Before AI Agents Negotiate for Us: A Competitive Tournament of Rule-Based, Search, Optimization, Learned, and Language-Model Negotiators Under Scarcity

Eduardo de Avila-Armenta[1,2], Raúl Alejandro Aquino-Ibarra[2], Jorge Octavio Nicolás-Díaz[2], Umberto Leon-Dominguez[2,3]*

[1]Tecnológico de Monterrey, School of Engineering and Sciences,
Av. Eugenio Garza Sada 2501 Sur, Tecnológico, 64849 Monterrey, N.L., Mexico
[2]AI Lab, Freight Technologies Inc.
[3]School of Psychology, University of Monterrey, Monterrey, Mexico

November 11, 2026

Abstract

Automated bilateral negotiation underlies two-sided matching markets such as freight brokering, where many shippers and carriers negotiate price before a matching mechanism decides who is paired with whom. Prior strategy comparisons typically use isolated one-on-one duels or a single fixed market, so they cannot show how a strategy behaves against several live rivals under a changing balance of supply and demand. We built nine negotiation architectures spanning rule-based, search-based, optimization-based, imitation-learning, and reinforcement-learning families, and ran all nine simultaneously on both sides of a Gale-Shapley matching market under a balanced condition and two scarcity conditions rotated across all 36 exclusion pairs, with every pairwise comparison Holm-Bonferroni corrected. Eight of the nine architectures share one normalized *negotiation distance* – the position of a price between the agent's worst acceptable deal (0) and its ambitious target (1), i.e. how close the table is to a deal the agent wants – as their common scale for valuing, optimizing, or learning offers. In the balanced carrier market, an imitation-learned engine (ML2), a reinforcement-learning policy trained only on isolated negotiations (RL), and an expected-utility optimization engine (OPT) formed a statistically tied top group, and no architecture led in every condition and role. RL beat all eight carrier rivals when carriers were scarce, but its match rate collapsed from 62% to 16% when shippers were scarce and it had to compete with the other carrier agents for demand. A second reinforcement-learning policy trained against rival agents (RL2) kept its match rate above 91% in every condition, yet lost on realized payoff to all eight shipper-side rivals in two of the three conditions, traced to a confirmed train/deployment feature mismatch. We also tested nine large language models (LLMs), from 1 to 72 billion parameters, under the same protocol: they lost significantly to the two strongest engines (OPT2 and ML2) in 32 of 36 comparisons and won none. The strongest model, `llama3.3:70b`, posted the highest raw profit of all twelve shipper-side participants without reaching significance, while `deepseek-r1:70b` collapsed to a 2.5% match rate because its responses hit the 90-second timeout on our single-GPU workstation. These results show that isolated or single-condition evaluation can hide failure modes that a many-sided, scarcity-varying, statistically corrected tournament exposes directly.

Keywords: automated negotiation · multi-agent systems · two-sided matching · reinforcement learning · large language models · stable matching · tournament evaluation

*Corresponding author. Email: `umberto.leon@udem.edu`

1 Introduction

Two-sided matching markets in which many buyers and many sellers must both agree on a price and be paired with a counterpart – freight brokering, ride-hailing dispatch, labor and housing markets – are negotiated today by human brokers, by software agents, or by both, and increasingly depend on autonomous negotiation agents (this study evaluates software agents only, with no human participants) that can reach a deal quickly, on favorable terms, while remaining attractive enough to actually be the counterpart someone else chooses. A negotiation agent in such a market therefore faces two linked problems at once: winning a good price at the bilateral table, and winning the competition for a match once every table's outcome is fed into a stable-matching mechanism [7] that may discard an agreed deal in favor of a better one found elsewhere.

Work on automated negotiation strategies has produced a wide range of architectures – hand-written concession rules, value-function-guided heuristics, lookahead search over sampled opponent models, continuous expected-utility optimization, strategies that imitate a stronger policy's offers, and reinforcement-learned policies trained end to end [3, 6, 18] – but these architectures are typically compared in one of two limited ways: isolated one-on-one duels against a small number of fixed opponents, or a single population competing under one fixed market condition. Neither setting can reveal how an architecture behaves once it must negotiate against several rival architectures simultaneously, nor how its behavior changes as the balance between supply and demand shifts from a balanced market toward scarcity on one side. This gap matters because the matching stage can decouple negotiation skill from final outcome entirely: an architecture that reaches good prices in isolation can still fail to be matched at all if it is rarely the counterpart a scarce resource prefers, and a reinforcement-learned policy trained only against a fixed, non-adapting opponent has no guarantee that what it learned survives contact with genuine competitive pressure from rival architectures of its own side.

Several of this paper's hand-designed architectures – the rule-based, search-based, and optimization-based engines, whose decision logic is written explicitly rather than learned from data – encode specific findings from the negotiation-psychology literature rather than arbitrary heuristics: an ambitious opening anchor calibrated to the observed spread between the parties' positions [15, 20], a geometrically decreasing concession schedule [19], every offer framed against the counterpart's own reference point [9], and a portfolio of concurrent tables treated as leverage in the same way holding multiple outside alternatives distorts a negotiator's perceived bargaining zone [6, 17] – design choices this paper's tournament lets compete against each other directly, rather than evaluating any one of them in the isolated, single-negotiation settings the underlying behavioral studies used. Large language models are a natural further architecture to include in such a tournament: a concurrent study found that an LLM prompted to select game moves directly loses to an agent built around an explicit, LLM-synthesized world model combined with search, in the large majority of the games tested [10] – a direct-policy-versus-model-and-search pattern this paper tests again in a different domain (Section 4.6).

This paper reports a controlled tournament designed to surface exactly these effects. We implemented nine negotiation-engine architectures, spanning rule-based, value-function-augmented, search-based, continuous-optimization, imitation-learning, and reinforcement-learning designs, on both the shipper and carrier side of a bilateral freight-price negotiation, and ran all nine simultaneously – never in isolated duels – under three market conditions: a balanced market with all nine architectures present on both sides, and two scarcity conditions in which one side's roster is reduced from nine to seven architectures, with the excluded pair rotated systematically across all $\binom{9}{2} = 36$ possible exclusions so that no architecture is systematically favored or penalized by the experimental design itself. Every pairwise comparison between two architectures under a given market condition and role is tested with a paired Wilcoxon signed-rank test on same-day outcomes and corrected for the resulting 36 simultaneous comparisons with the Holm-

Bonferroni procedure, and every architecture's own outcome is summarized with a percentile bootstrap confidence interval rather than a normal-approximation interval, since margin and profit outcomes in this market are not symmetrically distributed.

The results show that formulation sophistication alone does not guarantee good outcomes under competition: a reinforcement-learned policy trained only on isolated negotiations suffers a severe match-rate collapse once rival agents (the other architectures of its own side, all of them software agents) compete with it for scarce demand, while the same architecture trained against rival agents avoids that collapse but is instead undone by a confirmed mismatch between the feature values it saw during training and the feature values it receives at deployment time, a finding that would not surface in an isolated duel. We report both findings directly, alongside an extension of the same tournament to nine large-language-model-based negotiation agents, including a qualitative look at individual negotiation trajectories reconstructed from the same call logs. Section 2 positions this tournament against prior automated-negotiation platforms, learned dialogue agents, and recent LLM-negotiation benchmarks; Section 3 describes the negotiation protocol, market model, matching mechanism, the nine architectures, the experimental design, and the statistical analysis; Section 4 reports the tournament's results; Section 5 discusses their implications, compares them against that prior work, and states limitations; Section 6 concludes.

2 Related Work

Automated negotiation has been studied through two largely separate ecosystems, each with its own standard platform and evaluation protocol, and a third, more recent line specific to large language models.

The first is the GENIUS/ANAC ecosystem [12]: a shared Java platform for specifying negotiation domains (a fixed set of issues and one private linear-additive utility function per party) and running bilateral or small multilateral sessions between submitted agents, hosted yearly as the International Automated Negotiating Agents Competition since 2010. GENIUS/ANAC's domains and its published yearly rankings are the closest thing automated negotiation has to a shared benchmark, but every session is still an isolated duel (or a small fixed multilateral group) over one utility function, with no matching stage that can discard an agreed deal and no systematic variation in how many rivals of the same side are present. de Jonge [4] shows this isolation has a concrete cost: MiCRO, a strategy with no opponent modeling or learning at all, outperforms the strongest submitted agents across four ANAC editions simply by exploiting regularities of the classic linear-additive domains themselves, a result later given a full theoretical treatment [5]. That finding argues the classic ANAC domains are too simple on the *utility-function* axis; this paper's tournament instead varies difficulty on a different axis entirely, the *market-structure* axis (how many rivals are present and how scarce the opposite side is), while keeping each individual bilateral exchange comparatively simple.

The second ecosystem is end-to-end learned negotiation dialogue, seeded by Lewis et al. [11]'s "Deal or No Deal" task (a multi-issue bargaining dialogue with hidden per-party rewards, solved with a sequence-to-sequence policy trained by self-play and rollouts) and extended by He et al. [8]'s CraigslistBargain, which decouples a coarse dialogue-act strategy from natural-language generation to make the learned strategy itself inspectable. Both trained and evaluated their agents in isolated bilateral dialogues against a fixed human or self-play distribution, never against a population of rival negotiators competing for the same scarce counterparts – the same isolation this paper's RL architecture (Section 3.4) exhibits by design, and whose cost under real competitive pressure Section 4.3 measures directly.

The third, most recent line evaluates large language models themselves as negotiators, outside any hand-designed architecture. Abdelnabi et al. [1] scores LLM agents in multi-issue negotiation games with tunable difficulty and finds a large capability gap between model gen-

erations, though a 2026 reproduction [2] finds that ranking models on this benchmark is less objective than it first appears, cautioning against reading any single LLM-negotiation leaderboard as settled. Zhang et al. [21] and Oh et al. [13] both push past a single deal-rate or win-rate summary, diagnosing surplus extraction, belief calibration, and strategy quality separately – the same spirit as this paper's own decomposition of the LLM comparison into match rate, realized margin/profit, and the qualitative trajectory case studies of Section 4.7, rather than a single pass/fail number per model. None of these three LLM-negotiation benchmarks places its agents inside a populated market with a real matching mechanism or a rival roster of hand-designed and learned (non-LLM) architectures; Lehrach et al. [10]'s finding that a direct LLM policy loses to an explicit model-and-search agent in most games tested is, to our knowledge, the closest prior result to this paper's own finding that language models lose to this tournament's two strongest engines in the large majority of comparisons (Section 4.6), but it is measured on a disjoint set of classic games, not a negotiation market.

Across all three lines, no prior evaluation places rule-based, search-based, optimization-based, imitation-learned, reinforcement-learned, and LLM-based negotiation agents under one shared, statistically corrected protocol, inside a populated market with a real matching mechanism and systematically rotated scarcity. That combination – not a harder utility function, and not a single additional agent family, but the market structure and the shared cross-paradigm comparison together – is this paper's point of departure from each of the three lines above, and is revisited directly against each of them in Section 5.1.

3 Methods

Figure 1 summarizes the full pipeline described in this section, from the market model through the three market conditions of Section 3.6 to the statistical analysis of Section 3.8.

