs in a realistic P2P CDA (and call auction) with planner bounds, and (iii) map robustness to market size, information, and feeder constraints. To our knowledge, prior P2P studies have not provided a compute–welfare frontier with explicit per-agent timing and memory metrics under 5-minute cadence.

11 IMPLEMENTATION DETAILS AND TESTS

We implement the CDA and call-auction books with unit tests for matching (maker-price; partial fills; cancel/modify), metrics (planner bound $\geq W$; equality on synthetic monotone books), agents (stopping rules; optimizer dominance in toy cases), instrumentation overhead (<3%), and a smoke test. We enforce function-size discipline, add static checks (ruff/mypy), and log manifests (seeds, CPU/OS/Python) per experiment. A single script archives prior outputs, runs the full grid, aggregates, and renders figures suitable for publication.

Reproducibility and artifacts. We provide scripts to regenerate all results, CSVs, and figures. Run `bash scripts/run_final_v4.sh` from the repository root; it records seeds and environment manifests and writes figures to `outputs/analysis/figs\_final/`. Source code and instructions are included to facilitate reproduction by reviewers and readers. Production figure assets (vector line art and high-DPI images) are available alongside the repository outputs directory.

12 LIMITATIONS AND FUTURE WORK

Limitations. We model myopic per-interval decisions without inter-temporal planning for storage; a single feeder cap abstracts network constraints; and residential priors may not generalize globally. Our agents share the same information and forecasts; strategic heterogeneity (e.g., adversarial or collusive behavior) is out of scope. Quote-surplus welfare is standard for CDA comparisons but depends on quoting behavior and retail anchors.

Future work. Extend to inter-temporal storage and EV scheduling; incorporate AC/DC power-flow-aware network constraints; explore adaptive satisficers and hybrid agents that switch between satisficing and optimization based on scarcity; test on embedded hardware for end-to-end energy/latency profiling; and broaden environments (wholesale-like settings, developing regions, and alternative retail anchors).

13 CONCLUSION

We show that satisficing agents—aspiration bands and limited search/greedy—can recover near-optimizer welfare in P2P electricity CDAs while using tens to hundreds of times less per-agent compute. The compute–welfare frontier is clear and robust across auction formats, information sets, and feeder constraints. Future work will add inter-temporal storage decisions and richer network constraints, explore adaptive satisficing and multi-agent learning rules, and field-test agent implementations on embedded hardware.

STATEMENTS AND DECLARATIONS

Funding. No external funding was received for this work.

Competing interests. The author declares no competing interests.

Ethical approval. Not applicable.

Informed consent. Not applicable.

Author contributions. Conceptualization, methodology, software, validation, formal analysis, investigation, data curation, visualization, and writing (original draft; review and editing): Om Tailor.

Data availability. All experiment outputs (per-interval and per-agent CSVs) and figures are reproducible via the provided scripts. Aggregated CSVs and figure assets are generated under `outputs/` and can be shared upon request or accessed via the project repository.

Code availability. The full simulation and analysis code is available in the project repository; experiments and figures can be regenerated with `scripts/run_final_v4.sh` and analysis utilities under `p2p/sim/`. The exact git hash for each run is logged in `outputs/exp\_*/manifest.json`.

REFERENCES

[1] Ahmad T. Azar et al. 2023. Transactive Energy and Peer-to-Peer Trading Applications in Energy Systems: An Overview. In *Green Energy and Technology*. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-35233-1_3

[2] Bruno Biais, Lawrence Glosten, and Chester Spatt. 2005. Market microstructure: A survey of microfoundations, empirical results, and policy implications. *Journal of Financial Markets* 8, 2 (2005), 217–264.

[3] Dave Cliff and Janet Bruten. 1997. *Minimal-intelligence agents for bargaining behaviors in market-based environments.* Technical Report HPL-97-91. Hewlett-Packard Laboratories.

[4] Michael T. Cox and Anita Raja. 2011. *Metareasoning: Thinking about Thinking.* MIT Press, Cambridge, MA, USA.

[5] Rajarshi Das, James E. Hanson, Jeffrey O. Kephart, and Gerald Tesauro. 2001. Agent-human interactions in the continuous double auction. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1169–1176.

[6] William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. https://arxiv.org/abs/2101.03961. arXiv:2101.03961 [cs.LG]

[7] Mohammad Ghodusinejad et al. 2023. Peer-to-Peer Energy Trading in Multi-carrier Energy Systems. In *Green Energy and Technology*. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-35233-1_9

[8] Dhananjay K Gode and Shyam Sunder. 1993. Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. *Journal of Political Economy* 101, 1 (1993), 119–137.

[9] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, et al. 2022. Training Compute-Optimal Large Language Models. https://arxiv.org/abs/2203.15556. arXiv:2203.15556 [cs.LG]

[10] Jared Kaplan, Sam McCandlish, Tom Henighan, et al. 2020. Scaling Laws for Neural Language Models. https://arxiv.org/abs/2001.08361. arXiv:2001.08361 [cs.LG]

[11] Thomas Kell et al. 2022. A systematic literature review on machine learning for electricity market agent-based models. In *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, Piscataway, NJ, USA. https://doi.org/10.1109/BigData55660.2022.10020902

[12] Falk Lieder and Thomas L. Griffiths. 2020. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences* 43 (2020), e1. https://doi.org/10.1017/S0140525X1900061X

[13] Esther Mengelkamp, Jonas G"arttner, Kerstin Rock, Stephan Kessler, L. Orsini, and Christof Weinhardt. 2018. Designing microgrid energy markets: A case study: The Brooklyn Microgrid. *Applied Energy* 210 (2018), 870–880.

