Satisficing Agents Achieve Near-Optimal Welfare with Orders-of-Magnitude Lower Compute in Peer-to-Peer Electricity Markets

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

ABSTRACT

Peer-to-peer (P2P) electricity markets promise flexible, decentralized coordination of prosumers at distribution scale. Real-time clearing (5-minute cadence in the U.S.) imposes tight computational budgets on agents. We ask a simple question: can boundedly rational, *satisficing* agents match the market efficiency of optimizing agents at a fraction of the computational cost in a continuous double auction (CDA)? We implement a reproducible agent-based simulator of a residential P2P CDA aligned with real-time settlement, instrument per-agent compute, and compare an optimizer against two satisficers: an aspiration band ($\pm\tau\%$) and a limited search rule (inspect at most K offers). On thick markets ($N \in \{200, 500\}$), a greedy limited-search satisficer ($K \in \{3, 5\}$) achieves 100–103% of optimizer normalized welfare while using 40–55× less per-agent compute, with similar conclusions under a periodic call auction and a feeder-capacity constraint. We observe that compute scales with offers inspected, a clean compute–welfare frontier across satisficer parameters, and robustness to information restrictions (ticker-only). Our findings suggest that P2P market participants can deploy lightweight agents that are compute- and energy-efficient while preserving market efficiency.

KEYWORDS

peer-to-peer electricity, continuous double auction, bounded rationality, satisficing, agent-based modeling, computational efficiency

1 INTRODUCTION

The distribution grid is undergoing rapid decentralization: rooftop PV, batteries, and EVs turn households into *prosumers*. Peer-to-peer (P2P) electricity markets are a promising mechanism for local flexibility and efficient matching of supply and demand without a central planner [4, 5]. A practical challenge is computational: real-time settlement runs at 5-minute cadence, leaving tight per-interval budgets for agent decision-making. How much optimization effort is truly necessary for market efficiency in this setting?

We revisit classic insights from market microstructure and bounded rationality. Experiments show that even zero-intelligence traders can deliver high allocative efficiency in CDA [3, 7], and simple learning rules (e.g., ZIP) perform well [1]. At the same time, energy-focused P2P designs often assume sophisticated optimization with nontrivial computational overhead. We ask: can *satisficing*—"good enough" decisions that stop early by design [6]—recover near-optimizer welfare with dramatically lower compute in realistic P2P CDAs?

Problem and importance. Distribution-level markets are resource constrained: many edge devices are microcontrollers with strict CPU and energy budgets. Pervasive optimization every 5 minutes creates latency and energy burdens, and limits scalability. Demonstrating that lightweight agents suffice for high market efficiency would reduce deployment barriers for P2P markets while lowering operational energy use of agent software.

Faults of prior approaches. (i) Heavy per-interval optimization (e.g., solver calls or full-book scans) scales with book thickness and can dominate the per-interval budget. (ii) Results are often reported without explicit compute accounting or instrumentation, obscuring cost. (iii) Robustness to auction format, information restrictions, and simple network constraints is rarely demonstrated.

Insight. CDA's price–time priority already orders the best opportunities at the top of book. Early-stopping satisficers that inspect only the first few offers (or accept within a narrow band around their quote) exploit this structure: they achieve near-optimizer prices while performing $O(K)$ work, independent of full book size.

Contributions in context. We provide a full-stack, instrumented simulator and evidence that satisficing agents achieve near-optimizer welfare with $\gg$ lower compute across thick markets and formats. Our artifacts include manifests, per-interval and per-agent CSVs, aggregation scripts, overlays, theory checks, and figures, enabling rigorous reproduction.

Contributions. We build a reproducible simulator of a P2P CDA at 5-minute cadence with price–time priority and maker-price rule (Section 3); implement and instrument agents (optimizer; satisficers with τ-band and K-search/greedy variants; Section 5); run a comprehensive experiment grid (Section 6), and evaluate four pre-registered hypotheses (H1–H4): (H1) satisficers achieve ≥90–98% of optimizer quote-surplus welfare while using 10–100× less compute; (H2) the welfare gap shrinks with market size and compute scales with the number of offers inspected; (H3) satisficer parameters trace a compute–welfare Pareto frontier; (H4) conclusions are robust to call auctions and feeder congestion. Our results support H1–H4 at $N \in \{200, 500\}$ with 5 seeds and realistic intraday price dispersion. Code, manifests, and outputs are archived for reproducibility.