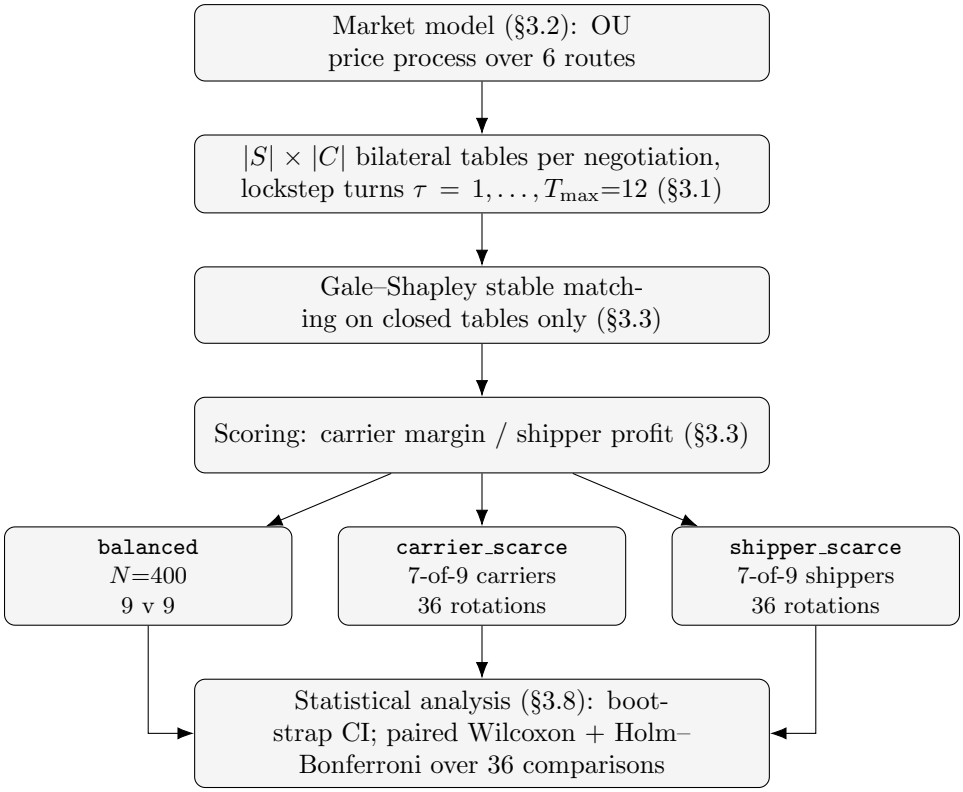


Figure 1: Full experimental pipeline, from the market model through the three market conditions (Section 3.6) to the statistical analysis (Section 3.8). S is the set of shippers and C the set of carriers in one negotiation, so $|S| \times |C|$ is the number of shipper-carrier tables negotiated in parallel; OU is the Ornstein-Uhlenbeck price process (Eq. (5)). Every negotiation runs independently through the top four stages; which of the three condition boxes it belongs to only changes how many architectures are present on which side.

3.1 Negotiation protocol

One negotiation instantiates a set of shippers S and a set of carriers C, who interact over every one of the $|S| \times |C|$ bilateral tables simultaneously, one table per shipper-carrier pair. Every table runs on the same global turn clock $\tau = 1, \ldots, T_{\max}$ with $T_{\max} = 12$: shippers act on odd τ and carriers act on even τ, so each side has exactly six decision opportunities per negotiation. At each decision, an engine receives a table state containing the full turn history of that table (every offer either side has made), the counterpart's current offer if any, and a snapshot of the agent's *other* live tables in the same negotiation (whether each is closed, in impasse, or still open, and its latest offer) – this portfolio view is the only information an architecture has about its own concurrent negotiating load, and it never sees any information about a rival architecture's tables. A carrier's legal offer range (its "hard band") is

$$\begin{aligned} RP_t &= \bar{\mu} - \tfrac{1}{2}\sigma, \\ \mathrm{Tgt}_t &= \bar{\mu} + \tfrac{1}{2}\sigma, \end{aligned} \tag{1}$$

where $\bar{\mu}$ and σ are the route's true reference-price mean and standard deviation – synthetic values drawn by the market model of Section 3.2, not taken from real freight data, and, like every price, rate, margin, and profit in this paper, expressed in US dollars (USD) – and the subscript t denotes this table (held fixed for the whole negotiation), not the turn counter τ used below; every carrier offer x must satisfy $x \in [RP_t, \mathrm{Tgt}_t]$ and must never exceed the carrier's own previous offer on that table (monotonically non-increasing concessions). A shipper's legal

offer range is bounded above by its hard reservation price and anchored below by an ambitious target,

$$\begin{aligned} RP &= p_{\text{sale}} - c_{\text{goods}}, \\ \text{Tgt} &= \widehat{RP}_{\text{est}} - \tfrac{1}{2}\sigma + k\,\sigma_\varepsilon, \end{aligned} \tag{2}$$

where p_{sale} is the shipper's sale price for the shipment, c_{goods} is its goods cost, $\widehat{RP}_{\text{est}}$ is the shipper's own noisy estimate of the route's true reference price (Section 3.2), $\sigma_\varepsilon = 0.3\,\sigma$ is that estimate's noise scale, and k is a per-engine cushion constant; every shipper offer must satisfy $x \le RP$ and must never fall below the shipper's own previous offer (monotonically non-decreasing concessions). An offer violating either the band or the monotonicity constraint is an illegal move and is treated as an automatic forfeit of that table, exactly like an explicit reject; an accept is only legal when something is actually on the table and that offer already satisfies the acceptor's own band. If a table reaches $\tau = T_{\max}$ without a deal, it ends in impasse and never enters the matching stage.

3.2 Market model

All prices in this market are synthetic and in USD. Six routes are drawn once per round – the balanced condition's 400 negotiations form one round, and each of a scarce condition's 36 sub-rounds of 12 negotiations is its own round (Section 3.6) – from

$$\begin{aligned} &\mu_r \sim \text{Unif}(800, 3000), \\ &\sigma_r = 0.15\,\mu_r, \qquad r = 1, \ldots, 6, \end{aligned} \tag{3}$$

and in every negotiation of that round each shipper is assigned one of the six routes at random; μ_r and σ_r instantiate $\bar{\mu}$ and σ in Eq. (1) for every table on that route. The shipper does not observe μ_r directly: it only observes a noisy estimate

$$\begin{aligned} \widehat{RP}_{\text{est}} &= \mu_r + \epsilon, \\ \epsilon &\sim \mathcal{N}\big(0,\ (0.3\,\sigma_r)^2\big)\,, \end{aligned} \tag{4}$$

which is the one deliberate information asymmetry in the market: the carrier's true band is exact, the shipper's belief about it is not. Sale price and goods cost are drawn once per negotiation index $n = 1, \ldots, N$ from an independent, discretized, log-space Ornstein-Uhlenbeck process for each quantity,

$$\begin{aligned} \log P_n &= \log P_{n-1} + \kappa\,(\log P_0 - \log P_{n-1}) + \eta_n, \\ \eta_n &\sim \mathcal{N}(0, \sigma_{OU}^2), \end{aligned} \tag{5}$$

with $\kappa = 0.2$ and $\sigma_{OU} = 0.12$, clamped to a fixed [min, max] range after each step ($[1500, 15000]$ for sale price, base 5000; $[500, 8000]$ for goods cost, base 2000), so that both prices mean-revert around their base value rather than drifting without bound.

3.3 Matching and scoring

Once every table in a negotiation has closed or reached impasse, the set of *closed* tables is passed to Gale-Shapley deferred acceptance [7], with shippers proposing: a shipper's preference over carriers ranks by closing rate ascending (cheapest first, ties broken by fewer turns to close and then a seed-derived pseudo-random order), and a carrier's preference over shippers ranks by closing rate descending (priciest first, same tie-break); impasse tables are excluded from both preference lists entirely, so an architecture that never closes a table at all is never a candidate for a match on that table. The output is a stable matching: no shipper-carrier pair that is not matched to each other would both prefer each other over their assigned (or absent) match. A

matched carrier's *margin* is rate $-\bar{\mu}$ (the realized price minus the route's true reference-price mean); an unmatched carrier scores a margin of 0 for that negotiation. A matched shipper's *profit* is $p_{\text{sale}} - (\text{rate} + c_{\text{goods}})$; if that quantity would be negative the shipper declines the match (modeling a no-ship option) and scores a profit of 0, as does any shipper left unmatched by Gale-Shapley. Every negotiation thus yields exactly one margin observation per carrier and one profit observation per shipper, regardless of how many tables that agent was simultaneously negotiating.

3.4 Nine negotiation architectures

Figure 2 groups the nine architectures described in this section by family.

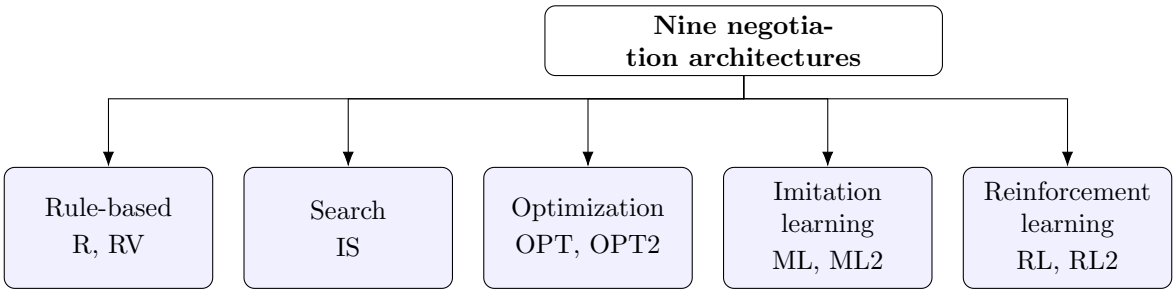


Figure 2: The nine negotiation architectures, grouped by family. Every architecture is built in a mirrored carrier/shipper pair from the same underlying logic, with the anchors of Eqs. (1)-(2) swapped per role.

All nine architectures share one decision interface, `next_move(state) -> Offer | Accept | Reject`, are evaluated frozen (no online parameter updates during the tournament), and are built in mirrored pairs for the shipper and carrier role from the same underlying logic with anchors swapped per Eqs. (1)-(2); the descriptions below are given for the carrier side and apply symmetrically to the shipper side unless noted. Every architecture shares one occupancy heuristic,

$$p_{\text{occ}} = \frac{1}{|\mathcal{P}|} \sum_{j \in \mathcal{P}} \mathbb{1}[\text{Ready}(j)] \,, \tag{6}$$

over the agent's portfolio $\mathcal{P}$ of other live tables in the same negotiation, where $\text{Ready}(j)$ denotes that table j is already closed, or its current counterpart offer already clears the agent's own tactical threshold on that table, and one *negotiation distance* that normalizes any price p between a worst anchor w and a best anchor b,

$$\bar{v}(p; w, b) = \text{clip}_{[0,1]}\left(\frac{p - w}{b - w}\right), \tag{7}$$

with $(w, b) = (RP_t, \text{Tgt}_t)$ on the carrier side and $(w, b) = (RP, \text{Tgt})$ on the shipper side. The negotiation distance measures how close a price leaves the agent to the deal it wants: $\bar{v} = 0$ is the worst deal it may legally accept, $\bar{v} = 1$ its ambitious target. Because it is expressed relative to each table's own band rather than in dollars, it puts routes whose reference prices differ by a factor of almost four (Eq. (3)) and the two roles, whose bands point in opposite directions, on one common 0-1 scale. Eight of the nine architectures (all but R) are built on it, each in a different way: RV and IS use it to value closing now against continuing (Eq. (10)); OPT and OPT2 maximize an expected negotiation distance over candidate prices (Eq. (13)); ML and ML2 receive their input features and predict their offers in it; and RL and RL2 act in it and are rewarded in it (Section 3.5).

3.4.1 Rule-based: R and RV

R commits to no search and no learning. Its acceptance rule compares the counterpart's current offer against a time-varying threshold

$$\begin{aligned} RP_t^{\text{eff}} &= RP_t + f\,(\text{Tgt}_t - RP_t)\,, \\ \theta(\tau) &= RP_t^{\text{eff}} + \left(\text{Tgt}_t - RP_t^{\text{eff}}\right)\left(1 - \frac{\tau}{T}\right)^{\gamma}, \end{aligned} \tag{8}$$

where f is a tactical-floor cushion fraction, T is the number of turns available to that side, and $\gamma > 1$ curves the threshold so it falls slowly at first and quickly as turns run out; the counterpart's offer is accepted once it clears $\theta(\tau)$, or unconditionally on the agent's last turn provided it clears the hard floor RP_t. Absent an acceptable offer, R counters with a fixed opening anchor on the first turn and a geometrically decaying concession step thereafter,

$$\begin{aligned} x_1 &= \text{Tgt}_t - \delta_0\sigma, \\ x_k &= x_{k-1} - \delta_1\sigma\,\rho^{\,k-2} \quad (k \geq 2), \end{aligned} \tag{9}$$

clamped into $[RP_t, \text{Tgt}_t]$ and into non-increasing order relative to its own last offer.

