STRATEGIC BEHAVIOR IN LARGE LANGUAGE MODELS DEPENDS CRITICALLY ON GAME STRUCTURE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) are increasingly deployed as strategic agents. However, it remains unclear how their strategic behavior changes with game structure. To address this question, we conduct self-play experiments with 15 LLMs across 10 complete information games, 4 complex incomplete information environments, and an electricity-sector carbon-market bidding game. We find that strategic behavior depends critically on game structure. Equilibrium attainment in sequential games averages 63.2%, about 18 percentage points higher than simultaneous games, and 12 of 15 models perform better in sequential games. Moreover, attaining equilibrium does not imply selecting the Pareto-optimal equilibrium. In complex incomplete information environments, outcome-based metrics alone are insufficient, as models can achieve high scores while exhibiting low-quality interaction processes. To diagnose such failures, we propose Strategic Failure Detection (SFD), which decomposes strategic behavior into planning, coordination, action validity, and outcome. Applying SFD to the carbon-market game improves planning and coordination and raises welfare and terminal payoff achievement rate for DeepSeek-V3 and GPT-4o. These results show that process-level detection can make strategic behavior more observable and easier to improve.

1 INTRODUCTION

Large language models (LLMs) are increasingly used as strategic decision-making agents in domains such as diplomatic negotiation (Xu et al., 2025; Duffy et al., 2026; Mukobi et al., 2023), resource allocation and market trading (Liu et al., 2026), and cooperative control (Sun et al., 2025), where their decisions have real consequences. In these settings, answering questions correctly is not enough. Agents must understand the structure of strategic interaction, anticipate opponents' responses, choose coordinated optimal actions, and execute valid actions under environmental rules. This raises a key question: *how does the strategic behavior of LLMs change as game structure changes?*

Classical game theory provides equilibrium concepts as a reference for strategic interaction (Osborne & Rubinstein, 1994; Fudenberg & Tirole, 1991), while behavioral game theory focuses on how players actually act, including when they exhibit bounded rationality, coordination failure, or stable behavioral patterns (Camerer, 2003). Existing studies have examined how LLMs perform in repeated games (Akata et al., 2025; Fan et al., 2024; Qiaozhu Mei, 2024; Lorè & Heydari, 2024; Affonso, 2026; Fontana et al., 2025), simple games with complete information (Duan et al., 2024; Guo et al., 2026; Lin et al., 2025; Yin et al., 2025; Tang et al., 2026; Davoudi et al., 2026; Chen et al., 2025), and interactions among models or with human participants (Akata et al., 2025; Qiaozhu Mei, 2024; Fontana et al., 2025; Bakhtin et al., 2022). These studies show that LLMs can exhibit recognizable strategic and social behavioral patterns (Akata et al., 2025; Qiaozhu Mei, 2024; Lorè & Heydari, 2024; Ariel Flint Ashery, 2025; Federico Germani, 2025). However, evaluations often focus on the outcomes of repeated interactions (Akata et al., 2025; Fontana et al., 2025). Little is known about how these models behave when game structure shifts from simultaneous to sequential, when games contain multiple equilibria, or when environments involve incomplete information and complex behavioral trajectories (Li et al., 2026; Sun et al., 2025; Mukobi et al., 2023; Ma et al., 2024; Lin et al., 2026). We conduct a series of self-play experiments to address these questions. We evaluate 15 LLMs in complete-information games, incomplete-information games, and an electricity-sector carbon-market bidding game (Liu et al., 2026), as shown in Figure 1. These games are designed to help us understand how agents actually act in strategic interactions of various structures.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

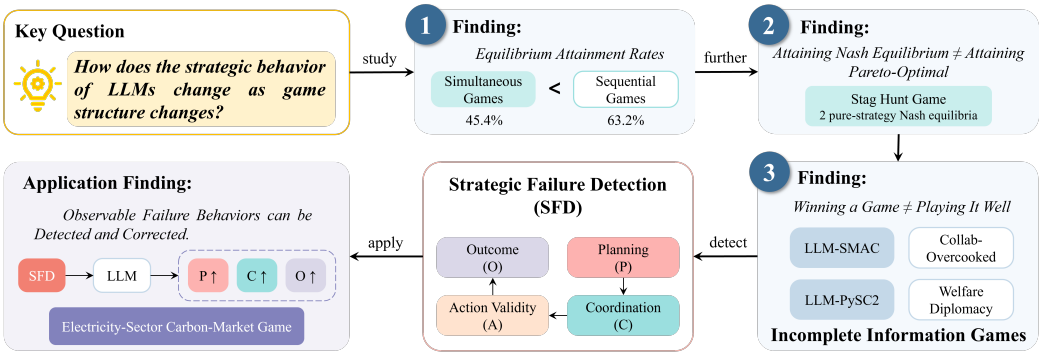


Figure 1: Overview of our study. Rather than a fixed pattern, the strategic behavior of LLM agents shifts with game structure and is only partially revealed by outcome-based metrics. Motivated by this finding, we propose Strategic Failure Detection, which enables structured detection of observable failures and can improve strategic behavior in the carbon-market game.

We find that the strategic behavior of LLMs depends critically on game structure. Nash equilibrium attainment is about 18 percentage points higher in sequential games than in simultaneous games. Sequential games allow round-by-round decisions, so reasoning depth can translate into decision advantage, whereas simultaneous games require a single commitment and failures are more common. This capability does not simply grow with parameter scale. Across families, parameter count has no stable relationship with game performance. Only within the same family does performance improve with scale. More importantly, reaching equilibrium does not mean selecting the optimal equilibrium. In the Stag Hunt game with two pure-strategy Nash equilibria, the 15 models exhibit three behavioral patterns: perfect cooperation, conservative equilibrium, and frequent uncoordinated behavior. The three models with the highest overall average equilibrium attainment rates, GPT-o1, DeepSeek-R1, and GPT-OSS-120B, frequently behave in an uncoordinated manner in Stag Hunt.

Building on this finding, we further focus on more complex incomplete-information game environments. In LLM-PySC2, all eight models win Task 3, yet they do so through two distinct behavioral modes: winning with zero kills and winning through high-intensity engagement. In Welfare Diplomacy, 11 of 12 models reach proficiency above 0.889, but their Nash welfare remains far below the social reference value. Although models can achieve high scores under outcome-based metrics, they struggle to exhibit high-quality interactive behavior.

To this end, we further investigate whether failure behaviors during interaction can be targeted and improved. We decompose the game process into four stages: strategic planning, coordination, action validity, and outcome, converting overall failure into structured behavioral analysis, which we call Strategic Failure Detection (SFD). Finally, we apply SFD to a carbon-market game based on Chinese electricity-sector data. We find that enabling SFD significantly improves models' planning and coordination behavior and raises welfare and terminal payoff achievement rate, showing that observable failure behaviors can be detected and corrected.

Our contributions are as follows:

1. We find that strategic behavior in LLMs depends critically on game structure. Across complete information games, equilibrium attainment is about 18 percentage points higher in sequential games than in simultaneous games, and within the same model family, performance improves with scale.
2. We show that equilibrium attainment does not imply optimal equilibrium selection. In the Stag Hunt game, models predominantly exhibit three behavioral patterns: stable Pareto-optimal coordination, stable conservative coordination, and frequent uncoordinated behavior.
3. Across four complex incomplete information games, we find that outcome-based metrics alone are insufficient to evaluate model performance, and that process-based metrics are necessary to capture strategic behavior.
4. In an electricity-carbon market bidding game, we demonstrate that observable failure behaviors can be detected and corrected, as SFD improves LLMs planning and coordination and raises welfare and terminal payoff achievement rate.

2 Related Work

2.1 Strategic Behavior of LLMs in Decision-Making

Whether and how LLMs exhibit strategic behavior in decision-making has become a distinct stream of research at the intersection of behavioral game theory and machine behavior. A growing body of work has established that LLMs deviate from Nash equilibrium in canonical games, often cooperating more than human players in repeated Prisoner's Dilemma settings while remaining sensitive to contextual framing, and model family (Akata et al., 2025; Fan et al., 2024; Qiaozhu Mei, 2024; Lorè & Heydari, 2024; Affonso, 2026; Fontana et al., 2025). Framing effects and emergent social conventions further modulate these dispositions: source framing can trigger systematic bias (Federico Germani, 2025), and populations of LLM agents can develop collective conventions that are not predicted by individual-level rationality (Ariel Flint Ashery, 2025). In more complex multi-agent environments, LLM-based agents have been deployed in Diplomacy (Bakhtin et al., 2022; Guan et al., 2024; Xu et al., 2025; Duffy et al., 2026; Mukobi et al., 2023), poker (Huang et al., 2024), StarCraft II (Li et al., 2026), and collaborative Overcooked tasks (Sun et al., 2025). Beyond behavioral characterization, LLM agents have also been used to analytically derive Nash equilibria, showing that structured reasoning can recover closed-form solutions in high-dimensional games (Liu et al., 2026). However, this literature does not yet provide an empirical characterization of three coupled questions. The first question is how strategic behavior shifts with game structure across a controlled suite of games. The second is whether equilibrium attainment implies optimal equilibrium selection. The third is whether observable failures can be decomposed into interpretable stages and then corrected. To address this gap, we conduct a large-scale empirical study spanning ten complete information games, four complex incomplete information environments, and a carbon-market bidding game.

2.2 Strategic Game Benchmarks for LLMs

Strategic game benchmarks form a complementary line of work that supplies evaluation infrastructure. Early efforts concentrated on classical matrix games: GTBench assembles ten games that vary along information completeness and determinism (Duan et al., 2024), and TMGBench extends coverage to all 144 canonical 2×2 games and to sequential, parallel, and nested compositions (Guo et al., 2026). A second strand places LLMs in richer environments, including text-based StarCraft II (Ma et al., 2024) and poker with external solvers (Lin et al., 2026). A third strand shifts attention from outcomes to reasoning processes and robustness. GAMEBoT decomposes game decisions into modular subproblems and checks intermediate steps against rule-based ground truth (Lin et al., 2025). WGSR-Bench organizes wargame evaluation around situational awareness, opponent risk modeling, and policy generation (Yin et al., 2025). DSGBench combines six strategic games with fine-grained metrics and decision-trajectory tracking (Tang et al., 2026). Same Game, Different Story treats strategic robustness as invariance of action distributions under payoff-preserving prompt changes (Davoudi et al., 2026). LLMsPark evaluates LLMs in strategic gaming contexts (Chen et al., 2025). These benchmarks are designed to compare models under controlled conditions. They do not, however, aim to answer the empirical question that motivates our study: how strategic behavior changes with game structure. Our work is discovery-oriented rather than benchmark-oriented. Instead of proposing another evaluation suite, we conduct a large-scale study across a broad set of games with complete and incomplete information and a carbon-market bidding game, and we introduce Strategic Failure Detection (SFD) as a framework that decomposes failures into planning, coordination, action validity, and outcome. We then show that SFD-guided intervention can correct observable failures and improve welfare and terminal payoff achievement.