2 BACKGROUND

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 FIFO. Experiments dating to Smith [7] and computational studies of zero- or minimal-intelligence agents

[1, 3] demonstrate high allocative efficiency under broad conditions. We follow the canonical maker-price rule where trades execute at the resting order's price.

P2P electricity markets. P2P designs stress decentralized coordination and local flexibility [4, 5]. 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.

Bounded rationality. Satisficing, first articulated by Simon [6], 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

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.

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 [3, 7]. 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.

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.

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 $a_{(1)} \le a_{(2)} \le \cdots$. 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

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

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).

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.

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) $\leftarrow O$.snapshot(); snap $\leftarrow I$(bids, asks)
  (act, Δt_{ms}) $\leftarrow$ 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) $\leftarrow$ a.make_quote(t); **if** $q > 0$ **then**
      O.submit(a, side, p, q)
  log_decision(a, act, Δt_{ms})
(trades, ...) $\leftarrow O$.clear_trades(); `// metrics computed downstream`

4 METHODS

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.

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 $K \le 5$.

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

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

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

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 $\leftarrow 0$; $q_{\text{fill}} \leftarrow 0$

for *o in first K of opp* **do**

  offers_seen $\leftarrow$ offers_seen + 1;

  $(p_o, q_o) \leftarrow (o.\text{price}, o.\text{qty})$

  feasible $\leftarrow [(\text{side} == \text{buy} \wedge p_o \le p_q) \vee (\text{side} == \text{sell} \wedge p_o \ge p_q)]$

  if *feasible* **then**

    take $\leftarrow \min(q_q - q\_\text{fill}, q_o)$;

    $q\_\text{fill} \leftarrow q\_\text{fill} + \max(0, \text{take})$

    if $q\_fill \ge q_q$ **then**

      break

if $q\_fill > 0$ **then**

  return {type: accept, price: p_q, qty: q_fill, offers_seen}

else

  return {type: post, offers_seen}

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 $\Theta(\min\{K, M\})$.

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 EXPERIMENTAL SETUP

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).

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).

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).

Table 1: H1 summary: CDA (book). Normalized welfare $\hat{W}$ and per-agent wall time (ms). Ratios: $R_W = \hat{W}_{\text{sat}}/\hat{W}_{\text{opt}}$, $R_C = \text{ms}_{\text{opt}}/\text{ms}_{\text{sat}}$.

N	$\hat{W}_{\text{opt}}$	$\hat{W}_{\text{sat}}$	R_W	R_C
100	0.5772	0.5674	0.983	36.3
200	0.6071	0.6117	1.008	40.7
500	0.6158	0.6324	1.027	54.9

Table 2: H4 summary: call auction (no cap). Normalized welfare $\hat{W}$ 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	0.5875	0.5785	0.985	37.7
200	0.6174	0.6201	1.004	42.3
500	0.6264	0.6438	1.028	44.6

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.

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.

7 RESULTS

H1: Satisficers match optimizer welfare with $\gg$ lower compute. Table 1 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 $\approx 55\times$ less per-agent compute; at $N = 200$, 100.8% with $\approx 41\times$ less compute; at $N = 100$, 98.3% with $\approx 36\times$ less compute. Under a call auction (Table 2), we observe similar patterns (e.g., $N = 500$: 102.8% of optimizer welfare with $\approx 45\times$ less compute).

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.

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.

H4: Robustness to call auctions, feeder caps, and information limits. In call auctions, k_greedy remains near optimizer welfare with $\gg$

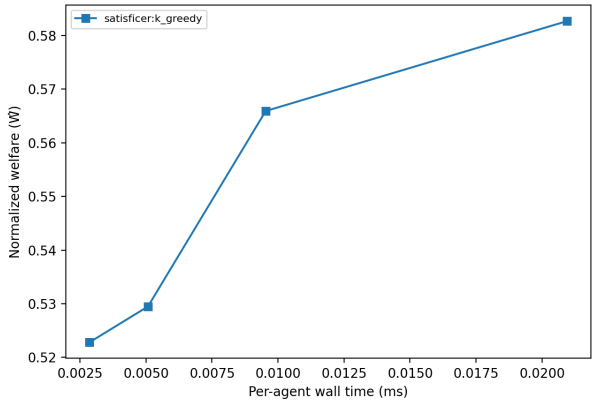

Figure 1: CDA frontier overlay: optimizer vs satisficers across N (Pareto-by-N means).