RV reuses R's band, threshold, and concession schedule unchanged, but replaces the time-pressure-only continue/accept comparison with the shared value function (Eq. (10)) whenever the counterpart's current offer already clears the hard floor but not the threshold $\theta(\tau)$, and scales its own concession step by an urgency factor $1 + (1 - p_{\text{occ}}) \cdot 0.5$ derived from Eq. (6), so a carrier whose other tables look unpromising concedes up to 50% faster. The shared value function, evaluated once for a terminal close and once for continuing at the agent's current position, is

$$\begin{aligned} V(\text{terminal}, p) &= \bar{v}(p; RP_t, \text{Tgt}_t), \\ V(\text{continue}) &= \underbrace{\tfrac{1}{2}(\bar{v}(\text{offer on table}) + \bar{v}(\text{my last offer}))}_{h} \\ &\quad \cdot \underbrace{\left(\frac{T-\tau}{T}\right)^{1/2}}_{\text{time pressure}} \cdot \underbrace{[(1 - w_{\text{occ}}) + w_{\text{occ}}\, p_{\text{occ}}]}_{\text{occupancy adjustment}}, \end{aligned} \tag{10}$$

with $w_{\text{occ}} \in [0, 1]$ a per-engine occupancy weight; RV accepts when $V(\text{terminal}) \geq V(\text{continue})$ and counters otherwise.

3.4.2 Search-based: IS

IS adapts the particle-based opponent determinization of information-set Monte Carlo tree search [3] to a single bilateral decision point rather than a branching multi-ply tree, so it never performs the tree search proper and does not grow a search tree at all; a full particle-and-tree budget of roughly 10^3 simulations per decision is impractical at the scale of this tournament. The counterpart's hidden reservation price is determinized by sampling M particles from the only information actually observable – the counterpart's own last offer, a hard lower bound since a rule-based counterpart never offers past its own reservation price –

$$\begin{aligned} \theta^{(1)}, \ldots, \theta^{(M)} &\sim \text{Unif}(\ell,\ \ell + b\,\sigma)\,, \\ \ell &= \text{counterpart's last offer (or } RP_t \text{ if none yet)}, \end{aligned} \tag{11}$$

with b a particle-band width constant; a set of K candidate counter-offers evenly spaced between the agent's current ceiling and RP_t is then scored by expected utility aggregated across every

particle,

$$\begin{aligned}\widehat{P}_{\text{accept}}(c) &= \frac{1}{M}\sum_{m=1}^{M} \mathbb{1}\Big[c \le \theta^{(m)}\Big], \\ \widehat{EU}(c) &= \widehat{P}_{\text{accept}}(c)\cdot V(\text{terminal}, c) + \Big(1-\widehat{P}_{\text{accept}}(c)\Big)\cdot V(\text{continue at } c),\end{aligned} \tag{12}$$

and the candidate $c^\star = \arg\max_c \widehat{EU}(c)$ is offered, or the counterpart's current offer is accepted instead whenever its own terminal value already meets or beats $\widehat{EU}(c^\star)$.

3.4.3 Continuous optimization: OPT and OPT2

OPT treats the next offer as a real-valued variable $x \in [RP_t, \text{ceiling}]$ and climbs an expected-utility surface directly rather than ranking a small candidate set. A belief about the counterpart's current acceptance threshold $\hat{\theta}$ (the counterpart's last offer plus a fixed cushion) feeds a differentiable acceptance-probability model,

$$\begin{aligned}P_{\text{accept}}(x) &= \text{sigmoid}\left(-\beta\,\frac{x-\hat{\theta}}{\sigma}\right), \\ EU(x) &= P_{\text{accept}}(x)\,\bar{v}(x; RP_t, \text{Tgt}_t) + (1-P_{\text{accept}}(x))\,V(\text{continue at } x),\end{aligned} \tag{13}$$

with β a steepness constant. Both terms of EU are negotiation distances (Eq. (7)): the first directly, the second through the continuation value of Eq. (10), which is itself built from the distances of the offer on the table and of the agent's own last offer. $EU(x)$ is therefore the negotiation distance the agent expects to end at if it offers x, and the optimum

$$x^\star = \arg\max_{x\in[RP_t,\,\text{ceiling}]} EU(x) \tag{14}$$

is located by bisecting on the *sign* of a central-difference gradient at the window's midpoint m,

$$\begin{aligned}g(m) &= \frac{EU(m+h)-EU(m-h)}{2h}, \\ h &= \max\big(0.01\,(\text{hi}-\text{lo}),\,10^{-9}\big),\end{aligned} \tag{15}$$

narrowing the window to $[m, \text{hi}]$ if $g(m) > 0$ and to $[\text{lo}, m]$ otherwise, for a fixed number of iterations, rather than stepping by the gradient's magnitude, since EU lives on the 0-1 negotiation-distance scale, unrelated to the dollar scale being searched. $x^\star$ is thus the price whose expected negotiation distance is highest, and OPT's accept rule compares like with like: it accepts the counterpart's current offer whenever that offer's own negotiation distance already meets or beats EU at the offer OPT would otherwise make. A final offer is then emitted a small, reproducibly-seeded distance below $x^\star$ in the concession direction only, so that a deterministic optimizer is not trivially fingerprintable by a counterpart observing many negotiations.

OPT2 keeps OPT's optimizer and acceptance model but replaces its single-point belief with a line $\text{offer}_k \sim a + b\,k$ fitted to the counterpart's entire observed offer sequence by explicit gradient descent on squared error over the (a, b) parameters, extrapolated one step beyond the last observed offer and then cushioned exactly as OPT's point estimate is; it also replaces OPT's fixed-magnitude jitter with a performance-neutral randomization envelope, widening a window downward from $x^\star$ for as long as EU stays within a fixed fractional tolerance ϵ_{tol} of $EU(x^\star)$,

$$\begin{aligned}a &= \inf\{x \le x^\star : EU(x^\star) - EU(x) \le \epsilon_{\text{tol}}\,|EU(x^\star)|\}, \\ \text{window} &= [\,a,\ x^\star\,],\end{aligned} \tag{16}$$

and emitting an offer sampled uniformly from that window with a seed derived from the negotiation, so that the resulting randomization never costs expected utility by more than ϵ_{tol} while still varying the realized offer.

3.4.4 Imitation learning: ML and ML2

ML works entirely in negotiation distance (Eq. (7)): every price-like input feature (offers on the table, the agent's own last offer) is expressed as a distance, and the model predicts, from the current table state, the negotiation distance $p \in [0, 1]$ at which to make its next offer, via a gradient-boosted regressor trained on OPT2's self-play offers [14] (Section 3.5 gives training details), maps it to a price $x = RP_t + p\,(\mathrm{Tgt}_t - RP_t)$, and clamps x into the same legal band and monotonicity constraint every other engine respects; the accept/continue decision is deliberately not left to the model and instead reuses the shared value-function comparison of Eq. (10), compared against the agent's own already-stated last offer rather than its freshly predicted x, since comparing against a hypothetical, not-yet-offered x was found to trigger accepting far too early.

ML2 extends ML with a second, independent classifier that also predicts whether to accept the counterpart's current offer, $\widehat{P}_{\mathrm{accept}}(\mathrm{state})$, trained separately from the offer regressor so that neither model's output is fed into the other as an input; the offer regressor and the hard last-turn floor behave exactly as in ML, and ML2 accepts an otherwise-legal counterpart offer whenever $\widehat{P}_{\mathrm{accept}}(\mathrm{state}) \geq 0.5$.

3.4.5 Reinforcement learning: RL and RL2

RL is a policy trained with proximal policy optimization [16, 18] against a single-agent negotiation environment in which the policy's reward depended only on closing a legal deal in that isolated environment, with no rival carriers present during training. At inference the policy outputs a single aspiration $p \in [0, 1]$, a negotiation distance (Eq. (7)) mapped to a price exactly as in ML/ML2, and the agent accepts the counterpart's current offer outright whenever that offer already clears both the hard floor and the policy's own aspiration, rather than deferring to a hand-derived value-function comparison.

RL2 is trained with the same action mechanism and the same algorithm, but its training reward depended on actually winning the Gale-Shapley matching stage (Section 3.3) against rival carriers, not merely on closing a deal in isolation. Tracing a deployed round, however, surfaced a confirmed, regression-tested bug in RL2's inference path: its training environment always held the policy's own portfolio-occupancy feature fixed at a neutral default, since the learning agent's own table never had real siblings during training, while a deployed RL2 genuinely negotiates several tables at once and would otherwise present that feature at whatever its real, non-neutral portfolio state is; RL2's inference code pins the feature back to its neutral training value before every decision, restoring the input distribution the policy was actually trained on rather than feeding it an input it never saw in training. We deliberately evaluate RL2 as actually deployed, with this feature pinned, rather than silently repairing the training/deployment mismatch itself, since the gap between what an architecture could in principle do and what the tournament actually measures is itself one of this paper's results (Section 4.4).

3.5 Architecture parameters and training

Table 1 lists each architecture's governing hyperparameters, held fixed across every market condition and role (Section 5 returns to the consequences of not tuning them further). **ML** and **ML2**'s offer regressors are gradient-boosted regression trees trained to imitate **OPT2** specifically – not a from-scratch policy or a hand-authored oracle – on several hundred self-play negotiations of OPT2 against every other hand-designed architecture, evaluated by held-out mean absolute error against OPT2's own band position; **ML2**'s accept classifier is a second, independently trained gradient-boosted classifier on the same feature vector, fit only on turns where accepting was legal, and never fed the offer regressor's output or vice versa. **RL** and **RL2** are both trained with proximal policy optimization [16, 18] (a two-layer, 64-unit multilayer-

perceptron policy, discount factor $\gamma = 0.99$, for 300,000 environment steps), differing only in the environment and reward that training optimizes against: RL's terminal reward is the negotiation distance of the closing price, $\bar{v}(\text{closing price}; RP_t, \text{Tgt}_t)$ (Eq. (7)), whenever its one training opponent for that episode (resampled each episode from {R, RV, IS, OPT, OPT2}) accepts a legal deal in an isolated, single-table environment, and zero otherwise; RL2 is trained inside a small competitive sub-round of one to two same-side rivals (sampled from the same five-architecture pool) against one shared counterpart, cleared by the same Gale-Shapley mechanism as the full tournament (Section 3.3), with reward -0.5 on impasse, -0.15 if its own table closes but the match assigns the counterpart to a rival instead, and $\bar{v}(\text{closing price}; RP_t, \text{Tgt}_t)$ conditional on winning the match.

Table 1: Governing hyperparameters for all nine architectures, held fixed across every market condition and role.

Architecture	Key parameters
R	Opening offset 0.15σ (carrier) / band width 0.5σ (shipper); concession decay $\rho = 0.5$; threshold curvature $\gamma = 2.0$
RV	As R, plus the occupancy-aware urgency factor derived from Eq. (6) (up to 50% faster concession when the agent's own match prospects look poor)
IS	24 belief particles; 7 candidate offers evaluated per turn; particle-sampling band 1.5σ wide; deterministic per-decision seeding (no run-to-run variance at a fixed seed)
OPT	Continuous bisection search over expected utility (24 steps); acceptance-probability sigmoid slope 4.0; belief cushion 0.5σ
OPT2	Extends OPT with an online gradient-fitted linear opponent-offer-trend model (60 gradient steps, learning rate 0.15) and an anti-fingerprinting envelope search (200-iteration cap, 5% expected-utility tolerance)
ML	Gradient-boosted regression tree (max depth 4, 200 boosting iterations, learning rate 0.08), trained to imitate OPT2's self-play offers
ML2	Same offer regressor as ML, plus an independently trained gradient-boosted accept classifier (same hyperparameters), decision threshold 0.5
RL	PPO, 2×64 MLP policy, $\gamma = 0.99$, 300,000 training steps, one isolated opponent per episode resampled from {R, RV, IS, OPT, OPT2}
RL2	Same PPO configuration as RL, trained inside a Gale-Shapley-cleared competitive sub-round against one to two same-side rivals from the same five-architecture pool

3.6 Experimental design

Figure 3 illustrates the exclusion-pair rotation scheme used to build each scarce condition.