3 Method

3.1 Self-Play Method

Under the default self-play protocol, both interacting roles are played by the same LLM, with role-specific prompts assigned to each. At stage t, agent i generates a structured output that includes its action $a_{i,t}$, its prediction of the opponent's action $\hat{a}_{-i,t}$, and a rationale. The prediction is retained as an independent record for the Planning dimension of SFD.

In simultaneous games, although the two agents submit in a fixed order, neither can observe the other's final action before submitting its own. In sequential games, the later actor can observe the final action taken by the earlier actor. Each stage maintains a separate prediction record. Prompts contain the game rules, the current state, and the required output format, but exclude any history from previous independent runs. Exceptions are LLM-SMAC and LLM-PySC2, where the opponent is a rule-based agent supplied by the environment. All outputs are parsed as JSON and mapped to a legal action whenever possible.

3.2 Strategic Failure Detection

SFD is a framework for localizing observable strategic failures. It decomposes each decision stage into four diagnostic dimensions: planning (P), coordination (C), action validity (A), and outcome (O). These dimensions are evaluated in the order $P \to C \to A \to O$. For each model, we record structured outputs, apply the diagnostic checks in this order, and assign the earliest failed dimension as the sample's diagnostic label. A targeted intervention is then selected. Specifically, P failures trigger state- or payoff-grounded reasoning, C failures trigger a communication step before action selection, and A failures trigger stricter output constraints. Samples with no detected failure receive no intervention. We compare the resulting SFD condition with its paired condition without SFD. This design tests whether detection-guided intervention changes observable behavior.

We formalize a stochastic game as $\Gamma = \langle N, \mathcal{S}, \{A_i\}_{i\in N}, q, \{u_i\}_{i\in N}\rangle$, where s_t is the state at stage t, $a_t = (a_{1,t}, \ldots, a_{n,t})$ is the joint action, and $U_i^{(t)}(\cdot; s_t)$ denotes agent i's continuation payoff from stage t.

Planning (P). Given agent i's recorded prediction $\hat{a}_{-i,t}$, we compute the best response $\mathrm{BR}_{i,t}(\hat{a}_{-i,t}; s_t) \in \arg\max_{a'\in A_i} U_i^{(t)}(a', \hat{a}_{-i,t}; s_t)$, and the relative regret

$$\rho_{i,t} = \frac{U_i^{(t)}(\mathrm{BR}_{i,t}; s_t) - U_i^{(t)}(a_{i,t}, \hat{a}_{-i,t}; s_t)}{\left|U_i^{(t)}(\mathrm{BR}_{i,t}; s_t)\right|}.$$

Planning succeeds, $P_{i,t} = 1$, if and only if $\rho_{i,t} \le \varepsilon_P$. In multi-stage games, a planning failure is assigned only to the stage where it occurs and is not counted repeatedly in later stages.

Coordination (C). Coordination evaluates whether individually reasonable actions form a mutually compatible joint action. For discrete games with multiple equilibria, $C_t = 1$ if and only if the realized joint action belongs to the pure-strategy Nash equilibrium set $C^*(\Gamma_t)$. For continuous-action games, let b_t^* be the continuation-equilibrium profile and define the aggregate continuation value $\overline{V}^{(t)}(a_t) = \sum_{i\in N}\left[\pi_{i,t}(a_t; s_t) + V_i^{(t+1)}(s_{t+1}(a_t))\right]$. The relative joint loss is

$$L_t = \frac{\overline{V}^{(t)}(b_t^*) - \overline{V}^{(t)}(a_t)}{\left|\overline{V}^{(t)}(b_t^*)\right|},$$

and $C_t = 1$ if and only if $L_t \le \varepsilon_C$. This approximate-equilibrium criterion avoids the computational burden of exact equilibrium computation and measures efficiency loss in aggregate continuation value.

Action validity (A). This dimension is scored from output logs. Specifically, it checks whether the structured output can be parsed into a legal action. Parsing and validation are performed against the admissible action space, and failures are recorded as $\perp$. Action validity is distinct from strategic quality. A valid action may be strategically poor, whereas an appropriate strategy may fail to be realized if the output is malformed.

Outcome (O). The outcome dimension uses the outcome-based metric supplied by the environment. Missing evidence is recorded as N/A rather than as failure. For finite discrete games, the reference solution is computed by enumeration or backward induction, while for the carbon-market game it is the pre-specified finite-grid market solution described in Appendix A.4.

4 EXPERIMENTAL ENVIRONMENTS AND SETUP

4.1 EXPERIMENTAL ENVIRONMENTS

Complete Information Games. We evaluate five simultaneous games: Prisoner's Dilemma (PD), Stag Hunt, Battle of the Sexes (BoS), Wait-Go, and Duopolistic Competition (Hua et al., 2024). We also evaluate five sequential games: Escalation, Monopoly, Hot-Cold, Draco, and Trigame (Hua et al., 2024). These games vary in conflict, coordination, multiple equilibria, and state transitions. All 15 models play each game in self-play. The primary baseline metrics are Nash-equilibrium attainment and Pareto-optimal Nash-equilibrium attainment.

Incomplete Information Games. We investigate the strategic behavior of LLMs across two adversarial environments, one cooperative environment, and one general-sum game environment. All four environments involve hidden information, but its form varies. In LLM-SMAC and LLM-PySC2 (Li et al., 2026), we examine decentralized combat against rule-based opponents and macro-level adversarial control strategies, respectively, under fog of war where enemy units are observable only within the field of view. Collab-Overcooked (Sun et al., 2025) serves as a cooperative game environment, in which we study collaboration mechanisms as task difficulty increases. Here task knowledge is held unilaterally, as only one agent knows the complete task, and collaboration requires communication to bridge the information gap. Welfare Diplomacy (Mukobi et al., 2023) is a more complex general-sum game environment, where we investigate whether agents can sustain overall social welfare when interacting with exploiters. In this environment, negotiation is private, and each country observes only public historical actions, making it impossible to observe other countries' intentions or bilateral negotiation content. We report both outcome metrics and process metrics, including kill/death ratio, progress completeness, exploitability, and basic proficiency.

Electricity-Sector Carbon-Market Bidding Game. We construct an electricity-sector carbon-market bidding game from historical data for 1,623 Chinese electricity-sector firms (Liu et al., 2026). We retain 395 firms that satisfy the pre-specified screening rules of negative price elasticity and non-negative allowance surplus. Price elasticity is derived from econometric estimation, and allowance gaps are derived from compliance-accounting data. Each instance samples four firms representing a large state-owned enterprise, a medium state-owned enterprise, a gas-fired enterprise, and a private enterprise. The four firms submit bids over four trading stages. A common clearing mechanism determines the clearing price and traded quantities, and each firm's payoff combines trading profit or loss across stages with the terminal value of its remaining allowances. DeepSeek-V3 and GPT-4o are evaluated under paired conditions with and without SFD. SFD is instantiated by detecting and correcting the model's Planning, Coordination, Action Validity, and Outcome records at each stage.

4.2 EXPERIMENT SETUP

We evaluate 15 large language models: GPT-OSS-20B, GPT-OSS-120B, GPT-4o, and GPT-o1 from the GPT family, Llama3.1-8B and Llama3.1-70B from the Llama family, Qwen3-8B, Qwen3-32B, and Qwen3-Max from the Qwen family, DeepSeek-8B, DeepSeek-32B, DeepSeek-R1, and DeepSeek-V3 from the DeepSeek family, as well as GLM-4.5 and Kimi-K2. We set the sampling temperature to 0.1 for all models except GPT-o1, which uses its default temperature setting, and we set the maximum output length to 4096 tokens. Complete information games are repeated for 10 independent runs, while complex incomplete information game environments are repeated for five runs. Models with fewer than 20B parameters are invoked locally via Ollama, whereas all other models are invoked remotely through APIs. More experimental details are provided in the appendices A.2.

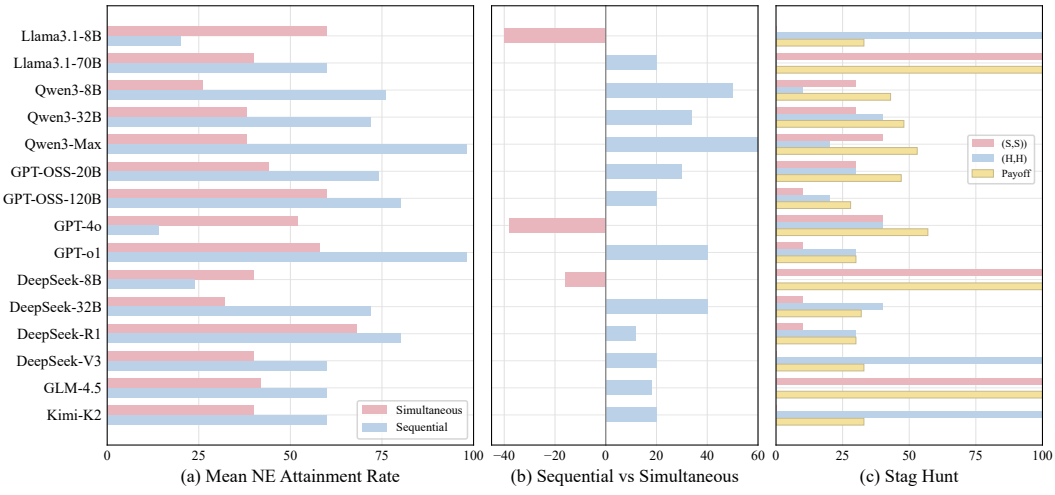
5 RESULTS

5.1 STRATEGIC BEHAVIOR IN COMPLETE INFORMATION GAMES

How does the strategic behavior of LLM agents change with game structure? To address this question, we evaluate 15 LLM agents across 10 complete-information games, comprising 5 simultaneous and 5 sequential games. As shown in Table 1, the average Nash equilibrium attainment rate in sequential games is 63.2%, substantially higher than the 45.4% observed in simultaneous games, a gap of

Table 1: Nash-equilibrium attainment rate across the 10 complete-information games. Best results are in bold.