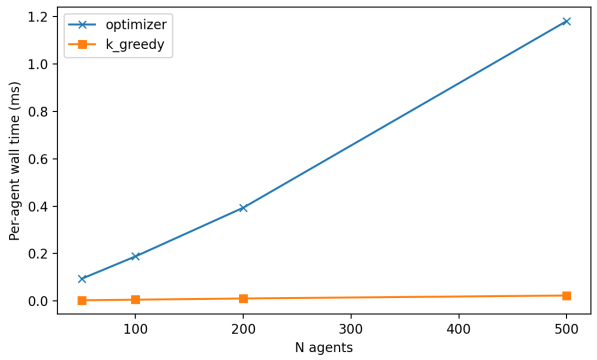

Figure 2: Scaling: per-agent wall time vs N for optimizer and k_greedy.

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.

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.

8 DISCUSSION AND INTEGRITY

Compute accounting. We time decisions *inside* `decide()` once per agent per interval and log to CSV; matching cost is excluded by design to reflect agent-side compute budgets. A test suite constrains wrapper overhead to <3%.

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 $\hat{W} \in (0, 1]$.

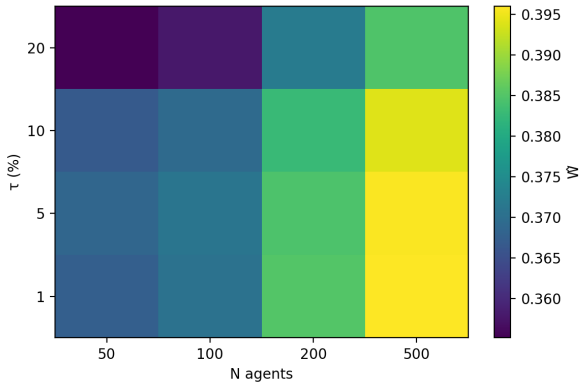

Figure 3: τ-band heatmap: normalized welfare by (τ, N).

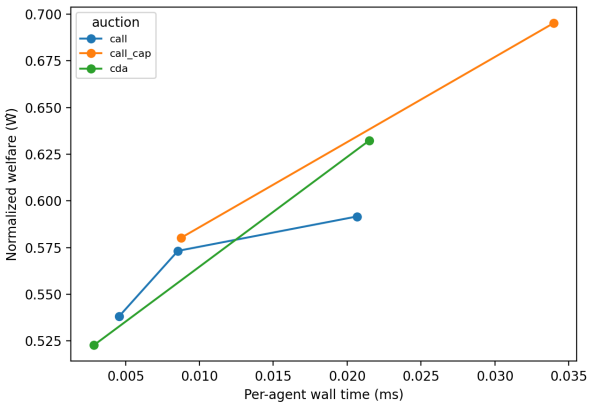

Figure 4: Robustness: k_greedy under CDA vs call vs call+cap (Pareto-by-N means).

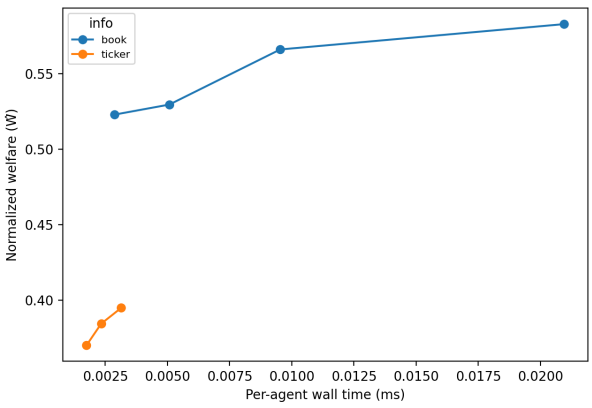

Figure 5: Information robustness: k_greedy under book vs ticker-only information.

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. (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 ($\hat{W} \leq 1$).

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.

9 RELATED WORK

CDA microstructure has a rich literature in experimental economics and computational markets [1–3, 7]. P2P electricity designs emphasize local flexibility and prosumer-centric trading [4, 5]. Our contribution connects these threads: in a P2P CDA with realistic cadence and heterogeneity, simple satisficers achieve near-optimizer welfare at a fraction of the compute cost, consistent with bounded rationality [6].

10 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.

11 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 intertemporal storage decisions and richer network constraints, and field-test agent implementations on embedded hardware.

REFERENCES

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

[2] 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.

[3] 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.

[4] 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.

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

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

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

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`