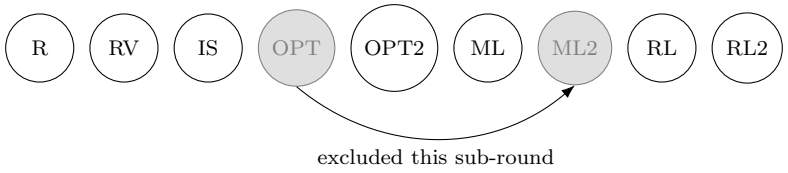


Figure 3: One of the $\binom{9}{2} = 36$ exclusion pairs used to build a scarce condition: the nine architectures on the scarce side, with two (shaded; OPT and ML2 shown here purely for illustration) withheld for that sub-round's $N_{\text{pair}} = 12$ negotiations while the other seven compete and all nine architectures remain present on the abundant side. The excluded pair cycles through all 36 combinations in turn, so every architecture is withheld from exactly 8 of the 36 sub-rounds and present in the remaining 28, per Eq. (17).

Every architecture appears at most once per side per negotiation, so that no result reflects one architecture competing against a copy of itself rather than against the other eight. The

balanced condition runs $N = 400$ negotiations with all nine architectures present on both sides. Each *scarce* condition reduces one side's roster from nine architectures to seven, holding the other side at all nine; which two architectures are excluded is rotated systematically through every one of the $\binom{9}{2} = 36$ possible exclusion pairs, running $N_{\text{pair}} = 12$ negotiations per excluded pair, for $36 \times 12 = 432$ negotiations total per scarce condition. Because each architecture is excluded only in the 8 pairs that name it (one pairing with each of the other eight architectures), it is present in $36 - 8 = 28$ of the 36 sub-rounds, contributing an effective sample size on the scarce side of

$$N_{\text{eff,scarce}} = N_{\text{pair}} \times 28 = 336, \tag{17}$$

while the abundant side of the same scarce condition, present in every one of the 36 sub-rounds, contributes the full $N_{\text{pair}} \times 36 = 432$ observations per architecture; both figures are counted directly from the resulting data rather than assumed, and matched the values above exactly. We refer to the condition in which carriers are reduced to seven architectures as `carrier_scarce` (shippers compete for scarce carrier capacity) and the mirror condition as `shipper_scarce` (carriers compete for scarce shipper demand).

3.7 Computational setup

All experiments – the nine-architecture tournament, the training of ML/ML2 and RL/RL2, and the language-model comparison of Section 4.6 – ran on a single workstation: an Intel Core i9-14900F CPU (24 cores, 32 threads), 128 GB of RAM, and one NVIDIA GeForce RTX 4090 GPU with 24 GB of memory, under Windows 11 Pro. RL2 was trained on the CPU; RL used the training library's default device selection, which picks the GPU when one is available [16]. Language models were served locally through Ollama with each model tag's default quantization and no other request options; the exact Ollama server version in use during data collection was not logged (the installed client is version 0.34.2). This matters for the language-model results: at that quantization each of the three 70B-class models (`llama3.3:70b`, `qwen2.5:72b`, `deepseek-r1:70b`) needs more memory than the GPU's 24 GB, so part of each model ran from CPU memory, which slows inference and makes the latencies reported in Section 4.6 specific to this hardware.

3.8 Statistical analysis

Every architecture's outcome under a given market condition and role is summarized by a percentile bootstrap confidence interval rather than a normal-approximation interval, since margin and profit outcomes are not symmetrically distributed: given n observations $y_1, \ldots, y_n$, we resample $B = 5000$ bootstrap samples of size n with replacement, compute the sample mean $\bar{y}^{(b)}$ of each resample, let q_α denote the α-quantile of the resulting set $\{\bar{y}^{(b)}\}_{b=1}^{B}$, and report

$$\widehat{\text{CI}}_{95\%} = [\, q_{0.025},\ q_{0.975} \,]. \tag{18}$$

For every pair of architectures present in the same market condition and role, we restrict to the subset of negotiation-days on which *both* architectures were actually present (every day, in the balanced condition; the subset of sub-rounds excluding neither architecture, in a scarce condition), form the paired per-day difference $d_i = y_i^{(A)} - y_i^{(B)}$, and test $H_0 : \text{median}(d) = 0$ with a two-sided Wilcoxon signed-rank test, a non-parametric choice appropriate for the same skewed outcome distributions the bootstrap CI accounts for. Because every market condition and role admits $\binom{9}{2} = 36$ such pairwise comparisons answering the same family of questions ("who beats whom under this market condition"), we correct the 36 raw p-values within each (market condition, role) family separately with the Holm-Bonferroni step-down procedure at $\alpha = 0.05$: sorting the family's p-values $p_{(1)} \le \cdots \le p_{(36)}$, a hypothesis at rank i is rejected only if $p_{(j)} \le \alpha/(36 - j + 1)$ for every $j \le i$, which controls the family-wise error rate without assuming independence between the 36 tests.

4 Results

4.1 Descriptive ranking by market condition

Table 2 and Table 3 report each architecture's match rate and bootstrap mean outcome (margin for carriers, profit for shippers) in every market condition, both in USD and as a percentage that puts the dollar figures in scale: ML2's balanced-condition mean margin of 98.5 USD, for example, is 6.6% of the matched route's reference price, and a shipper's typical profit of roughly 1,100 USD is about 20% of its sale price; Figure 4 plots the same rankings with their confidence intervals. In the balanced carrier market, the top of the ranking mixes three different families rather than one: by point estimate the imitation-learned ML2 is followed by the reinforcement-learned RL, then the continuous-optimization OPT2 and OPT, but none of these four differs significantly from any other after correction (Section 4.2), so they form a statistically tied top group rather than a strict order; the search-based IS sits in the upper-middle, and the two rule-based architectures R and RV occupy the bottom two positions, both with a mean margin confidence interval that excludes zero on the losing side, R by the widest margin of any architecture. In the balanced shipper market the ordering is far more compressed among the top eight architectures (RL through ML2, all within roughly 85 profit units of each other) with one striking exception: RL2 is isolated at the bottom of the shipper ranking by a margin of roughly 250-340 profit units in the balanced and `shipper_scarce` conditions, a gap large enough to be visible in Table 3 without any statistical test; under `carrier_scarce`, however, RL2 moves into the middle of the ranking (849.8, fifth of nine) and it is instead RL that falls to the bottom (779.2), a reordering discussed further in Section 4.5.

Table 2: Carrier-side match rate and bootstrap mean margin in USD (95% CI) by architecture and market condition. N is the effective sample size (Eq. (17) for the scarce side of a scarce condition). "% of ref." is the mean, over negotiation-days, of the margin as a percentage of the matched route's reference price $\bar{\mu}$ (an unmatched day counts as 0%); because expensive routes weigh more in dollars than in percent, the two columns can rank close architectures differently (e.g. ML2 and RL in the balanced condition). 77 of the tournament's 8,256 matched carrier-days (0.9%) could not be paired with their route from the stored logs and are left out of this column only.

	balanced ($N = 400$)				carrier_scarce ($N = 336$)				shipper_scarce ($N = 432$)			
Arch.	match	mean margin (USD)	% of ref.	95% CI	match	mean margin (USD)	% of ref.	95% CI	match	mean margin (USD)	% of ref.	95% CI
ML2	0.79	98.5	6.6	[82.2, 115.6]	0.84	111.9	8.0	[91.2, 135.1]	0.62	65.1	4.5	[52.9, 78.7]
RL	0.62	90.4	6.9	[74.6, 107.5]	0.84	142.0	10.6	[123.3, 162.0]	**0.16**	40.8	3.2	[27.8, 55.2]
OPT2	0.80	83.5	5.9	[67.7, 101.7]	0.91	113.8	7.3	[95.4, 133.4]	0.69	64.4	4.4	[53.0, 77.3]
OPT	0.79	79.8	5.3	[66.5, 94.3]	0.87	78.7	4.9	[66.8, 92.1]	0.76	75.4	5.1	[62.5, 90.1]
IS	0.69	71.4	5.0	[57.6, 86.5]	0.82	115.2	8.1	[95.0, 136.9]	**0.23**	30.6	2.6	[19.8, 42.4]
ML	0.89	46.4	3.2	[33.4, 61.9]	0.94	61.4	3.9	[45.2, 79.7]	0.86	59.9	4.4	[44.7, 76.5]
RL2	0.92	28.7	1.6	[20.6, 38.2]	0.93	35.5	2.2	[23.7, 48.9]	0.92	36.4	2.1	[25.7, 48.5]
RV	0.92	−21.8	0.0	[−38.5, −2.6]	0.93	0.2	1.3	[−20.4, 23.2]	0.89	−40.4	−1.3	[−51.4, −28.8]
R	0.93	−45.8	−1.6	[−59.0, −31.0]	0.93	−18.5	0.2	[−36.0, 0.8]	0.94	−62.1	−2.8	[−70.2, −53.0]

Table 3: Shipper-side match rate and bootstrap mean profit in USD (95% CI) by architecture and market condition. "% of sale" is the mean, over negotiation-days, of the profit as a percentage of that day's sale price p_{sale} (an unmatched or declined day counts as 0%).

	balanced ($N = 400$)				carrier_scarce ($N = 432$)				shipper_scarce ($N = 336$)			
Arch.	match	mean profit (USD)	% of sale	95% CI	match	mean profit (USD)	% of sale	95% CI	match	mean profit (USD)	% of sale	95% CI
RL	0.82	1124.8	20.8	[1032.4, 1220.5]	0.63	779.2	14.2	[699.9, 863.6]	0.85	1233.7	21.7	[1115.5, 1352.1]
R	0.82	1094.4	20.1	[1002.5, 1193.9]	0.66	800.5	14.8	[716.2, 883.9]	0.89	1212.9	21.9	[1111.6, 1315.5]
OPT	0.84	1082.5	20.0	[990.3, 1177.8]	0.69	867.1	15.9	[781.3, 951.3]	0.87	1143.5	20.0	[1033.2, 1250.3]
IS	0.79	1068.9	19.6	[976.0, 1165.3]	0.70	833.1	15.3	[752.5, 918.0]	0.85	1149.7	20.8	[1044.5, 1253.1]
RV	0.80	1066.9	19.5	[973.1, 1167.3]	0.66	795.4	14.6	[713.8, 879.7]	0.86	1122.9	19.9	[1010.5, 1237.6]
ML	0.83	1063.9	19.6	[969.6, 1163.2]	0.71	891.6	16.1	[800.7, 982.9]	0.87	1109.8	20.1	[1008.0, 1215.0]
OPT2	0.82	1055.7	19.3	[963.4, 1149.8]	0.69	885.0	16.2	[801.3, 971.7]	0.87	1139.6	21.0	[1042.4, 1236.2]
ML2	0.83	1038.8	19.1	[950.2, 1135.2]	0.69	852.0	15.6	[766.7, 938.1]	0.88	1240.8	22.0	[1128.4, 1354.3]
RL2	0.81	**788.2**	14.5	[722.8, 857.0]	0.81	**849.8**	15.6	[787.8, 914.6]	0.84	**776.8**	14.3	[706.7, 848.2]



Figure 4: Bootstrap mean outcome (margin for carriers, profit for shippers) by architecture, for all nine architectures under all three market conditions. Error bars are 95% percentile bootstrap confidence intervals.