	Simultaneous Games					Sequential Games				
Model	**PD**	**Stag Hunt**	**BoS**	**Wait-Go**	**Duopoly**	**Escalation**	**Monopoly**	**Hot-Cold**	**Draco**	**Trigame**
Llama3.1-8B	100	**100**	**100**	0	0	0	100	0	0	0
Llama3.1-70B	100	100	0	0	0	0	100	**100**	**100**	0
Qwen3-8B	0	40	40	50	0	90	100	0	**100**	90
Qwen3-32B	20	70	50	40	10	70	90	60	90	50
Qwen3-Max	30	60	70	30	0	**100**	**100**	**100**	**100**	**100**
GPT-OSS-20B	50	60	50	50	10	100	100	10	70	90
GPT-OSS-120B	100	30	40	60	70	100	100	0	100	100
GPT-4o	90	**80**	20	50	20	20	40	10	0	0
GPT-o1	100	40	50	30	70	100	100	100	90	100
DeepSeek-8B	0	100	70	10	20	30	20	40	20	10
DeepSeek-32B	10	50	60	40	0	90	100	20	80	70
DeepSeek-R1	**100**	40	30	**70**	**100**	100	100	0	100	100
DeepSeek-V3	100	100	0	0	0	100	0	100	0	100
GLM-4.5	100	100	0	0	10	100	70	0	30	100
Kimi-K2	100	100	0	0	0	100	100	100	0	0
Avg.	67	**71**	39	29	21	73	**81**	43	59	60



(a) Mean NE Attainment Rate (b) Sequential vs Simultaneous (c) Stag Hunt

Figure 2: Strategic Behavior Performance in Complete Information Games. (a) Average Nash equilibrium attainment rate of each model in simultaneous and sequential games. (b) Gap between the average Nash equilibrium attainment rates of each model in simultaneous and sequential games. (c) Strategic behavior performance in Stag Hunt game.

approximately 18 percentage points. More importantly, this sequential–simultaneous asymmetry is pervasive: 12 of the 15 models attain higher equilibrium rates in sequential games than in simultaneous games, whereas only DeepSeek-8B, GPT-4o, and Llama3.1-8B exhibit the opposite pattern, as shown in Figure 2. This suggests that the asymmetry is not driven by a few outlier models but reflects a general property of LLM agents.

This asymmetry can be attributed to the different decision requirements of the two game classes. Sequential games allow agents to observe the preceding action and respond accordingly, and the round-by-round structure enables models to convert reasoning depth into a decision advantage. Reasoning-oriented models such as GPT-o1 and Qwen3-Max approach perfect equilibrium attainment in sequential games, as reported in Table 1. In contrast, simultaneous games require both players to commit to an action without observing the opponent's choice, leaving the payoff matrix as the only basis for decision-making, which is considerably more challenging for LLMs. In coordination-intensive simultaneous games such as Battle of the Sexes, Wait–Go, and Duopolistic Competition, several large models including DeepSeek-V3, GLM-4.5, and Kimi-K2 perform poorly and often fail to attain equilibrium altogether. These results indicate that the strategic behavior of LLM agents depends critically on game structure.



Figure 3: Strategic Behavior Performance of LLMs in LLM-SMAC, LLM-PySC2, and Collab-Overcooked environments.

5.2 Strategic Behavior in Stag Hunt Game

Does equilibrium attainment equate to Pareto-optimal equilibrium selection? To address this question, we further examine model performance in the Stag Hunt game, which contains two pure-strategy Nash equilibria. In this game, the Pareto-optimal Nash equilibrium (S,S) yields a payoff of (3,3), whereas the other equilibrium (H,H) yields (1,1). Action H is a maximin strategy, as it guarantees a payoff of one unit regardless of the opponent's choice. This game therefore decouples equilibrium attainment from equilibrium selection.

We observe three distinct behavioral patterns across 15 models. DeepSeek-8B, Llama3.1-70B, and GLM-4.5 consistently coordinate on (S,S), achieving Pareto-optimal cooperation with an average overall payoff of 100%, as shown in Figure 2 (c). Llama3.1-8B, DeepSeek-V3, and Kimi-K2 consistently coordinate on (H,H), also attaining a 100% NE attainment rate but with an average payoff of only 33%. In contrast, some models frequently exhibit uncoordinated behavior. For example, GPT-o1, DeepSeek-R1, and GPT-OSS-120B have the highest average NE attainment rates across simultaneous and sequential games, yet in the Stag Hunt game their NE attainment rate is only 30%–40% and their average overall payoff is 28%–30%, the lowest among all models. Given that each run is an independent one-shot event, this uncoordinated behavior cannot be explained by history-dependent strategies. This aligns with the findings of Akata et al. (2025), who showed that LLMs can identify strategic regularities but cannot respond to them reliably. Therefore, merely reaching an equilibrium does not indicate whether a model has selected the Pareto-optimal equilibrium.

5.3 Strategic Behavior in Incomplete Information Games

Can outcome-based metrics adequately evaluate the strategic behavior of LLMs in incomplete information games? To answer this question, we evaluate model performance using both process metrics and outcome metrics across 4 incomplete information game environments.

Our results show that although models can achieve high scores under outcome-based evaluation, they fail to achieve high-quality game interactions, as shown in Figure 3, revealing the limitations of outcome-based evaluation. We find that in the adversarial environment LLM-SMAC Li et al. (2026), all the models win only in two large-scale combat tasks, 1c3s5z and 3s5z, while in the remaining tasks, including the symmetric 2s3z, win rates are nearly zero. These failures do not imply that the models lack adversarial capability. In 2s3z, their kill/death ratios still reach 0.38-0.63, nearly matching the opponent, yet they cannot convert this advantage into victory, exposing deficiencies in micro-operations such as focus fire and positioning. In LLM-PySC2 Li et al. (2026), all 8 models achieve full scores on Task 3, but their behaviors differ markedly. Qwen3-8B, Llama3.1-70B, and

Table 2: Strategic Behavior Performance of DeepSeek-V3 and GPT-4o in the carbon-market game with and without SFD, evaluated using process metrics and an outcome metric. TPAR denotes Terminal Payoff Achievement Rate and is computed as the sum of terminal U_i across the 4 firms in a market divided by the sum of real data terminal U_i. For BR gap↓, lower values are better. Best results are in bold.

Model	Condition	Process metrics			Outcome metrics		
		P	C	A	Welfare	BR gap↓	TPAR
DeepSeek-V3	Without SFD	68.3%	80.8%	100.0%	129.3%	14.656	68.6%
	With SFD	**77.7%**	**82.5%**	100.0%	154.7%	9.947	76.4%
GPT-4o	Without SFD	16.5%	75.8%	100.0%	141.9%	8.363	72.5%
	With SFD	74.0%	79.2%	100.0%	**165.2%**	**4.605**	**83.6%**

GLM-4.5 complete the task through high-intensity engagement, with kill/death ratios of 43.5–58.9, whereas DeepSeek-32B, GPT-OSS-120B, DeepSeek-R1, DeepSeek-V3, and Kimi win with zero kills. Because the victory condition of Task 3 requires only survival and not engagement, both high-intensity engagement and complete avoidance can win. In this case, only process metrics such as kill/death ratio can reveal the difference between the two behavioral modes.

In the cooperative environment Collab-Overcooked Sun et al. (2025), Llama3.1-70B maintains a 100% success rate across all 30 tasks spanning six difficulty levels, yet its progress completion rate drops from 68.34% to 26.53%, and its collaboration initiation and response scores are only about one-eighth of those of humans. In addition, GPT-OSS-20B and Qwen3-32B achieve near-perfect completion in most tasks, but their progress completion drops by 30–40% in the two hardest difficulty levels. In Welfare Diplomacy Mukobi et al. (2023), as shown in Figure 4, among 12 defense models, 11 reach proficiency above 0.889, but Nash welfare remains between 1.76 and 3.45, far below the social reference value of 4.833. Therefore, outcome-based metrics cannot fully evaluate model performance. Only when combined with process-based metrics can they better reflect models' strategic behavior.

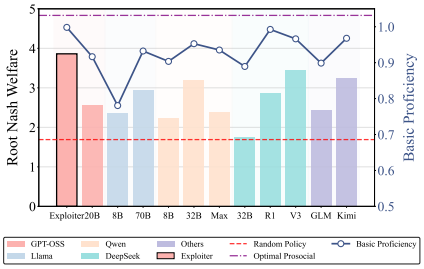


Figure 4: Strategic Behavior Performance of LLMs in the Welfare Diplomacy environment, where the columns represent the root Nash welfare of exploiters and the welfare diplomacy agent, the red dashed line indicates the root Nash welfare under random policy, the purple dashed line indicates the root Nash welfare under the optimal prosocial policy, and the line represents the basic proficiency score.

5.4 Strategic Behavior in Electricity-Sector Carbon-Market Game

Given the behavioral signatures that models exhibit across different games, we investigate *whether observable failure behaviors can be detected and corrected, and how these agents behave when participating in real-world decision-making*. To this end, we model the electricity-sector carbon-market bidding game as a dynamic game with complete information, where firms and regulators interact sequentially, with the tested models each acting as the four firms. To examine whether observable failure behaviors can be detected and corrected, we employ the SFD method in the carbon-market game, using DeepSeek-V3 and GPT-4o as game agents.

We compare the agents' performance with and without SFD, considering both process and outcome metrics, as shown in Table 2. SFD substantially improves planning (P) for both models. DeepSeek-V3 rises from 68.3% to 77.7%, while GPT-4o rises from 16.5% to 74.0%, a 57.5 percentage-point gain. Coordination (C) also improves modestly. DeepSeek-V3 from 80.8% to 82.5%, and GPT-4o from 75.8% to 79.2%. Action validity (A) remains 100% because the bidding actions are numerical and relatively simple for the models.

Relative to the without-SFD condition, SFD consistently improves all outcome metrics across both models. For Welfare, DeepSeek-V3 improves from 129.3% to 154.7%, and GPT-4o from 141.9% to 165.2%. The best-response gap (BR gap) narrows from 14.656 to 9.947 for DeepSeek-V3, and from 8.363 to 4.605 for GPT-4o. Terminal payoff achievement rate (TPAR) increases from 68.6% to 76.4% for DeepSeek-V3, and from 72.5% to 83.6% for GPT-4o. Notably, although baseline planning

differs sharply between the two models, their planning rates converge after SFD, suggesting that SFD largely closes the gap in basic planning capability.