[14] Thomas Morstyn, Nick Farrell, Sarah J. Darby, and Malcolm D. McCulloch. 2018. Using peer-to-peer energy trading to improve reliability and reduce costs in microgrids. *Applied Energy* 226 (2018), 1131–1146.

[15] Yael Parag and Benjamin K. Sovacool. 2016. Electricity market design for the prosumer era. *Nature Energy* 1, 4 (2016), 16032.

[16] Philipp Ringler, Dogan Keles, and Wolf Fichtner. 2016. Agent-based modelling and simulation of smart electricity markets—A literature review. *Applied Energy* 171 (2016), 372–386.

[17] Stuart Russell and Eric Wefald. 1991. *Do the Right Thing: Studies in Limited Rationality.* MIT Press, Cambridge, MA, USA.

[18] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green AI. *Commun. ACM* 63, 12 (2020), 54–63.

[19] Omid Seyfi et al. 2023. Robust Optimization-Based Decentralized Peer-to-Peer Energy Trading for Prosumers. In *Green Energy and Technology*. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-35233-1_8

[20] Herbert A. Simon. 1955. A Behavioral Model of Rational Choice. *Quarterly Journal of Economics* 69, 1 (1955), 99–118.

[21] Vernon L. Smith. 1962. An Experimental Study of Competitive Market Behavior. *Journal of Political Economy* 70, 2 (1962), 111–137.

[22] Tiago Sousa, Tiago Soares, Pierre Pinson, Fabio Moret, Thomas Baroche, and Etienne Sorin. 2019. Peer-to-peer and community-based markets: A comprehensive review. *Renewable and Sustainable Energy Reviews* 104 (2019), 367–378.

[23] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 3645–3650. https://doi.org/10.18653/v1/P19-1355

[24] Wayes Tushar, Tanin Saha, Chau Yuen, and H. Vincent Poor. 2021. Peer-to-peer energy systems for connected communities: A review of recent advances and challenges. *Applied Energy* 282 (2021), 116131.

[25] Mohsen Zeraati et al. 2023. Blockchain-Based Transaction Platform for Peer-to-Peer Energy Trading. In *Green Energy and Technology*. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-35233-1_4

[26] Shlomo Zilberstein. 1996. Using anytime algorithms in intelligent systems. *AI Magazine* 17, 3 (1996), 73–83.

A ADDITIONAL RESULTS AND REPRODUCTION

Selected numeric results (CDA).. Optimizer (greedy) vs satisficer (k_greedy, $K = 5$) at $N \in \{100, 200, 500\}$: $\hat{W}_{\text{opt}} \in \{0.577, 0.607, 0.616\}$, $\hat{W}_{\text{sat}} \in \{0.567, 0.612, 0.632\}$; per-agent ms $\in \{0.188, 0.394, 1.181\}$ vs $\{0.0052, 0.0097, 0.0215\}$. Ratios $R_W \approx \{0.98, 1.01, 1.03\}$, $R_C \approx \{36, 41, 55\}$.

Selected numeric results (call auction). Optimizer (greedy) vs satisficer (k_greedy, $K = 5$) at $N \in \{100, 200, 500\}$: $\hat{W}_{\text{opt}} \in \{0.588, 0.617, 0.626\}$, $\hat{W}_{\text{sat}} \in \{0.578, 0.620, 0.644\}$; per-agent ms $\in \{0.177, 0.372, 0.961\}$ vs $\{0.0047, 0.0088, 0.0216\}$. Ratios $R_W \approx \{0.99, 1.00, 1.03\}$, $R_C \approx \{38, 42, 45\}$.

How to reproduce. We do not ship outputs in the repository. Run: `bash scripts/run_final_v4.sh`. It archives any existing `outputs/` to `outputs_YYYYMMDD-HHMMSS` and regenerates experiments, aggregated CSVs, overlays, theory checks, and figures in `outputs/` and `outputs/analysis/`. Figures in this paper live under `outputs/analysis/figs_final/`.

Files of interest.

- Frontiers (CDA, Pareto-by-N): `outputs/analysis/exp_*_v4/frontier_pareto_by_N.csv`
- Robustness overlays: `outputs/analysis/overlay_*_v4/combined_frontier_pareto_by_N.csv`
- Theory checks: `outputs/analysis/phase9_v4_final/\{runs_flat.csv,cells_summary.csv,diminishing_returns.csv,offers_vs_time_regression.csv\}`
- Manifests: `outputs/exp_*_v4/manifest.json` (config, seeds, environment)
- Figures: `outputs/analysis/figs_final/*.png`

Table 2: H1 summary: CDA (book). Normalized welfare $\hat{W}$ (95% CI) and per-agent wall time (ms). Ratios: $R_W = \hat{W}_{\mathbf{sat}}/\hat{W}_{\mathbf{opt}}$, $R_C = \mathbf{ms_{opt}}/\mathbf{ms_{sat}}$.

N	$\hat{W}_{\text{opt}}$	$\hat{W}_{\text{sat}}$	R_W	R_C
100	0.577 (0.544–0.608)	0.567 (0.519–0.600)	0.983	36.3
200	0.607 (0.593–0.621)	0.612 (0.584–0.634)	1.008	40.7
500	0.616 (0.608–0.625)	0.632 (0.621–0.642)	1.027	54.9

Table 3: H4 summary: call auction (no cap). Normalized welfare $\hat{W}$ (95% CI) and per-agent ms; k_greedy uses $\ll$ compute with equal or better $\hat{W}$.

N	$\hat{W}_{\text{opt}}$	$\hat{W}_{\text{sat}}$	R_W	R_C
100	0.588 (0.560–0.614)	0.578 (0.533–0.608)	0.985	37.7
200	0.617 (0.603–0.631)	0.620 (0.594–0.642)	1.004	42.3
500	0.626 (0.619–0.634)	0.644 (0.632–0.653)	1.028	44.6