4.2 Pairwise significance after correction

After Holm-Bonferroni correction within each (market condition, role) family of 36 comparisons, 27 of 36 carrier comparisons were significant in the balanced condition, 28 of 36 in `carrier_scarce`, and 30 of 36 in `shipper_scarce`; on the shipper side, 8 of 36 comparisons were significant in the balanced condition, *zero* of 36 in `carrier_scarce`, and 8 of 36 in `shipper_scarce`. The complete absence of significant shipper-side differences under `carrier_scarce` is itself a finding: once shippers must compete for a reduced pool of carriers, no shipper architecture's negotiating strategy produces a statistically distinguishable advantage in realized profit over any other, even though the same nine architectures are clearly separated from each other in the balanced and `shipper_scarce` conditions. Figure 5 shows the balanced-condition shipper win matrix: RL2 loses every one of its eight significant comparisons and wins none, the only architecture with a completely one-sided record in the balanced condition in either role.

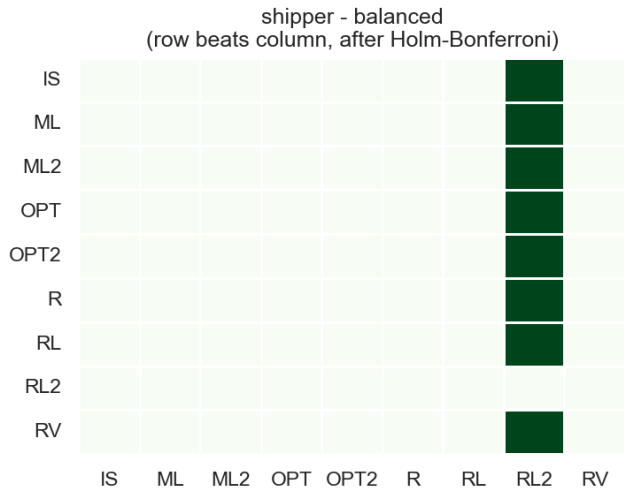
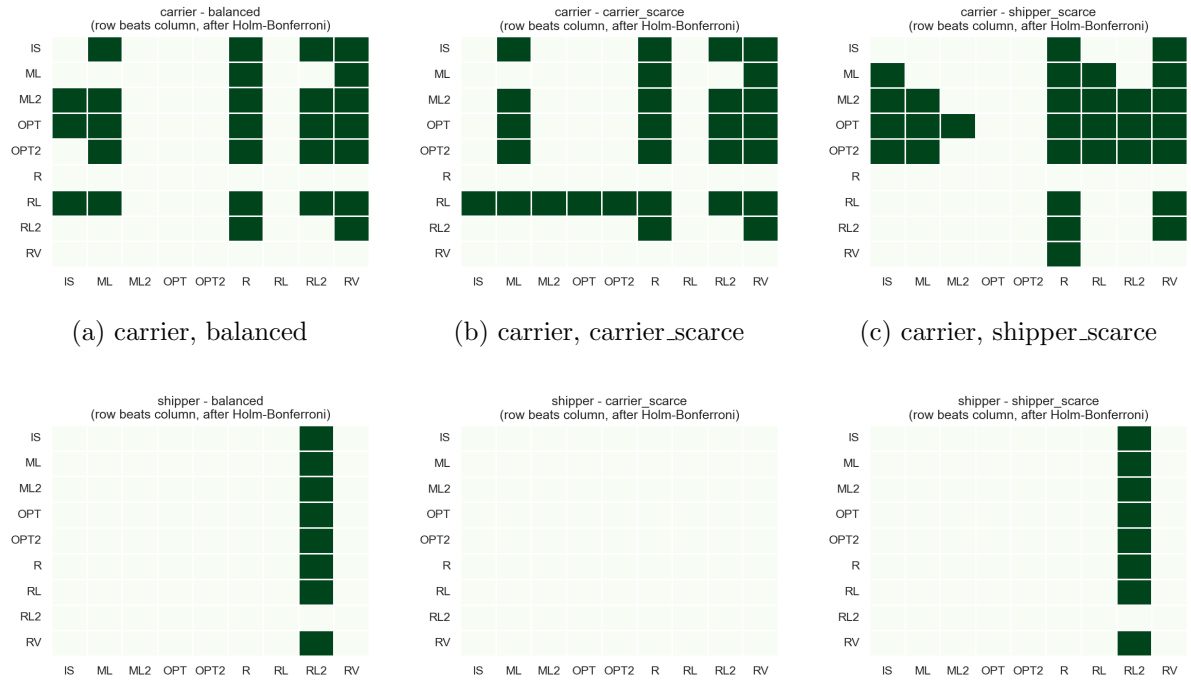


Figure 5: Shipper-side win matrix, balanced market condition: a filled cell in row i, column j means architecture i significantly outperforms architecture j after Holm-Bonferroni correction. RL2 wins zero of its eight comparisons.

Figure 6 reproduces all six win matrices (both roles, all three market conditions) for completeness; panel (f) (shipper, `shipper_scarce`) and panel (c) (carrier, `shipper_scarce`) are the two panels most relevant to the match-rate collapse of Section 4.3, since RL and IS's matching failures there mean the two architectures contribute very few observations to panel (c)'s comparisons even where they remain nominally present.



(a) carrier, balanced (b) carrier, carrier_scarce (c) carrier, shipper_scarce

(d) shipper, balanced (e) shipper, carrier_scarce (f) shipper, shipper_scarce

Figure 6: All six pairwise win matrices (row significantly beats column, after Holm-Bonferroni correction within each panel's 36 comparisons): carrier role (top row) and shipper role (bottom row), across the three market conditions. Panel (d) is reproduced larger as Figure 5 above.

4.3 Competitive training and match-rate robustness under scarcity

Table 2 shows that two carrier architectures, RL and IS, lose the large majority of their matching opportunities specifically under `shipper_scarce`, the one condition in which carriers must genuinely out-compete each other for a reduced pool of shipper demand: RL's match rate falls from 0.62 (balanced) and 0.84 (`carrier_scarce`, where carriers are *not* competing with each other for demand) to **0.16** under `shipper_scarce`, and IS falls from 0.69 and 0.82 to **0.23** under the same condition. No other carrier architecture shows a comparable drop: RL2's match rate instead stays within a narrow band (0.92, 0.93, 0.92) across all three conditions. R (0.93, 0.93, 0.94) and RV (0.92, 0.93, 0.89) are comparably stable, however, despite neither being trained with any competitive pressure at all; match-rate stability on its own therefore does not establish that RL2's stability is caused by its training against rival carriers, only that RL2 avoids the specific failure mode that collapses RL and IS. RL was trained with proximal policy optimization against an environment whose reward depended only on closing a deal in an isolated, non-competitive setting (Section 3.4); IS's opponent model is a one-shot particle determinization of a single counterpart's reservation price, with no representation of rival carriers at all. Neither architecture's training or search procedure ever represented the possibility of losing a deal to a rival carrier after agreeing a price with a shipper, and Table 2 shows the direct cost of that gap exactly in the one market condition designed to expose it.

4.4 The RL2 train/deployment gap

RL2 avoids the match-rate collapse of Section 4.3, but Table 2 and Table 3 show it ranks at or near the bottom of realized outcome in the balanced and `shipper_scarce` conditions, despite that stable match rate; under `carrier_scarce` it instead rises to the middle of the shipper ranking, a reordering Section 4.5 discusses in full. On the shipper side, every one of RL2's eight pairwise comparisons in the balanced condition is significant after correction, all favoring the rival architecture, with $p_{\text{holm}} < 10^{-12}$ in every case (Section 4.2, Figure 5); on the carrier side, seven of RL2's eight balanced-condition comparisons are significant and all favor the rival, the sole exception (RL2 vs. ML, mean difference 17.7, $p_{\text{holm}} = 1.0$) not reaching significance after correction. This outcome is consistent with the confirmed feature mismatch described in Section 3.4: because RL2's training environment always held its own portfolio-occupancy feature at a fixed neutral value, and the pinned inference-time value reproduces that same neutral value rather than differing from it, the policy never had an opportunity to learn how to moderate its aspiration in response to genuine competitive pressure; a trace of a deployed round confirmed the shipper-side policy opening at roughly 94% of its own worst-case ceiling on the very first turn and being accepted instantly by whichever counterpart it faced, the behavior of a policy whose aspiration was shaped entirely by a training signal that never varied with real portfolio risk. RL2 therefore demonstrates that training a policy against genuine competitive pressure can solve the specific failure mode of Section 4.3 (match-rate collapse under scarcity) without producing a policy that is competitive on realized payoff, if a feature the policy depends on is not presented identically at training and at deployment time.

4.5 Rank stability of margin and profit across market conditions

Every ranking in this subsection orders architectures by realized economic outcome – mean margin for carriers, mean profit for shippers, in USD (Tables 2-3) – not by match rate. The carrier-side top four under the balanced condition (ML2, RL, OPT2, OPT) remain in the top five under both scarce conditions, though their relative order shifts: RL rises to first under `carrier_scarce` and IS rises into the top four under both scarce conditions, while OPT and ML2 fall in relative rank specifically under `shipper_scarce` even though their mean margin changes comparatively little. On the shipper side, the top eight architectures reorder modestly

across conditions (R and RL exchange the top position; ML2 rises from eighth under balanced to first under `shipper_scarce`), and RL2 is not uniformly the bottom performer: it is last in the balanced and `shipper_scarce` conditions by the same large margin documented in Section 4.2, but under `carrier_scarce` it rises to fifth of nine (849.8) while RL itself drops to the bottom (779.2) – RL's own worst result of the three conditions, and the only condition in which RL2 is not the shipper-side floor.

4.6 Comparison against LLM-based negotiation agents

As a supplementary extension of the same tournament methodology, we additionally embedded large-language-model-based negotiation agents (served locally via Ollama, prompted with the agent's exact hard-band numbers and its own table's full turn history, using the same legality and monotonicity enforcement as every architecture in Section 3.4) into a small market alongside three fixed same-side rivals spanning this tournament's weakest, strongest hand-designed, and strongest learned carrier architectures (R, OPT2, and ML2), against a fixed four-architecture counterpart roster (R, RV, OPT2, IS), across nine language models spanning three parameter-count tiers (tiny, small, and medium: under 3B, 4-14B, and 15-70B parameters respectively) and three model families (Llama, Qwen, and a DeepSeek-R1 distillation) at each tier, for 40 independent negotiation-days per (model, role) combination, with each call bound by a 90-second request timeout and a `max_retries=2` budget before that day forfeited. Two methodological notes before the results, reported transparently rather than silently absorbed. First, an unattended multi-day run that must survive repeated crashes and restarts – realized by resuming from whichever days a prior process already wrote – turned up a seeding bug this comparison alone exposed: the per-(model,role) combination seed was originally derived from Python's built-in `hash()` on a string tuple, which is randomized per process by design and therefore not reproducible across the many restarts this study required; we traced it after finding genuinely divergent outcomes recorded under the same day index in two (model, role) files, fixed it with a process-independent hash, and deduplicated the affected files down to one valid realization per day before any statistic below was computed. Same-day comparisons between a language model and its three fixed rivals are unaffected by this bug, since every agent at one table shares one draw of the market regardless of which seed produced it; what is not guaranteed is that the exact day-by-day sequence would reproduce bit-for-bit from the base seed alone before this fix. Second, partway through data collection we introduced an explicit `think=false` request to Ollama for the DeepSeek-R1 family specifically, after observing call latencies far more variable than any other family's; this had no effect on Llama or Qwen (confirmed directly: identical response behavior with and without the flag) and was applied starting from that family's shipper-role runs onward, so DeepSeek-R1's carrier-role data reflects the model's default reasoning behavior while its shipper-role data reflects the explicit request to suppress it. Comparing the two directly (Table 6) shows this request did *not* reliably reduce latency at any DeepSeek-R1 scale – mean latency is flat-to-slightly-worse between carrier and shipper for all three DeepSeek-R1 sizes – so we report both halves of the dataset honestly rather than claim a fix that the data does not support, and the headline finding below (Tables 4-5) holds identically in the unaffected carrier-role data alone.