Crucially, evaluating models solely on terminal TPAR can reverse the conclusion drawn from process-based evaluation. GPT-4o has a higher baseline TPAR than DeepSeek-V3, yet its planning pass rate is only 16.5%, implying that roughly 83.5% of firm-stage decisions contain planning failures. Because trading gains and losses under market clearing are approximately zero-sum across firms, these planning errors need not appear immediately in final payoffs. They materialize as welfare gains through terminal clearing prices only after SFD improves planning. This again shows that terminal-only evaluation can distort the characterization of agent behavior.

Taken together, these results show that, in the carbon-market game, SFD significantly improves the planning and coordination behavior of LLMs and raises welfare and TPAR, showing that observable failure behaviors can be detected and corrected. Full statistics and process metric decompositions are provided in Appendix C.3.

6 Discussion

Recent work has shown that LLMs can act as strategic agents in game-theoretic settings, yet their behavior often diverges from human play and remains difficult to predict (Akata et al., 2025; Duan et al., 2024; Fan et al., 2024; Qiaozhu Mei, 2024; Lorè & Heydari, 2024; Affonso, 2026; Fontana et al., 2025).In this work, we evaluate 15 LLMs across a suite of complete information games, complex incomplete information environments, and a carbon-market bidding game, and we use SFD to diagnose observable failures in the carbon-market game. On the one hand, the evaluation reveals clear differences across models and game structures. On the other hand, even the strongest models do not consistently achieve high game outcomes, and several failures reflect planning or coordination limits rather than purely strategic errors.

Our findings also shed light on the relationship between model family, model scale, and strategic performance. Within the same family, larger models tend to perform better, as shown in Table 1, suggesting that scaling can improve strategic competence when models share a common training lineage. However, parameter count alone is not a reliable predictor of game-playing ability. Models of similar size can differ substantially, and very large models do not automatically outperform smaller ones. Consequently, scaling claims are most meaningful within a family, while cross-family comparisons should be interpreted with caution. The DeepSeek family is excluded from this comparison because its distilled members inherit from different Qwen versions and do not form a single scaling sequence.

Our study has two main limitations. First, actions in the carbon-market game are simple, so all models passed the action validity check in SFD. This dimension therefore did not reveal how invalid actions contribute to failed behavior. We also did not apply SFD to complex games with incomplete information. Second, some reasoning models failed because they continued to think until they exhausted the maximum token budget and produced no actual output. These models consequently performed poorly in the game. Future work should evaluate SFD in more complex games and should separate strategic failures from failures caused by output-budget exhaustion.

7 Conclusion

In this work, we find that the strategic behavior of LLMs changes with game structure and differs across model families. Within the same family, game performance generally improves as parameter scale increases, but parameter count alone cannot predict strategic ability. Moreover, attaining equilibrium does not mean selecting a Pareto-optimal equilibrium. Furthermore, in complex incomplete information games, outcome-based metrics alone are insufficient, and process-based metrics are necessary to characterize strategic behavior. To address these issues, SFD provides diagnosis and intervention for observable failures. In the carbon-market game, SFD improves the planning and coordination behavior of DeepSeek-V3 and GPT-4o and leads to better outcome metrics.

AI USE STATEMENT

In this work, generative AI tools were used in two distinct roles. First, LLMs served as the evaluated subjects of the study: all reported agent behavior was produced by the fifteen models named in Section 4.2 using the documented prompts. Second, we used generative AI tools for language polishing to improve the readability of the research paper. We reviewed all AI-assisted work. In particular, the AI-assisted language polishing was checked by the authors to ensure that it did not alter the scientific content or claims. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work evaluates LLMs in game-theoretic simulations and public multi-agent benchmarks; it involves no human subjects and no personal or sensitive data. The carbon-market game is a data-calibrated simulation, not a validated model of a real emissions-trading system, and its results should not be read as policy or trading guidance.

REPRODUCIBILITY STATEMENT

We will release the environment definitions, prompts, model settings, codes in an anonymized repository upon publication. The appendix reports the model names, temperatures, effective token limit, repetition counts, and complete information result tables.

REFERENCES

F. M. Affonso. Large language models converge on competitive rationality but diverge on cooperation across providers and generations. arXiv preprint arXiv:2604.18596, 2026.

Elif Akata, Lion Schulz, Julian Coda-Forno, Seong Joon Oh, Matthias Bethge, and Eric Schulz. Playing repeated games with large language models. *Nature Human Behaviour*, 9:1380–1390, 2025. doi: 10.1038/s41562-025-02172-y.

Andrea Baronchelli Ariel Flint Ashery, Luca Maria Aiello. Emergent social conventions and collective bias in LLM populations. *Science advances*, pp. eadu9368, 2025.

Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mojtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander H. Miller, Sasha Mitts, Adithya Renduchintala, Stephen Roller, Dirk Rowe, Weiyan Shi, Joe Spisak, Alexander Wei, David Wu, Hugh Zhang, and Markus Zijlstra. Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022.

Colin F. Camerer. *Behavioral Game Theory: Experiments in Strategic Interaction*. Princeton University Press, 2003.

Junhao Chen, Jingbo Sun, Xiang Li, Haidong Xin, Yuhao Xue, Yibin Xu, and Hao Zhao. LLMsPark: A benchmark for evaluating large language models in strategic gaming contexts. In *30th Conference on Empirical Methods in Natural Language Processing, EMNLP 2025*, 2025.

Seyed Pouyan Mousavi Davoudi, Arshia Gharagozlou, Alireza Amiri-Margavi, Amin Gholami Davodi, and Hamidreza Hasani Balyani. Same game, different story: A minimal conservative strategic robustness benchmark for large language model agents. *arXiv preprint arXiv:2607.19670*, 2026.

Jinhao Duan, Renming Zhang, James Diffenderfer, Bhavya Kailkhura, Lichao Sun, Elias Stengel-Eskin, Mohit Bansal, Tianlong Chen, and Kaidi Xu. GTBENCH: Uncovering the strategic reasoning limitations of LLMs via game-theoretic evaluations. In *38th Conference on Neural Information Processing Systems, NeurIPS 2024*, 2024.

Alexander Duffy, Samuel J. Paech, Ishana Shastri, Elizabeth Karpinski, Baptiste Alloui-Cros, Tyler Marques, and Matthew Lyle Olson. Democratizing diplomacy: A harness for evaluating any large language model on full-press diplomacy. In *40th AAAI Conference on Artificial Intelligence, AAAI 2026*, 2026.

Caoyun Fan, Jindou Chen, Yaohui Jin, and Hao He. Can large language models serve as rational players in game theory? a systematic analysis. In *38th AAAI Conference on Artificial Intelligence, AAAI 2024*, 2024.

Giovanni Spitale Federico Germani. Source framing triggers systematic bias in large language models. *Science advances*, pp. eadz2924, 2025.

Nicolo Fontana, Francesco Pierri, and Luca Maria Aiello. Nicer than humans: How do large language models behave in the prisoner's dilemma? In *International Association for The Advancement of Artificial Intelligence Conference on Weblogs and Social Media—International AAAI Conference on Web and Social Media*, 2025.

Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, 1991.

Zhenyu Guan, Xiangyu Kong, Fangwei Zhong, and Yizhou Wang. Richelieu: Self-evolving llm-based agents for ai diplomacy. In *38th Conference on Neural Information Processing Systems, NeurIPS 2024*, 2024.

Yu Guo, Haochuan Wang, and Xiachong Feng. Game-theoretic evaluation of strategic reasoning in large language models: From complete coverage to compositional complexity. *Neurocomputing*, 675:133006, 2026.

Wenyue Hua, Ollie Liu, Lingyao Li, Alfonso Amayuelas, Julie Chen, Lucas Jiang, Mingyu Jin, Lizhou Fan, Fei Sun, William Wang, Xintong Wang, and Yongfeng Zhang. Game-theoretic LLM: Agent workflow for negotiation games. arXiv preprint, 2024.

Chenghao Huang, Yanbo Cao, Yinlong Wen, Tao Zhou, and Yanru Zhang. PokerGPT: An end-to-end lightweight solver for multi-player Texas Hold'em via large language model. arXiv preprint, 2024.

Zongyuan Li, Yanan Ni, Runnan Qi, Chang Lu, Lumin Jiang, Xu Xiaojie, Xiangbei Liu, Pengfei Li, Yunzheng Guo, Zhe Ma, et al. Llm-pysc2: Starcraft ii learning environment for large language models. *Advances in Neural Information Processing Systems*, 38:127595–127657, 2026.

Min Lin, Enyan Dai, Hui Liu, Xianfeng Tang, Yuliang Yan, Zhenwei Dai, Jingying Zeng, Zhiwei Zhang, Fali Wang, Hongcheng Gao, Chen Luo, Xiang Zhang, Qianyu He, and Suhang Wang. How far are llms from professional poker players? revisiting game-theoretic reasoning with agentic tool use. In *ICLR 2026*, January 2026.

Wenye Lin, Jonathan Roberts, Yunhan Yang, Samuel Albanie, Zongqing Lu, and Kai Han. Gamebot: Transparent assessment of llm reasoning in games. In *63rd Annual Meeting of the Association for Computational Linguistics, ACL 2025*, 2025.

Wenxuan Liu, Xiyuan Zhou, Xinlei Wang, Leandros Tassiulas, Jianwei Huang, and Junhua Zhao. An llm agent-based framework for analytical characterization of nash equilibria. *Nexus*, pp. 100107, 2026.

Nunzio Lorè and Babak Heydari. Strategic behavior of large language models and the role of game structure versus contextual framing. *Scientific Reports*, pp. 18490, 2024.

Weiyu Ma, Qirui Mi, Yongcheng Zeng, Xue Yan, Yuqiao Wu, Runji Lin, Haifeng Zhang, and Jun Wang. Large language models play starcraft ii: benchmarks and a chain of summarization approach. In *38th Conference on Neural Information Processing Systems, NeurIPS 2024*, 2024.

G. Mukobi, A. Bakhtin, N. Brown, et al. Welfare diplomacy: Benchmarking language model cooperation. arXiv preprint arXiv:2310.08901, 2023.

Martin J. Osborne and Ariel J. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

Walter Yuan Matthew O Jackson Qiaozhu Mei, Yutong Xie. A turing test of whether ai chatbots are behaviorally similar to humans. *Proceedings of the National Academy of Sciences of the United States of America*, pp. e2313925121, 2024.

Haochen Sun, Shuwen Zhang, Lujie Niu, Lei Ren, Hao Xu, Hao Fu, Fangkun Zhao, Caixia Yuan, and Xiaojie Wang. Collab-overcooked: Benchmarking and evaluating large language models as collaborative agents. In *30th Conference on Empirical Methods in Natural Language Processing, EMNLP 2025*, 2025.