Table 4: Carrier-side LLM results: match rate and bootstrap mean margin (95% CI), by tier and family, against the three fixed same-side rivals in the same markets.

Tier	Model	match	mean margin	95% CI
fixed rivals (pooled across all nine LLM markets)				
	ML2	1.00	68.6	[66.5, 70.8]
	OPT2	1.00	63.9	[61.5, 66.3]
	R	1.00	−60.9	[−68.8, −53.2]
tiny	llama3.2:1b	0.38	−18.1	[−33.2, −3.8]
tiny	qwen2.5:1.5b	0.80	−63.6	[−88.1, −40.7]
tiny	deepseek-r1:1.5b	0.90	−64.6	[−89.7, −39.0]
small	llama3.1:8b	1.00	−113.0	[−134.1, −90.7]
small	qwen2.5:7b	1.00	−136.0	[−157.4, −112.5]
small	deepseek-r1:7b	0.97	−87.6	[−114.3, −60.0]
medium	llama3.3:70b	0.82	**+25.4**	[+5.6, +43.2]
medium	qwen2.5:72b	1.00	−20.9	[−44.5, +0.9]
medium	deepseek-r1:70b	**0.03**	−0.4	[−1.3, 0.0]

Table 5: Shipper-side LLM results: match rate and bootstrap mean profit (95% CI).

Tier	Model	match	mean profit	95% CI
fixed rivals (pooled across all nine LLM markets)				
	R	1.00	1163.1	[1101.6, 1225.4]
	OPT2	1.00	1148.8	[1082.2, 1210.8]
	ML2	0.99	1117.7	[1055.4, 1180.2]
tiny	llama3.2:1b	0.50	2.7	[0.0, 7.8]
tiny	qwen2.5:1.5b	0.97	462.5	[294.4, 652.3]
tiny	deepseek-r1:1.5b	0.95	379.8	[224.6, 559.0]
small	llama3.1:8b	1.00	548.6	[390.0, 716.0]
small	qwen2.5:7b	0.60	406.9	[228.9, 610.6]
small	deepseek-r1:7b	1.00	584.9	[418.3, 756.8]
medium	llama3.3:70b	0.97	**1264.1**	[1073.5, 1454.0]
medium	qwen2.5:72b	1.00	908.8	[741.9, 1082.6]
medium	deepseek-r1:70b	**0.03**	3.8	[0.0, 11.3]

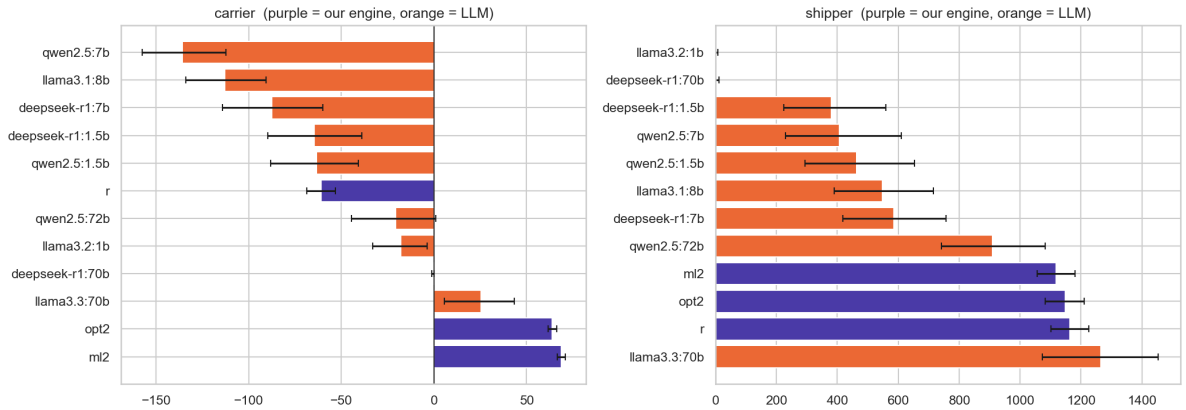


Figure 7: Each language model's bootstrap mean outcome against its three fixed same-side rivals (purple), by role. Orange bars are the nine language models.

Across all $9 \times 2 \times 3 = 54$ pairwise comparisons between a language model and one of its three

fixed rivals, 42 are significant losses for the language model, 4 are significant wins, and 8 are inconclusive after Holm-Bonferroni correction within each (model, role) family of three. The pattern is sharply asymmetric by rival: against OPT2 and ML2 combined, language models lose 32 of 36 comparisons significantly and never win a single one (the remaining 4 are inconclusive, not reversed); against R, the weakest hand-designed architecture, the record is far more even – 10 significant losses, 4 significant wins, 4 inconclusive. Every one of the four significant wins against R is on the carrier side (`llama3.2:1b`, `llama3.3:70b`, `qwen2.5:72b`, and `deepseek-r1:70b`); on the shipper side, every language model loses to R significantly except `llama3.3:70b`, whose comparison is inconclusive with a positive point estimate (+139.6, $p_{\text{holm}} = 0.80$). Two of those four wins, however, come from models with very low carrier-side match rates (`llama3.2:1b` at 38%, `deepseek-r1:70b` at just 3%, Table 4): since an unmatched day scores exactly 0 (Section 3.3) against R's own negative mean margin (−60.9, Table 4), most of each model's advantage reflects how often it scored a forfeit's 0 rather than consistently out-negotiating R; only `qwen2.5:72b` (100% match) and `llama3.3:70b` (82% match) win against R by actually closing deals at better terms. `llama3.3:70b` is this comparison's standout performer by a wide margin: it is the only language model to beat the weakest rival on both roles (significantly on carrier, inconclusively on shipper), the only one whose comparisons against OPT2 and ML2 are not significant losses on the shipper side (point estimates of +161.7 and +111.0 from the underlying paired tests of Section 3.8, not shown in a table), and it posts the single highest mean profit of all twelve shipper-side participants in this comparison, language models and fixed rivals alike (1264.1, ahead of R's 1163.1, Table 5).

`deepseek-r1:70b` shows the opposite pattern: a match rate of **2.5% on both roles**, against 38-100% for every other model in this comparison including its own 1.5B and 7B siblings (Tables 4-5). Table 6 traces this to response latency: `deepseek-r1:70b`'s calls average 91.6-92.0s against a 90s timeout, essentially pinned at the ceiling on both roles regardless of the `think=false` request, with a first-try structured-output success rate of only 25%/26% (carrier/shipper), far below its own 1.5B (49-52%) and 7B (55-88%) siblings and at or below every Llama or Qwen model at any tier (26-98%). Unlike the engine tournament's match-rate collapse (Section 4.3), which traced to a training-time gap (no competitive pressure during training), this one traces to an inference-time constraint: `deepseek-r1:70b` is slower at this task than either its own smaller siblings or the same-tier Llama/Qwen models, and the explicit `think=false` request did not change that latency profile. This constraint is specific to our hardware as much as to the model: on the single 24 GB GPU of Section 3.7, all three 70B-class models partly run from CPU memory, and only `deepseek-r1:70b` was slow enough there to hit the 90-second timeout on nearly every call. A machine able to hold the whole model in GPU memory, or a longer timeout, could change this result, so we report it as a deployment constraint of this setup, not as a property of the model family.

Table 6: Structured-output compliance and latency, by tier, family, and role (DeepSeek-R1's `think` setting differs by role; see the methodological note above).

Tier	Model (role)	first-try-ok	mean latency (s)	p95 latency (s)
tiny	llama3.2:1b (carrier / shipper)	0.32 / 0.26	2.3 / 2.3	2.5 / 2.4
tiny	qwen2.5:1.5b (carrier / shipper)	0.45 / 0.58	2.3 / 2.3	2.4 / 2.4
tiny	deepseek-r1:1.5b (carrier / shipper)	0.49 / 0.52	17.1 / 21.7	92.1 / 92.1
small	llama3.1:8b (carrier / shipper)	0.88 / 0.88	2.6 / 2.5	2.7 / 2.6
small	qwen2.5:7b (carrier / shipper)	0.55 / 0.81	2.4 / 2.4	2.5 / 2.6
small	deepseek-r1:7b (carrier / shipper)	0.55 / 0.88	9.0 / 10.6	15.0 / 22.9
medium	llama3.3:70b (carrier / shipper)	0.68 / 0.98	29.3 / 30.2	37.8 / 38.1
medium	qwen2.5:72b (carrier / shipper)	0.78 / 0.93	41.6 / 31.0	66.6 / 38.0
medium	**deepseek-r1:70b** (carrier / shipper)	**0.25 / 0.26**	**91.6 / 92.0**	**92.1 / 92.1**

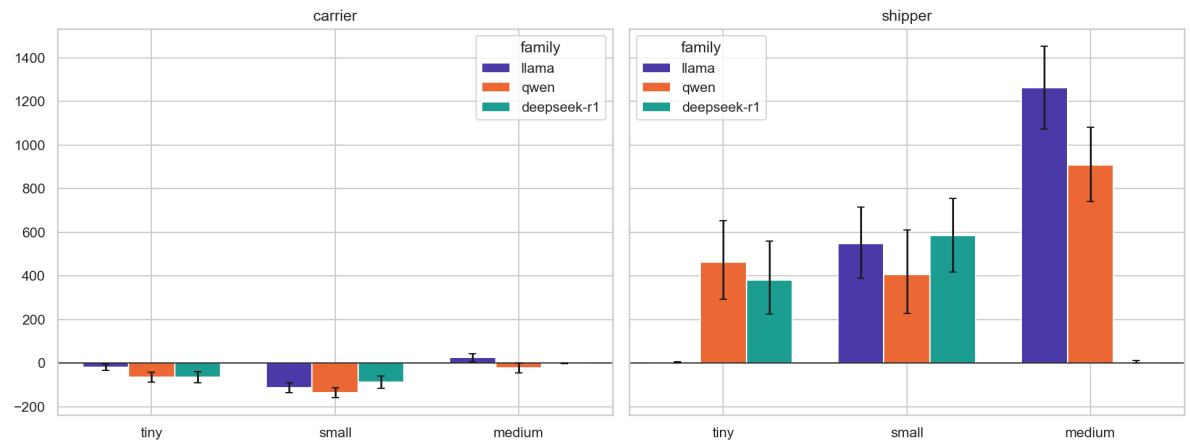


Figure 8: Bootstrap mean outcome by tier and family, language models only. Error bars are 95% bootstrap CIs.

Pooling across family, scale does not help uniformly: on the carrier side the tiny tier (−48.7) actually beats the small tier (−112.6), with the medium tier best of all (+1.4) – a non-monotonic pattern driven by the same forfeit-inflation effect that follows mechanically from the scoring convention established in Section 3.3, since a model that fails to produce a legal move scores exactly 0 for that day rather than a large negative margin, and the tiny tier's lower match rates (Tables 4) mechanically pull its mean toward zero. On the shipper side, by contrast, the pattern is monotonic – tiny (281.7) < small (511.7) < medium (721.0) – but this is not a clean scaling effect either: it is driven almost entirely by `llama3.3:70b`'s standout result averaged against `deepseek-r1:70b`'s near-total collapse, two same-tier models roughly 1260 profit units apart (Table 5). We report both the pooled tier summary and this per-family decomposition explicitly (Figure 8; full figures in the released `tier_summary.csv` and `tier_family_summary.csv`) rather than the tier average alone, since the average on its own would misrepresent both the carrier-side forfeit artifact and the shipper-side family divergence as if they were uniform scaling effects. We report this comparison as a direct extension of the same experimental and statistical protocol used for the nine architectures above, so that language-model-based agents can be read on the same ranking and the same corrected-significance terms as every hand-designed, search-based, imitation-learned, and reinforcement-learned architecture in this paper.