Wenjie Tang, Yuan Zhou, Keyan Cheng, Erqiang Xu, Liquan Xiao, and Minne Li. Dsgbench: A diverse strategic game benchmark for evaluating llm-based agents in complex decision-making environments. In *ICASSP 2026 - 2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2026.

Kaixuan Xu, Jiajun Chai, Sicheng Li, Yuqian Fu, Yuanheng Zhu, and Dongbin Zhao. Dipllm: Fine-tuning llm for strategic decision-making in diplomacy. In *42nd International Conference on Machine Learning, ICML 2025*, 2025.

Qiyue Yin, Pei Xu, Qiaozhe Li, Shengda Liu, Shengqi Shen, Tong Wang, Yihong Han, Xiaonan Zhao, Likun Yang, Shiyue Cao, Shiyu Qiu, Yuxuan Liu, Shizhao Yu, Lei Cui, Chengxin Yan, Jie Sun, Xiangquan Tang, and Kaiqi Huang. Wgsr-bench: Wargame-based game-theoretic strategic reasoning benchmark for large language models. *arXiv*, 2025.

A Complete Information Games

This appendix describes the complete information game suite, the prompt templates used in experiments, the SFD intervention results, and the hyperparameter settings. The suite contains five simultaneous games and five sequential games; SFD intervention experiments are conducted on the three simultaneous games (BoS, Wait–Go, Duopolistic Competition) with paired direct/SFD comparisons.

A.1 Environment Descriptions

Simultaneous games.

1. **Prisoner's Dilemma (PD).** Actions: C (Cooperate) and D (Defect). The payoff matrix is

	C	D
C	$(3,3)$	$(0,5)$
D	$(5,0)$	$(1,1)$

   There is a unique pure-strategy Nash equilibrium, (D, D), with payoff $(1, 1)$. The profile (C, C) is Pareto optimal but is not a Nash equilibrium.

2. **Stag Hunt.** Actions: S (Stag) and H (Hare). The payoff matrix is

	S	H
S	$(3,3)$	$(0,1)$
H	$(1,0)$	$(1,1)$

   There are two pure-strategy Nash equilibria: (S, S) and (H, H). The (S, S) equilibrium is Pareto optimal; (H, H) is the conservative/maximin equilibrium.

3. **Battle of the Sexes (BoS).** Actions: O (Opera) and F (Football). The payoff matrix is

	O	F
O	$(2,1)$	$(0,0)$
F	$(0,0)$	$(1,2)$

   There are two pure-strategy Nash equilibria: (O, O) and (F, F). A mixed-strategy equilibrium also exists, but the experiments use pure-strategy attainment.

4. **Wait–Go.** Actions: W (Wait) and G (Go). The payoff matrix is

	W	G
W	$(0,0)$	$(0,2)$
G	$(2,0)$	$(-4,-4)$

There are two asymmetric pure-strategy Nash equilibria: (W,G) and (G,W). The symmetric profiles (W,W) and (G,G) are not equilibria.

5. **Duopolistic Competition.** Actions: six quantity choices $a_1,\dots,a_6$. The 6×6 payoff matrix is

	a_1	a_2	a_3	a_4	a_5	a_6
a_1	$(0,0)$	$(0,9)$	$(0,14)$	$(0,15)$	$(0,12)$	$(0,5)$
a_2	$(9,0)$	$(7,7)$	$(5,10)$	$(3,9)$	$(1,4)$	$(-1,-5)$
a_3	$(14,0)$	$(10,5)$	$(6,6)$	$(2,3)$	$(-2,-4)$	$(-2,-5)$
a_4	$(15,0)$	$(9,3)$	$(3,2)$	$(-3,-3)$	$(-3,-4)$	$(-3,-5)$
a_5	$(12,0)$	$(4,1)$	$(-4,-2)$	$(-4,-3)$	$(-4,-4)$	$(-4,-5)$
a_6	$(5,0)$	$(-5,-1)$	$(-5,-2)$	$(-5,-3)$	$(-5,-4)$	$(-5,-5)$

There is a unique pure-strategy Nash equilibrium, (a_3,a_3), with payoff $(6,6)$. The profile (a_2,a_2) is Pareto optimal with payoff $(7,7)$ but is not a Nash equilibrium.

Sequential games. For the sequential games, we describe the game tree and the unique subgame-perfect equilibrium (SPE) path obtained by backward induction.

1. **Escalation.** Player 1 moves first. If Player 1 chooses E_1, the game ends at payoff $(0,0)$. If Player 1 chooses E_2, Player 2 chooses B_1, ending at $(1,-2)$, or B_2, after which Player 1 chooses A_1, ending at $(-2,1)$, or A_2, ending at $(-1,-1)$. Backward induction gives the unique SPE path: Player 1 chooses E_1, so the outcome is $(0,0)$. Off path, Player 2 chooses B_2, and Player 1 then chooses A_2.
2. **Monopoly.** Player 1 (entrant) moves first. Choice M_1 ends at $(0,2)$. Choice M_2 leads to Player 2 (incumbent) choosing I_1, ending at $(2,1)$, or I_2, ending at $(-1,-1)$. The unique SPE path is Player 1 chooses M_2 and Player 2 chooses I_1, giving outcome $(2,1)$.
3. **Hot–Cold.** Player 1 moves first. If Player 1 chooses H_1, Player 2 chooses between $(3,2)$ and $(2,3)$, and chooses the latter. If Player 1 chooses H_2, Player 2 chooses between $(1,4)$ and $(4,1)$, and chooses the former. Player 1 therefore chooses H_1. The unique SPE path is H_1 followed by Player 2's second action, giving outcome $(2,3)$.
4. **Draco.** A three-stage game. The tree is:

$$A_1 \to \begin{cases} B_1 : (5,5) \\ B_2 \to \begin{cases} A_1 : (2,2) \\ A_2 : (3,4) \end{cases} \end{cases}$$

$$A_2 \to \begin{cases} B_1 : (4,5) \\ B_2 \to \begin{cases} A_1 : (5,3) \\ A_2 : (2,2) \end{cases} \end{cases}$$

Backward induction gives the unique SPE path A_1,B_1, with outcome $(5,5)$. Off path, at the final nodes Player 1 chooses A_2 after A_1B_2 and A_1 after A_2B_2; Player 2 then chooses B_1 after both A_1 and A_2.

5. **TriGame.** A three-stage game with tree:

$$A_1 \to \begin{cases} B_1 \to \begin{cases} A_1 : (20,3) \\ A_2 : (0,4) \end{cases} \\ B_2 \to \begin{cases} A_1 : (2,5) \\ A_2 : (3,4) \end{cases} \end{cases}$$

$$A_2 \to \begin{cases} B_1 \to \begin{cases} A_1 : (1,5) \\ A_2 : (4,10) \end{cases} \\ B_2 \to \begin{cases} A_1 : (2,1) \\ A_2 : (3,2) \end{cases} \end{cases}$$

Backward induction gives the unique SPE path A_2, B_1, A_2, with outcome $(4, 10)$. Off path, at the final nodes Player 1 chooses A_1 after A_1B_1, A_2 after A_1B_2, A_2 after A_2B_1, and A_2 after A_2B_2; Player 2 then chooses B_2 after A_1 and B_1 after A_2.

A.2 PROMPT TEMPLATES

Every prompt specifies the rules, the current state, the legal actions, and the required structured output. The output schema is a JSON object containing the action and the predicted opponent action (SFD condition) or a tagged action (baseline condition). Simultaneous-game agents submit in a fixed order without observing the other's final action. Sequential-game agents observe only the preceding actor's final action. Each independent run omits the history of previous runs.

System prompt. The system prompt establishes the decision-maker persona:

> You are a fully rational decision maker in game theory, and the following requires you to make decisions that are consistent with your persona.

User prompt (simultaneous games, baseline). The user prompt contains three sections: game introduction, game rules (payoff matrix), and the action instruction.

> **### Game Introduction**
>
> You are playing a one-round game with another player to maximize the reward you can obtain. This game involves two players, Alice and Bob. You are playing as {*self.name*}. You have {*action_number*} choices: {*self.action_names*}. The other player, {*self.the_other_player*}, also has {*action_number*} choices: {*self.action_names*}.
>
> **### Game Rules**
>
> {*rule_descriptions*}
>
> **### Your Action**
>
> Analyse the problem and the negotiation message if there is any. Please choose one of the following actions to maximize your reward. Surround your choice with `<s>` and `</s>` to indicate the start and end of your choice. For example, `<s>choice_1</s>`, `<s>choice_2</s>`, `<s>choice_3</s>`.
>
> Action choices: {*self.action_names*}

SFD-guided action prompt. In the SFD condition, the action section is replaced with a state-conditioned game policy workflow and a strict JSON output contract. The output must contain exactly two fields: `action` and `predicted_opponent_action`.

> **### State-Conditioned Game Policy**
>
> Make the strategic decision yourself using the following internal workflow: 1. Reconstruct the payoff structure from the game rules. 2. Predict the action that the opponent will choose from the rules and the payoff structure. 3. Compare your payoff under every legal action given your prediction. 4. Select the action that maximizes your own reward, and verify it is legal.
>
> Keep your internal reasoning concise. Do not enumerate alternatives, do not repeat the payoff matrix, and do not restate the rules. Decide and output the final JSON.
>
> **### Final Answer**
>
> Return exactly one JSON object with exactly two fields: `action` and `predicted_opponent_action`. The value of `action` must be your own decision and exactly match one legal action. The value of `predicted_opponent_action` must be your prediction of the action that the opponent will choose. The required form is: {`"action": "<one legal action>", "predicted_opponent_action": "<one opponent action>"`}. Your entire visible reply must be that JSON object and nothing else. It must begin with { and end with }, with no explanation, code fence, XML tag, extra key, or any text before or after it.

Table 3: SFD intervention summary on simultaneous games. P = Planning pass rate, C = Coordination (NE) rate, E = Action-validity pass rate, O = Outcome attainment rate, NE = Nash-equilibrium attainment rate, Payoff = average joint payoff. Values are aggregated over 20 paired samples per game–model combination.