4.7 Negotiation trajectories: qualitative case studies

The aggregate statistics of Section 4.6 summarize $9 \times 2 \times 40 = 720$ negotiation-days into bootstrap means and pairwise tests; this subsection looks directly at four individual historical days, reconstructed from the per-call logs already saved during data collection, to give the aggregate numbers qualitative texture. No negotiation was re-run and no model was called again to produce these examples: every value and every action below is exactly what the model emitted at the time, read back from the stored JSON response for that call. One limitation of this reconstruction is unavoidable: the call log records only the language model's own emitted action each turn, never its counterpart's offer (which was visible to the model inside its prompt but was not separately persisted), so the figures below show one side's concession path only, not a two-sided price-convergence plot; a day index was also not logged directly, so individual historical days within one table were recovered by detecting resets in the turn/attempt sequence (a non-decreasing run of turns is one day; a drop back to an earlier turn marks the start of the next).

Figure 9 shows two ordinary, rule-compliant negotiations, one closed and one not. On the left, `llama3.1:8b` as carrier concedes downward across five turns against IS, from an opening

offer of \$2734.31 to \$2604.91 – a smooth, monotonically non-increasing path consistent with the hard-band rule every architecture in this tournament obeys (Section 3.1). On the right, `qwen2.5:72b` as shipper concedes upward from \$2800.00 to \$2975.00 across five offers against OPT2, and is still rejected at the final turn – a reminder that a reasonable-looking, rule-compliant concession path does not by itself guarantee a deal, consistent with this model's non-perfect match rate reported in Table 5.

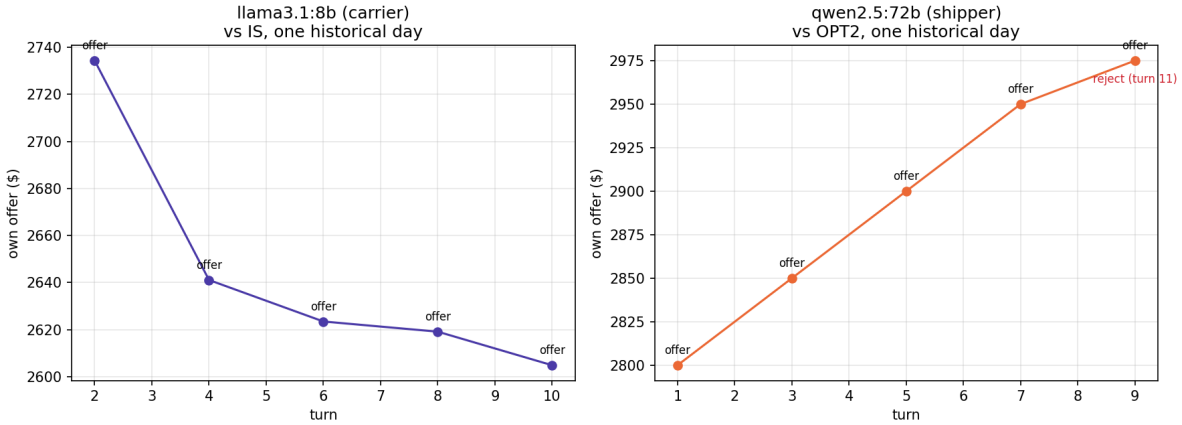


Figure 9: Two ordinary negotiations, reconstructed from saved per-call logs. Left: `llama3.1:8b` (carrier) vs. IS, conceding from \$2734.31 to \$2604.91 over five turns. Right: `qwen2.5:72b` (shipper) vs. OPT2, conceding from \$2800.00 to \$2975.00 over five offers before being rejected at the final turn. Only the language model's own emitted offers are shown; the counterpart's offers were not separately persisted during data collection.

Figure 10 shows `deepseek-r1:70b`'s collapse directly rather than only as a summary statistic: on three successive logged days, the model's first decision on one carrier table (turn 2) used up its whole retry budget – three call attempts, the original plus `max_retries=2` – and the table was forfeited each time. Eight of the nine attempts ended at essentially the same ∼92 s wall-clock latency against the 90 s request timeout, and the ninth failed earlier, after 11.7 s, with a server-side error from the local Ollama server; no attempt produced a usable response, the same pattern Table 6 reports in aggregate.

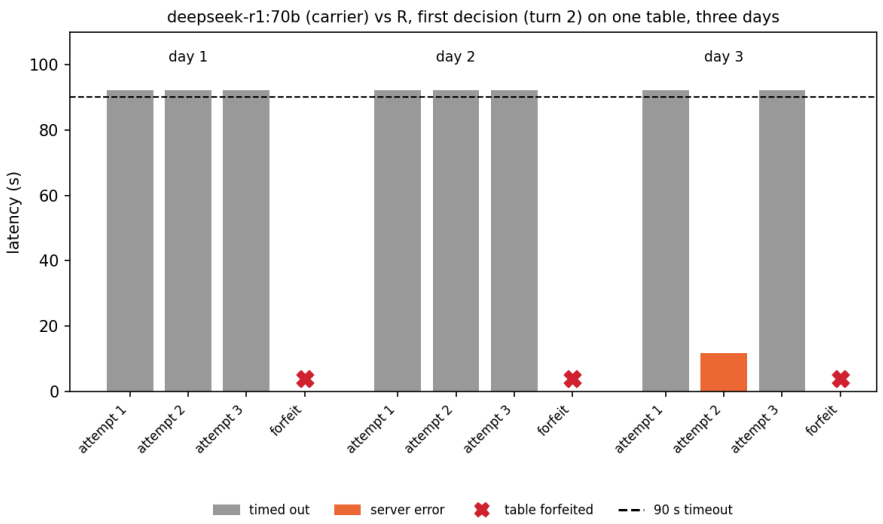


Figure 10: `deepseek-r1:70b` (carrier) vs. R, the model's first decision (turn 2) on one table on three successive logged days. Each bar is one call attempt: grey bars timed out near the 90 s request ceiling (dashed line), the orange bar failed earlier with a server-side error, and the red cross after each group of three attempts marks the table's forfeit once the retry budget was exhausted. No attempt produced a usable response.

Table 7 shows a different kind of failure: in one historical day, `deepseek-r1:7b` (shipper, vs. IS) repeatedly confuses itself about its own action schema rather than failing on latency or format. At turn 3 it twice emits `accept` with a non-null `amount` – contradicting the schema's own stated rule that `amount` must be null for any action other than `offer` – before a third attempt pivots to a legal `offer` that is used; at turn 7 it emits a schema-legal `accept` (`amount=null`) that is still not accepted (nothing on the table yet satisfied its own band), again pivoting to `offer` on the retry; at turn 9, shown in full below, it repeats the identical non-null-`amount` `accept` three consecutive times, exhausts its retry budget, and the table forfeits. This is the same model whose first-try structured-output compliance (Table 6) trails every other small-tier model on the shipper side; this example shows directly what that lower compliance rate consists of in one specific instance.

Table 7: `deepseek-r1:7b` (shipper, vs. IS), turn 9 of one historical day: three consecutive illegal `accept` attempts before the table forfeits. Reconstructed from the saved call log; no negotiation was re-run.

Attempt	Action	Amount	Outcome
0	accept	3022.19	retried: `amount` must be null for `accept`
1	accept	3022.19	retried: same schema violation repeated
2	accept	3022.19	retried: same schema violation repeated
–	forfeit	–	retry budget exhausted (`exhausted_retries`)

5 Discussion

Three results now stand together, each visible only because this tournament tests every architecture, hand-designed or learned, against genuine same-side competition under the same statistical protocol, rather than treating any one family's agents as a special case.

The first is a training-time gap: a reinforcement-learned policy trained without competitive pressure collapses in match rate once rival agents compete with it under scarcity (Section 4.3).

The second is a train/deployment feature mismatch: a policy trained *with* competitive pressure avoids that collapse, but is undone on realized payoff because the feature it depends on is not presented identically in training and at deployment (Section 4.4).

The third is an inference-time constraint specific to one model on our hardware: a large language model prompted with the same rules and the same opponents loses to this tournament's two strongest engines in the overwhelming majority of comparisons (32 of 36 against OPT2/ML2), with the one partial exception (`llama3.3:70b`) still never reaching significance against either, and the largest model in its own family (`deepseek-r1:70b`) failing for a third, distinct reason: a latency ceiling, on the single-GPU workstation of Section 3.7, that it hit on nearly every call, unlike its own smaller siblings and its same-tier rivals, and that did not respond to disabling its explicit reasoning step.

On the carrier side of a balanced market, this tournament's best-performing architectures are not the simplest rule-based ones, nor the most expensive search-based one, but the two families that compute or learn an opponent-aware offer directly: continuous expected-utility optimization with an explicitly fitted opponent belief (OPT2) and offer imitation learned from self-play together with a separately learned accept classifier (ML2); on the shipper side of the same balanced market, by contrast, OPT2 and ML2 rank seventh and eighth of nine (Table 3), so this opponent-awareness advantage does not generalize across roles. Sophistication in how an offer is computed is not, by itself, sufficient to guarantee a good outcome once competition from rival agents and scarcity are introduced: Section 4.3's match-rate collapse shows that an architecture (RL) trained to a plausible-looking reward in an isolated environment can fail specifically and severely once deployed against rival agents, in exactly the market condition designed to apply that competitive pressure, while Section 4.4's finding shows that training against genuine competitive pressure (RL2) is not sufficient either, if a feature the resulting policy depends on is not presented identically between training and deployment. Both findings would be invisible to an evaluation protocol built only from isolated one-on-one duels or a single fixed market condition, which is the gap this paper's many-sided, scarcity-varying, statistically corrected tournament was built to close.

5.1 Comparison with prior automated-negotiation and LLM-negotiation benchmarks

Section 2 positioned this tournament against three prior literatures; the results above let that comparison be made concretely rather than only structurally.

Against the GENIUS/ANAC ecosystem [12]: de Jonge [4] showed that a strategy with no opponent modeling or learning at all (MiCRO) beats the strongest submitted ANAC agents, because the classic linear-additive domains make opponent-aware sophistication unnecessary for winning a single isolated duel. This tournament's own ranking (Section 4.1) shows the opposite pattern once a populated market and real matching are added: the top of the carrier ranking is held by architectures that either learn an offer from data (the imitation-learned ML2, the reinforcement-learned RL) or compute one by optimizing against a modeled counterpart threshold (OPT and, with an explicitly fitted belief, OPT2), while the simplest architectures with neither property (R and, to a lesser extent, RV) occupy the bottom of the carrier ranking in every market condition (Table 2). The two results do not contradict each other – MiCRO's advantage is specific to ANAC's classic domains' utility-function structure, and this tournament deliberately keeps that structure simple while making the market structure itself (scarcity, rival density, a real matching stage) the source of difficulty instead – but together they show that whether opponent-aware sophistication pays off depends on which axis of the environment is actually hard, a question no single benchmark answers for both axes at once.

Against learned negotiation dialogue [8, 11]: those systems are trained and evaluated in the same isolated-bilateral setting their architecture assumes, so their own evaluation protocol has no equivalent of this tournament's match-rate collapse (Section 4.3). Our RL architecture

inherits the identical training assumption – reward from closing a deal in an isolated environment – and this tournament is, to our knowledge, the first time that specific assumption's cost under competition from rival agents and scarcity has been measured directly, rather than left untested by an evaluation protocol that never introduces rival agents.