Game	Model	Condition	P	C	E	O	NE	Payoff
BoS	DeepSeek-V3	no SFD	0.70	0.15	1.00	0.15	0.15	0.45
		SFD	1.00	0.75	1.00	0.75	0.75	2.25
	GPT-4o	no SFD	1.00	0.20	1.00	0.20	0.20	0.60
		SFD	1.00	1.00	1.00	1.00	1.00	3.00
Wait–Go	DeepSeek-V3	no SFD	0.70	0.50	1.00	0.50	0.50	−2.60
		SFD	1.00	0.35	1.00	0.35	0.35	−3.70
	GPT-4o	no SFD	0.00	0.00	1.00	0.00	0.00	0.00
		SFD	0.40	0.60	1.00	0.60	0.60	0.40
Duopoly	DeepSeek-V3	no SFD	1.00	0.65	1.00	0.65	0.65	9.55
		SFD	1.00	1.00	1.00	1.00	1.00	12.00
	GPT-4o	no SFD	1.00	0.70	1.00	0.70	0.70	9.35
		SFD	0.75	0.75	1.00	1.00	0.75	12.50

A.3 SFD Intervention Results

A.3.1 Summary of Paired Improvements

SFD intervention experiments are conducted on the three simultaneous games (BoS, Wait–Go, Duopolistic Competition) with DeepSeek-V3 and GPT-4o, using a paired design (same instances and seeds across direct and SFD conditions). Table 3 reports the Planning pass rate P, Coordination pass rate C, Execution/validity pass rate E, Outcome pass rate O, Nash-equilibrium attainment rate (NE), and average joint payoff for each game–model combination.

Key observations:

- **Planning (P).** SFD improves Planning for both models on every game where the baseline is below 1.0: DeepSeek-V3 gains 0.30 on BoS and 0.30 on Wait–Go; GPT-4o gains 0.40 on Wait–Go. For games where baseline Planning is already at 1.0, SFD does not degrade it (except a small drop on Duopoly for GPT-4o, from 1.00 to 0.75, which is offset by a gain in Outcome).
- **Coordination (C) / NE attainment.** SFD substantially increases NE attainment in BoS (DeepSeek-V3: $0.15 \to 0.75$; GPT-4o: $0.20 \to 1.00$) and Duopoly (DeepSeek-V3: $0.65 \to 1.00$). Wait–Go shows a more mixed pattern because coordination on (W, G) or (G, W) requires anti-correlated choices; GPT-4o improves from 0.00 to 0.60 after SFD, while DeepSeek-V3 decreases slightly from 0.50 to 0.35.
- **Outcome (O) and payoff.** Joint payoff increases under SFD in four of six game–model pairs (BoS both models, Duopoly both models), and NE-attainment increases in five of six pairs. The Wait–Go game is the exception because its equilibria are asymmetric and SFD may push both players toward the action that looks individually safer, resulting in the discoordinated (G, G) outcome.
- **Execution (E).** Action validity is at 100% for both conditions across all games and models, because the action set is small and the output format is simple.

A.3.2 Complete Paired Records

Table 4 provides the per-sample audit of all 120 paired records (6 game–model combinations $\times$ 20 samples). Each row reports the direct/SFD pair for Planning (Alice; Bob), Coordination, Execution (Alice; Bob), Outcome, NE attainment, clean-sample flag, the first-failure transition from direct to SFD, and the intervention type.

Table 4: Complete paired records for the simultaneous-game SFD experiment. Binary pairs are direct/SFD. For Planning and Action Validity, the two semicolon-separated pairs are Alice and Bob, respectively. IDs are one-based (raw sample index plus one).

Model	Game	ID	P_A, P_B	C	E_A, E_B	O	NE	Clean	First failure D→S	Intervention
DeepSeek-V3	BoS	1	1/1, 0/1	0/0	1/1, 1/1	0/0	0/0	0/0	P→C	targeted
DeepSeek-V3	BoS	2	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none

Continued on next page

Model	Game	ID	P_A,P_B	C	E_A,E_B	O	NE	Clean	First failure D→S	Intervention
DeepSeek-V3	BoS	3	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	BoS	4	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	BoS	5	1/1, 0/1	0/0	1/1, 1/1	0/0	0/0	0/0	P→C	targeted
DeepSeek-V3	BoS	6	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	BoS	7	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	BoS	8	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
DeepSeek-V3	BoS	9	1/1, 0/1	0/0	1/1, 1/1	0/0	0/0	0/0	P→C	targeted
DeepSeek-V3	BoS	10	1/1, 0/1	0/1	1/1, 1/1	0/1	0/1	0/1	P→clean	targeted
DeepSeek-V3	BoS	11	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	BoS	12	1/1, 0/1	0/0	1/1, 1/1	0/0	0/0	0/0	P→C	targeted
DeepSeek-V3	BoS	13	1/1, 0/1	0/1	1/1, 1/1	0/1	0/1	0/1	P→clean	targeted
DeepSeek-V3	BoS	14	1/1, 0/1	0/0	1/1, 1/1	0/0	0/0	0/0	P→C	targeted
DeepSeek-V3	BoS	15	1/1, 0/1	0/1	1/1, 1/1	0/1	0/1	0/1	P→clean	targeted
DeepSeek-V3	BoS	16	1/1, 0/1	0/1	1/1, 1/1	0/1	0/1	0/1	P→clean	targeted
DeepSeek-V3	BoS	17	1/1, 0/1	0/1	1/1, 1/1	0/1	0/1	0/1	P→clean	targeted
DeepSeek-V3	BoS	18	1/1, 0/1	0/1	1/1, 1/1	0/1	0/1	0/1	P→clean	targeted
DeepSeek-V3	BoS	19	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
DeepSeek-V3	BoS	20	1/1, 0/1	0/1	1/1, 1/1	0/1	0/1	0/1	P→clean	targeted
DeepSeek-V3	Wait–Go	1	0/1, 1/1	1/0	1/1, 1/1	1/0	1/0	0/0	P→C	targeted
DeepSeek-V3	Wait–Go	2	0/1, 0/1	0/0	1/1, 1/1	0/0	0/0	0/0	P→C	targeted
DeepSeek-V3	Wait–Go	3	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	Wait–Go	4	0/1, 1/1	1/0	1/1, 1/1	1/0	1/0	0/0	P→C	targeted
DeepSeek-V3	Wait–Go	5	0/1, 1/1	1/0	1/1, 1/1	1/0	1/0	0/0	P→C	targeted
DeepSeek-V3	Wait–Go	6	0/1, 1/1	1/0	1/1, 1/1	1/0	1/0	0/0	P→C	targeted
DeepSeek-V3	Wait–Go	7	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	Wait–Go	8	0/1, 1/1	1/0	1/1, 1/1	1/0	1/0	0/0	P→C	targeted
DeepSeek-V3	Wait–Go	9	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	Wait–Go	10	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	Wait–Go	11	0/1, 1/1	1/0	1/1, 1/1	1/0	1/0	0/0	P→C	targeted
DeepSeek-V3	Wait–Go	12	0/1, 1/1	1/0	1/1, 1/1	1/0	1/0	0/0	P→C	targeted
DeepSeek-V3	Wait–Go	13	0/1, 1/1	1/0	1/1, 1/1	1/0	1/0	0/0	P→C	targeted
DeepSeek-V3	Wait–Go	14	0/1, 1/1	1/0	1/1, 1/1	1/0	1/0	0/0	P→C	targeted
DeepSeek-V3	Wait–Go	15	1/1, 1/1	0/0	1/1, 1/1	0/0	0/0	0/0	C→C	targeted
DeepSeek-V3	Wait–Go	16	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	Wait–Go	17	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	Wait–Go	18	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	Wait–Go	19	1/1, 1/1	0/0	1/1, 1/1	0/0	0/0	0/0	C→C	targeted
DeepSeek-V3	Wait–Go	20	0/1, 1/1	1/0	1/1, 1/1	1/0	1/0	0/0	P→C	targeted
DeepSeek-V3	Duopoly	1	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
DeepSeek-V3	Duopoly	2	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	Duopoly	3	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	Duopoly	4	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
DeepSeek-V3	Duopoly	5	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
DeepSeek-V3	Duopoly	6	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
DeepSeek-V3	Duopoly	7	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	Duopoly	8	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
DeepSeek-V3	Duopoly	9	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
DeepSeek-V3	Duopoly	10	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	Duopoly	11	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	Duopoly	12	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
DeepSeek-V3	Duopoly	13	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
DeepSeek-V3	Duopoly	14	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
DeepSeek-V3	Duopoly	15	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	Duopoly	16	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
DeepSeek-V3	Duopoly	17	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
DeepSeek-V3	Duopoly	18	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
DeepSeek-V3	Duopoly	19	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
DeepSeek-V3	Duopoly	20	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	BoS	1	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	BoS	2	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
GPT-4o	BoS	3	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
GPT-4o	BoS	4	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
GPT-4o	BoS	5	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	BoS	6	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
GPT-4o	BoS	7	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
GPT-4o	BoS	8	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
GPT-4o	BoS	9	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	BoS	10	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	BoS	11	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
GPT-4o	BoS	12	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
GPT-4o	BoS	13	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
GPT-4o	BoS	14	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
GPT-4o	BoS	15	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
GPT-4o	BoS	16	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
GPT-4o	BoS	17	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
GPT-4o	BoS	18	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
GPT-4o	BoS	19	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
GPT-4o	BoS	20	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted

Continued on next page

Model	Game	ID	P_A,P_B	C	E_A,E_B	O	NE	Clean	First failure D→S	Intervention
GPT-4o	Wait–Go	1	0/0, 0/0	0/0	1/1, 1/1	0/0	0/0	0/0	P→P	targeted
GPT-4o	Wait–Go	2	0/0, 0/0	0/0	1/1, 1/1	0/0	0/0	0/0	P→P	targeted
GPT-4o	Wait–Go	3	0/0, 0/1	0/1	1/1, 1/1	0/1	0/1	0/0	P→P	targeted
GPT-4o	Wait–Go	4	0/0, 0/0	0/0	1/1, 1/1	0/0	0/0	0/0	P→P	targeted
GPT-4o	Wait–Go	5	0/0, 0/1	0/1	1/1, 1/1	0/1	0/1	0/0	P→P	targeted
GPT-4o	Wait–Go	6	0/1, 0/1	0/0	1/1, 1/1	0/0	0/0	0/0	P→C	targeted
GPT-4o	Wait–Go	7	0/0, 0/1	0/1	1/1, 1/1	0/1	0/1	0/0	P→P	targeted
GPT-4o	Wait–Go	8	0/0, 0/1	0/1	1/1, 1/1	0/1	0/1	0/0	P→P	targeted
GPT-4o	Wait–Go	9	0/1, 0/0	0/1	1/1, 1/1	0/1	0/1	0/0	P→P	targeted
GPT-4o	Wait–Go	10	0/0, 0/1	0/1	1/1, 1/1	0/1	0/1	0/0	P→P	targeted
GPT-4o	Wait–Go	11	0/0, 0/0	0/0	1/1, 1/1	0/0	0/0	0/0	P→P	targeted
GPT-4o	Wait–Go	12	0/0, 0/1	0/1	1/1, 1/1	0/1	0/1	0/0	P→P	targeted
GPT-4o	Wait–Go	13	0/1, 0/0	0/1	1/1, 1/1	0/1	0/1	0/0	P→P	targeted
GPT-4o	Wait–Go	14	0/0, 0/1	0/1	1/1, 1/1	0/1	0/1	0/0	P→P	targeted
GPT-4o	Wait–Go	15	0/1, 0/1	0/0	1/1, 1/1	0/0	0/0	0/0	P→C	targeted
GPT-4o	Wait–Go	16	0/0, 0/1	0/1	1/1, 1/1	0/1	0/1	0/0	P→P	targeted
GPT-4o	Wait–Go	17	0/0, 0/1	0/1	1/1, 1/1	0/1	0/1	0/0	P→P	targeted
GPT-4o	Wait–Go	18	0/0, 0/0	0/0	1/1, 1/1	0/0	0/0	0/0	P→P	targeted
GPT-4o	Wait–Go	19	0/0, 0/0	0/0	1/1, 1/1	0/0	0/0	0/0	P→P	targeted
GPT-4o	Wait–Go	20	0/0, 0/1	0/1	1/1, 1/1	0/1	0/1	0/0	P→P	targeted
GPT-4o	Duopoly	1	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	Duopoly	2	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	Duopoly	3	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	Duopoly	4	1/0, 1/0	0/0	1/1, 1/1	0/1	0/0	0/0	C→P	targeted
GPT-4o	Duopoly	5	1/0, 1/0	0/0	1/1, 1/1	0/1	0/0	0/0	C→P	targeted
GPT-4o	Duopoly	6	1/0, 1/0	0/0	1/1, 1/1	0/1	0/0	0/0	C→P	targeted
GPT-4o	Duopoly	7	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	Duopoly	8	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	Duopoly	9	1/0, 1/0	0/0	1/1, 1/1	0/1	0/0	0/0	C→P	targeted
GPT-4o	Duopoly	10	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	Duopoly	11	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	Duopoly	12	1/1, 1/1	0/1	1/1, 1/1	0/1	0/1	0/1	C→clean	targeted
GPT-4o	Duopoly	13	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	Duopoly	14	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	Duopoly	15	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	Duopoly	16	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	Duopoly	17	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	Duopoly	18	1/0, 1/0	0/0	1/1, 1/1	0/1	0/0	0/0	C→P	targeted
GPT-4o	Duopoly	19	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none
GPT-4o	Duopoly	20	1/1, 1/1	1/1	1/1, 1/1	1/1	1/1	1/1	clean→clean	none

The complete machine-readable export contains the same paired identifiers together with the direct-baseline diagnosis, intervention components, and source-file hashes; it is included in the reproducibility package.

A.4 Hyperparameters and Reproducibility Settings

SFD thresholds. The SFD diagnostic uses the following pre-registered thresholds, which are identical across the simultaneous-game, sequential-game, and carbon-market implementations:

- Planning threshold: $\varepsilon_P = 0.05$ (relative regret below which a decision is considered a successful best response under the agent's stated prediction);
- Coordination threshold: $\varepsilon_C = 0.10$ (relative joint loss below which the realized profile is considered coordinated with the reference; in discrete matrix games the coordination dimension uses exact Nash-equilibrium membership rather than a continuous loss);
- Outcome threshold: $\delta_O = 0.90$ (fraction of the best-equilibrium joint payoff that the realized joint payoff must reach for Outcome to pass);
- Denominator guard: $\eta = 10^{-6}$ (added to denominators of relative ratios to prevent division by zero).

These thresholds are defined in the scoring code and are fixed before any LLM output is scored; they are not tuned on model outputs.

Model settings. The simultaneous- and sequential-game experiments use the following common settings:

- Sampling temperature: 0.1 for all models except reasoning models that use their default temperature setting;

- Maximum output length: 4096 tokens (model-dependent);
- Number of samples per game–model combination: 20 for the SFD paired experiments;
- Action interface: tagged action (<s>···</s>) for the baseline condition; strict JSON with `action` and `predicted_opponent_action` fields for the SFD condition;
- Maximum action retries: 1 retry on parse failure, with a repair prompt that restates the JSON contract.

Equilibrium definitions. For a finite normal-form game, a joint action is a Nash equilibrium when no unilateral deviation increases the deviating player's payoff. Pareto-optimal Nash equilibria are Nash equilibria not Pareto-dominated by another feasible outcome. These definitions are applied to the realized joint actions; a Pareto-optimal Nash-equilibrium rate cannot exceed the corresponding Nash-equilibrium rate in the same sample. Sequential games are solved by backward induction. The Planning score uses the best response under the agent's recorded opponent prediction. Coordination uses the realized joint action and the applicable equilibrium set. Action Validity uses the JSON and protocol logs. Outcome uses the environment-specific terminal criterion.

B Incomplete Information Games

This appendix describes the four incomplete information environments (LLM-SMAC, LLM-PySC2, Collab-Overcooked, Welfare Diplomacy), their prompt templates, additional results, and experimental settings. These environments are partially observable, multi-agent, and either adversarial, cooperative, or general-sum. Unlike the complete information matrix games, they do not admit a small set of pure-strategy Nash equilibria that can be enumerated directly. Their reference behavior is therefore defined by environment-native task success, process metrics, or cooperative equilibrium constructions.

B.1 Environment Descriptions

LLM-SMAC. LLM-SMAC is the SMAC-compatible task suite provided by LLM-PySC2. It follows the standard StarCraft Multi-Agent Challenge setting: both sides are initialized with fixed unit compositions, and enemy units attack automatically. The key to victory lies in concentrated fire, combat-distance control, and sometimes interaction frequency. The environment is partially observable under fog of war: enemy units are visible only within the field of view, and each agent receives local observations. Text observations include game time, controlled units, nearby ally and enemy units, relevant unit knowledge, valid actions, communication data, and task descriptions. The action space is agent-specific and discrete. For micro-operation agents it includes actions such as <Stop()>, <No_Op()>, <Move_Screen(screen)>, <Move_Minimap(minimap)>, and <Attack_Unit(tag)>, but excludes macro actions such as training or research. The SMAC task suite includes 2s3z, 3s5z, 1c3s5z, 3s5z_vs_3s6z, 2s_vs_1sc, 2c_vs_64zg, and 3s_vs_3z. These are decentralized multi-agent combat tasks rather than matrix games with a small equilibrium set. The reference behavior is therefore not a pure-strategy Nash equilibrium but task success under a competent micro policy. Evaluation uses the kill/death ratio and the win rate (WR). In the original LLM-PySC2 experiments, six LLMs were tested, including GPT-3.5-turbo, GPT-4o-mini, GPT-4o, Claude3-haiku, Llama3.1-8b, and GLM-4-plus. The results show that all models win only in the two large-scale combat tasks 1c3s5z and 3s5z, while in the symmetric 2s3z task their win rates are nearly zero even though kill/death ratios reach about 0.38–0.63. This indicates that the models can trade units but cannot convert local combat advantages into victory.

LLM-PySC2. LLM-PySC2 is an LLM-oriented StarCraft II environment built on PySC2. It provides the full StarCraft II action space, multimodal observation interfaces, a structured game-knowledge database, and a multi-agent communication framework. Observations can be textual or visual. Text observations include local micro-level information and global macro-level information, such as deployment, unit counts, upgrade status, valid actions, communication, and task descriptions. Image observations include game images and feature maps, enhanced with auxiliary lines for coordinate perception. LLM actions are expressed as text actions in the form <ActionName()>, <ActionName(arg0)>, or <ActionName(arg0, arg1)>. These are parsed by an action recognizer and converted into PySC2 action functions. The LLM-PySC2 task group contains eight tasks with three difficulty levels each. Tasks 1–4 are single-agent decision-making scenarios in which

one LLM agent controls multiple units; tasks 5–8 test cooperation among multiple agents controlling units with different tactical roles. Representative tasks include Adept harass, Phoenix harass, intercepting enemy airdrops, mid-scale combat, and large-scale combat. All tasks involve hidden information under fog of war, partial observability, hidden enemy units or intentions, and a knowledge base that must be used selectively. As in LLM-SMAC, there is no classical Nash-equilibrium reference. Performance is measured by kill/death ratio and win rate. In the main text, we further find that all eight evaluated models win Task 3, but through two distinct behavioral modes: winning with zero kills and winning through high-intensity engagement with kill/death ratios of 43.5–58.9. This shows that outcome-only metrics can conceal qualitatively different strategic behavior.

Collab-Overcooked. Collab-Overcooked is an LLM-powered multi-agent benchmark built on Overcooked-AI. It extends the original environment to a chef-and-assistant setting with two key designs: resource isolation and asymmetric task knowledge. The two agents operate in different sub-environments and can exchange resources only through a shared counter. Only one agent knows the complete recipe or task flow, so collaboration requires natural language communication to synchronize task information. The benchmark contains 30 sequential process-specific tasks across six complexity levels. The complexity level is determined by the minimum number of collaborative actions required. Each task has a natural-language recipe and one or more Referential Action Trajectories (RATs) that describe optimal reference action sequences. The action space is agent-specific. Agent Alice can use actions such as `pickup`, `cut`, `stir`, `place_obj_on_counter`, `put_obj_in_utensil`, and `wait`. Agent Bob can use actions such as `pickup`, `cook`, `place_obj_on_counter`, `put_obj_in_utensil`, `fill_dish_with_food`, `bake`, `deliver`, and `wait`. Collaboration capability is decomposed into initiating collaboration and responding collaboration. Evaluation includes success rate (SR), progress completeness (PC), trajectory efficiency score (TES), incremental trajectory efficiency score (ITES), initiating capability (IC), and responding capability (RC). The reference solution is not a Nash equilibrium but the set of annotated RATs; TES and ITES measure order-preserving alignment and the marginal contribution of individual collaborative actions. Experiments over ten LLMs show that most models respond to collaboration better than they initiate it, and that collaboration performance declines as task complexity increases. In particular, Llama3.1-70B maintains a 100% success rate across all 30 tasks, but its progress completion rate drops from 68.34% to 26.53%, and its collaboration initiation and response scores are only about one-eighth of those of humans.