Against the 2023–2026 LLM-negotiation benchmarks [1, 2, 13, 21]: our own results echo their shared finding that a single success-rate or deal-rate number undersells what is actually happening. `llama3.3:70b` posts the single highest raw shipper-side profit of all twelve participants in its comparison (Table 5) without its comparisons against OPT2/ML2 reaching significance, and `deepseek-r1:70b`'s near-total match-rate collapse (Section 4.6) is a specific, diagnosable inference-time latency artifact of our hardware and 90-second timeout, not a general reasoning failure of that model family – exactly the kind of decomposition Zhang et al. [21] and Oh et al. [13] argue deal-rate alone would hide, and consistent with Carrasco Pollo et al. [2]'s caution that a single LLM-negotiation ranking is rarely the full story. What none of these benchmarks include is a rival roster of hand-designed and learned (non-LLM) architectures inside the same populated, scarcity-varying market: our finding that language models lose to this tournament's two strongest engines in the large majority of comparisons (Section 4.6) is, to our knowledge, the first time that specific comparison – an LLM negotiator against a genuinely competitive non-LLM field, under real matching and scarcity – has been measured directly, echoing in bilateral price negotiation the same direct-policy-versus-model-and-search pattern Lehrach et al. [10] report in general game-playing.

This paper's contribution, stated against that background, is not a new negotiation domain, a new learned architecture, or a new LLM-evaluation metric considered alone: each of those already exists in one of the three literatures above. It is the combination of (i) a populated, many-sided market cleared by a real matching mechanism rather than one isolated bilateral exchange, (ii) scarcity varied systematically across every possible exclusion pair rather than fixed at one condition, and (iii) one shared, statistically corrected protocol applied uniformly to rule-based, search-based, optimization-based, imitation-learned, reinforcement-learned, and large-language-model-based agents at once, letting all six families be ranked against each other directly rather than compared only within their own sub-literature's separate benchmark, with (iv) one normalized negotiation distance (Eq. (7)) as the common scale on which eight of the nine engine architectures value, optimize, or learn their offers.

This study has several limitations.

The market model (Section 3.2) is a synthetic, mean-reverting price process over six fixed routes, calibrated for internal consistency rather than fit to real freight-market data, so the absolute magnitudes reported here should not be read as predictions for a real market; the relative comparison between architectures under the same synthetic market is the result this paper makes.

Every architecture ran frozen, with no within-tournament parameter adaptation, so this paper measures nine fixed policies against each other and not an open-ended co-adaptation process; each architecture's hyperparameters (concession rates, particle counts, optimizer step counts, and so on) were set once rather than tuned per architecture or per market condition, so a reported ranking reflects one hyperparameter setting per architecture, not necessarily each architecture's best achievable performance.

The paired significance tests themselves have three unaddressed limitations: the underlying log-space Ornstein-Uhlenbeck price process (Section 3.2) generates serially correlated draws across consecutive negotiation indices rather than independent and identically distributed ones, and the paired Wilcoxon signed-rank test's standard assumptions do not account for that correlation; we do not report which zero-difference handling method (Wilcoxon's, Pratt's, or zero-split) was used on the rare pairs whose per-day outcomes tied exactly; and we report only Holm-Bonferroni-corrected significance for each of the 36 comparisons per condition and role, not a standardized effect size alongside it.

We deliberately report RL2 as actually deployed, with its confirmed train/deployment feature mismatch left unpatched, rather than silently correcting it before evaluation, since the gap between an architecture's potential and its actually-measured performance is itself part of what this tournament is designed to surface; a corrected RL2, retrained with real portfolio signal available during training as well as at deployment, is a direct next step rather than a result reported here. A reward-design alternative to that occupancy-feature explanation does not hold up under direct inspection of the training code: RL2's reward already includes an explicit margin-based term identical in form to RL's own terminal reward (the same bounded value-bar function of the closing price) whenever its table closes and it wins the match, conditioned on winning rather than omitted outright, so a training signal indifferent to realized margin is not a plausible competing explanation for its low realized payoff.

The language-model comparison adds its own limitations beyond those above: each model was prompted once, with one fixed system prompt and one fixed disclosure policy, so these results characterize that specific prompting choice rather than each model's best achievable performance under prompt engineering; the three fixed same-side rivals (R, OPT2, ML2) were chosen to span this tournament's weak/strong/learned range rather than to include every architecture, so a language model's record against the other six architectures is not directly measured; and, as reported in Section 4.6, the DeepSeek-R1 family's carrier-role and shipper-role data were collected under different explicit reasoning settings after a mid-study methodological change, which we verified does not alter this family's headline latency-collapse finding but which is disclosed rather than hidden; and every latency figure, including that collapse, was measured on one workstation whose 24 GB GPU cannot hold any of the 70B-class models whole (Section 3.7), so different hardware or a longer timeout could change the latency-driven results.

The trajectory case studies of Section 4.7 are similarly limited by construction: the call log used to reconstruct them never persisted a counterpart's offer, only the language model's own emitted action each turn, so those figures and that table show one side's concession path, not a two-sided price-convergence plot.

Future work includes retraining RL2 with real portfolio signal available during training, a hyperparameter sweep across architectures and market conditions, extending the scarcity design to ratios other than seven-of-nine, calibrating the market model against real freight-brokering data, and persisting the counterpart's offer directly during future data collection so that a trajectory analysis can be two-sided rather than reconstructed from one side's logs alone.

6 Conclusion

We built nine negotiation-engine architectures on a shared interface and compared them, for the first time in this project, inside a single many-sided competitive tournament rather than isolated duels, under three market conditions spanning a balanced market and two systematically-rotated scarcity conditions, with every pairwise comparison corrected for the resulting 36 simultaneous tests. Two results from that tournament would not have been visible to a narrower evaluation: a reinforcement-learned policy trained without competitive pressure suffers a severe, condition-specific match-rate collapse once rival agents compete with it under scarcity, and a reinforcement-learned policy trained *with* competitive pressure avoids that collapse but is instead undone, on realized payoff, by a confirmed mismatch between a feature's value during training and during deployment.

Extending the same protocol to nine language models tested them head-to-head against three of this tournament's nine architectures (R, OPT2, ML2) rather than leaving that class of agent untested: against the two strongest of those three, OPT2 and ML2, language models lost significantly in 32 of 36 comparisons and won none, confirming that the architectural advantages identified above are not artifacts of excluding a currently-popular class of agent, while the one partial exception (`llama3.3:70b`) and the one severe, inference-time collapse on

our hardware (`deepseek-r1:70b`) show that a language model's negotiating competence in this setting depends on which specific model and tier is deployed at least as much as on the fact that it is a language model at all. Together these results argue that a negotiation architecture's quality cannot be fully assessed from isolated duels or a single fixed market condition alone, and that a many-sided, scarcity-varying, statistically corrected tournament is a practical way to surface the specific failure modes – competitive collapse, and train/deployment gaps – that matter once such an architecture is actually deployed into a market with rival agents.

Declarations

Funding. This research was conducted as part of the authors' work in the AI Lab at Freight Technologies Inc. and received no external grant funding.

Competing interests. Eduardo de Avila-Armenta, Raúl Alejandro Aquino-Ibarra, Jorge Octavio Nicolás-Díaz, and Umberto Leon-Dominguez are employed by Freight Technologies Inc., a company that develops automated freight-brokering and negotiation technology directly related to the subject matter of this paper. This relationship is disclosed as a potential competing interest.

Data and Code Availability

The simulation engine, all nine negotiation architectures, the LLM validation harness, and the raw result logs analyzed in this paper (including the per-call logs reconstructed in Section 4.7, `data/llm_validation/*_calls.jsonl`, and the summary files `tier_summary.csv` and `tier_family_summary.csv` referenced in Section 4.6, and `data/validation/pct_outcomes.csv` with the percentage columns of Tables 2-3, computed from the stored results by `scripts/_compute_pct_outcomes.py`) are available at github.com/AILabFr8tech/wehaveadeal (the paths above are relative to its `paper/` folder) and archived on Zenodo at doi:10.5281/zenodo.24122038.

References

[1] Sahar Abdelnabi, Amr Gomaa, Sarath Sivaprasad, Lea Schönherr, and Mario Fritz. LLM-deliberation: Evaluating LLMs with interactive multi-agent negotiation games. arXiv:2309.17234, 2023.

[2] Jorge Carrasco Pollo, Ioannis Kapetangeorgis, Joshua Rosenthal, and John Hua Yao. [Re] benchmarking LLM capabilities in negotiation through scoreable games. arXiv:2602.18230, 2026. Also to appear in Transactions on Machine Learning Research.

[3] Peter I. Cowling, Edward J. Powley, and Daniel Whitehouse. Information set Monte Carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2): 120–143, 2012. doi: 10.1109/TCIAIG.2012.2200894.

[4] Dave de Jonge. An analysis of the linear bilateral ANAC domains using the MiCRO benchmark strategy. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI 2022)*, pages 223–229, 2022. doi: 10.24963/ijcai.2022/32.

[5] Dave de Jonge. Theoretical properties of the MiCRO negotiation strategy. *Autonomous Agents and Multi-Agent Systems*, 2024. doi: 10.1007/s10458-024-09678-1. Advance online publication.

[6] Tamara C. P. Florijn, Pinar Yolum, and Tim Baarslag. A negotiator's backup plan: Optimal concessions with a reservation value. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024)*, 2024. Extended abstract; arXiv:2404.19361.

[7] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962. doi: 10.2307/2312726.

[8] He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. Decoupling strategy and generation in negotiation dialogues. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2333–2343, 2018. doi: 10.18653/v1/D18-1256.

[9] Henrik Kristensen and Tommy Gärling. The effects of anchor points and reference points on negotiation process and outcome. *Organizational Behavior and Human Decision Processes*, 71(1):85–94, 1997.

[10] Wolfgang Lehrach, Daniel Hennes, Miguel Lázaro-Gredilla, et al. Code world models for general game playing. In *Proceedings of the 14th International Conference on Learning Representations (ICLR 2026)*, 2026. Preprint: arXiv:2510.04542 (2025).

[11] Mike Lewis, Denis Yarats, Yann N. Dauphin, Devi Parikh, and Dhruv Batra. Deal or no deal? end-to-end learning for negotiation dialogues. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2443–2453, 2017. doi: 10.18653/v1/D17-1259.

[12] Raz Lin, Sarit Kraus, Tim Baarslag, Dmytro Tykhonov, Koen Hindriks, and Catholijn M. Jonker. Genius: An integrated environment for supporting the design of generic automated negotiators. *Computational Intelligence*, 30(1):48–70, 2014. doi: 10.1111/j.1467-8640.2012.00463.x. Online-first 2012; issue date 2014 used here.

[13] Jihwan Oh, Murad Aghazada, Yooju Shin, Se-Young Yun, and Taehyeon Kim. MERIT: Feedback elicits better bargaining in LLM negotiators. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL 2026), Volume 1: Long Papers*, pages 25202–25223, 2026.

[14] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[15] Hannes M. Petrowsky, Lea Boecker, Yannik A. Escher, et al. The power and peril of first offers in negotiations: A conceptual, meta-analytic, and experimental synthesis. *Organizational Behavior and Human Decision Processes*, 191:104448, 2025. doi: 10.1016/j.obhdp.2025.104448.

[16] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[17] Michael Schaerer, David D. Loschelder, and Roderick I. Swaab. Bargaining zone distortion in negotiations: The elusive power of multiple alternatives. *Organizational Behavior and Human Decision Processes*, 137:156–171, 2016.

[18] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[19] Kian Siong Tey, Michael Schaerer, Nikhil Madan, and Roderick I. Swaab. The impact of concession patterns on negotiations: When and why decreasing concessions lead to a distributive disadvantage. *Organizational Behavior and Human Decision Processes*, 165: 153–166, 2021. doi: 10.1016/j.obhdp.2021.05.003.

[20] Dirk Van Poucke and Marc Buelens. Predicting the outcome of a two-party price negotiation. *Journal of Economic Psychology*, 23(1):67–76, 2002.

[21] Erica Zhang, Fangzhao Zhang, Aneesh Pappu, Batu El, Jose Blanchet, Susan Athey, Jiashuo Liu, and James Zou. TERMS-bench: Diagnosing LLM negotiation agents beyond deal rate. arXiv:2605.13909, 2026.