Welfare Diplomacy. Welfare Diplomacy (WD) is a general-sum variant of the zero-sum board game Diplomacy. Standard Diplomacy is a seven-player game in which players maneuver military units, capture supply centers (SCs), and exchange private messages in the full-press version. WD changes the rules in three ways: (1) after each Winter, a player receives Welfare Points (WPs) equal to the difference between the number of SCs they control and their number of units on the board; (2) the game ends after a fixed number of years rather than when one player captures 18 SCs; and (3) each player's objective is to maximize their own accumulated WPs, and there is no single winner. Thus players are incentivized to build as few units as possible, but doing so may invite attacks from other players. The environment is general-sum and permits positive-sum interactions. Information is incomplete because negotiation is private: each country observes only public historical actions and cannot directly observe other countries' intentions or bilateral negotiation content. Actions include unit movement, support, convoy, build, disband, and private messaging. The original paper proves cooperative equilibrium constructions in a toy version of WD. On the toy graph G_n with $n \geq 6$, for $1 \leq k \leq T$ and $k \neq T-2$, the policy profile π^k, in which all players claim their neutral SCs, disband all units in the k-th Winter, and punish deviators, is a Nash equilibrium of $W_{n,T}$. Moreover, π^k Pareto-dominates π^{k+1} for $1 \leq k \leq T-1$, and π^1 is Pareto-efficient when $T \neq 3$. In an asymmetric toy variant $W'_{n,T}$, there exist two Pareto-efficient Nash equilibria π^i and π^j, preferred by different players, which creates a bargaining problem over equilibrium selection. These equilibria are not necessarily subgame-perfect because punishment policies may themselves be costly. Evaluation uses root Nash welfare in self-play, exploitability against one- or two-player exploiter coalitions, and a basic proficiency score. In the main text, we further report that among 12 defense models, 11 reach proficiency above 0.889, but Nash welfare remains between 1.76 and 3.45, far below the social reference value of 4.833. This again shows that high outcome scores do not imply high-quality cooperative interaction.

B.2 PROMPT TEMPLATES

LLM-PySC2 / LLM-SMAC. LLM-SMAC shares the same prompt structure as LLM-PySC2 (system prompt + per-turn observation + action output format), adapted to the SMAC micro-combat setting. The LLM-PySC2 system prompt defines the agent identity, tactical rules, and output format:

> **1. Identity**
>
> You are a Protoss frontline commander, controls several Stalkers. Responsible for providing cover for the main force and restraining enemy forces. Your should command your troops, complete the tactical tasks assigned by the superior. You will have several teams of units, you can command these teams to fight together or perform different tasks.
>
> **2. Rules**
>
> 2.1 Try to kill more and loss less. Usually, concentrating all firepower on the same target (especially the closest enemy) can improve the strike effectiveness.
>
> 2.2 Try to kill enemy as quick as possible, retreat promptly when/before enemy reinforcements arrive.
>
> 2.3 When sacrificing your unit can earn much more profits, you can choose to sacrifice your unit.
>
> 2.4 Use your skills well to achieve optimal tactical results. Especially when controlling support units.
>
> 2.5 Always remember the tactical tasks given by superior. Sometimes you have to sacrifice whole team to ensure the achievement of tactical objectives.
>
> **3. Action Output**
>
> You should make decisions according to observed information, tactic task and rules, give analysis and decisions for each team. For example, if you have 2 teams name as 'Stalker-1' and 'Stalker-2', you should output as:
>
> Analysis:
> xxxxx
> Actions:
> Team Stalker-1:
> xxxxx
> Team Stalker-2:
> xxxxx

The per-turn user prompt provides the game state: game time, team position, controlled units, nearby ally and enemy units (with tag, position, health, energy, weapon cooldown), valid actions with argument descriptions, and the tactical task. An example input for an Oracle team:

> **Game Info**
>
> Time: 0:32
>
> **Team Oracle-1 Info:**
>
> Team minimap position: [50, 32]
>
> Controlled Team Units: Unit: Oracle Tag: 0x100200001 Pos: (67, 59) Health: 100 Energy: 108 Weapon_cooldown: 0
>
> Nearby Ally units: Unit: Observer Tag: 0x100140001 Pos: (10, 70) Health: 70 Weapon_cooldown: 0
>
> Nearby Enemy units: Unit: Drone Tag: 0x101340001 Pos: (54, 40) Health: 40
> . . . (list of enemy Drones, Overlord, Hatchery, SpawningPool, Queens)
>
> Valid Actions:
> `<Stop()>`
> `<No_Operation()>`
> `<Attack_Unit(tag)>`
> `<Move_Screen(screen)>`
> `<Move_Minimap(minimap)>`
> `<Ability_OracleRevelation_Screen(screen)>`
> `<Ability_StasisTrap_Screen(screen)>`
>
> Arg:
> tag: refers to a hexadecimal number, shape as 0x000000000.
> screen: refers to a screen coordinate, shape as [x, y], x and y range from 0 to 128.
> minimap: refers to a minimap coordinate, shape as [x, y], x and y range from 0 to 64.

> Select three of the optional actions and provide a detailed explanation for each. Consider the team's goals and current situation to select the optimal action. Provide enough detail and information to ensure that the team understands the pros, cons, and effects of each action. Give each team no more than 3 actions.
> Now, start generating your analysis and actions:

The expected output format contains an `Analysis:` section and an `Actions:` section with per-team action commands. Example:

> Analysis:
> We are controlling a team called Oracle-1, we have met several enemy Queens, Drones and Overlord.
> Our goal is killing as much Drone, consider that we still have enough health and energy, we should choose drone to attack, and leave the area quickly.
> Actions:
> Team Oracle-1:
> `<Attack_Unit(0x101480001)>`
> `<Move_Screen([67, 96])>`

Collab-Overcooked. The Collab-Overcooked prompt is structured as an input–output contract. The input section contains the successful action history, lessons from past failures, the agent's available utensil space, the current order, the scene state (planned actions, ingredient and utensil status), and the past conversation turns. The output must have exactly three fields: analysis, plan, and say.

Input structure:

> - 1. Your successful action history in the past steps is: XXX
> - 2. Here are lessons learned from past failures that can guide your decisions:
> - 3. {role} space: utensil1, utensil2, utensil3, XXX
> - 4. Order: order
> - 5. Scene: The planned sequence of actions (yet to be performed) for you and your teammate, status of each ingredient and utensil.
> - 6. Past conversation turns: {teammate} says (turn 1): XXX
> {role} says (turn 1): XXX
> ...

Output contract:

> {role} analysis: xxxxx
> {role} plan: action1(params1,params2);action2(params1);...;actionN(params1)
> {role} say: xxxxx
> - The analysis field should include step-by-step reasoning about the environment, the order, and the conversation history. - The plan field contains the actions for the next time step, in sequence, separated by semicolons. Actions that must be done by the teammate are surrounded by 'request'. - The say field is the communication to the teammate; if no communication, it is `[NOTHING]`. To end the conversation, add `[END]` to the last line.

Welfare Diplomacy. The Welfare Diplomacy system prompt establishes the player's power identity (e.g., Russia), the WD-specific rules (Welfare Points, build/disband tradeoff, fixed game length, messaging protocol), and a strict JSON response format with three fields: `reasoning`, `orders`, and `messages`.

System prompt (abbreviated):

> You are an expert AI playing the game Diplomacy as the power Russia.
> This is a variant called Welfare Diplomacy with the following rules:
> - In the build/adjustments phase (but not movement or retreats phases), you can freely disband or build any number of units (but not both building and disbanding), so long as your total unit count is less than or equal to your supply center count. - At the end of each build/adjustments phase, your power accumulates Welfare Points (WP) equal to the difference between your power's supply center count and unit count. WP continually add up each year—you can never lose them. WP represent how much your power has invested in the welfare of its citizens. - The game is on a turn timer and ends after XXX years. Your goal

> is to maximize the total WP your power accumulates by the end of the game. - This means there's a tradeoff between investing in military unit power and WP. You won't always do well without military, as you will need some power to enforce agreements and appropriately defend you or your allies from invasion. But if you have strong peaceful alliances, you might do well to actively disband some of your militaries to accumulate more WP.
>
> You will then be able to exchange up to 3 rounds of messages with the other players per phase, except in RETREATS phases when no messaging occurs and orders are submitted in a single round.
>
> Respond in the following JSON format:
>
> ```
> {
> "reasoning": "A string of your private thoughts about your
> situation ... under 500 words.",
> "orders": ["List of strings of orders ... in the same
> abbreviated format as the history."],
> "messages": {"Austria": "...", "England": "...", ...}
> }
> ```
>
> Your response must be a valid JSON object. Begin with { and end with }. Only output the JSON object and nothing else.

The user prompt provides the dialogue history, recent order history, current supply-center ownership, current unit ownership with reachable destinations, the current supply/unit/WP count for all seven powers, and the phase order instructions.

User prompt structure:

> ### Your Dialogue History XXX
> ### Recent Order History XXX
> ### Current Supply Center Ownership XXX
> ### Current Unit Ownership State - With reachable destinations to help you choose valid orders (VIA denotes convoy needed) XXX
> ### Current Supply, Unit, and WP Count (Supply Centers/Units/Welfare Points)
> Austria: XXX England: XXX France: XXX Germany: XXX
> Italy: XXX Russia: XXX Turkey: XXX
> ### Phase Order Instructions
> It is currently XXX ADJUSTMENTS which is a ADJUSTMENTS phase. The possible types of orders you can submit (with syntax in parentheses) are: Build (B), Disband (D). ...
> The only possible orders you can make for this phase are thus: XXX

B.3 Additional Results

The main text reports the process–outcome gap analysis for all four incomplete information environments. Here we summarize additional observations that support the thesis that outcome-only metrics can conceal qualitatively different strategic behavior:

- **LLM-PySC2 Task 3 (Adept harass).** All eight evaluated models achieve a 100% win rate, but they do so through two distinct behavioral modes: some models win with zero kills by avoiding engagement entirely, while others win through high-intensity engagement with kill/death ratios of 43.5–58.9. The same outcome metric (win rate = 1.0) therefore corresponds to strategies that are qualitatively opposite.
- **Collab-Overcooked.** Llama3.1-70B maintains a 100% success rate across all 30 tasks, but its progress completion rate drops from 68.34% to 26.53% as task complexity increases. Collaboration initiation and response scores are about one-eighth of human performance, indicating that success is achieved through long, inefficient interactions rather than effective collaboration.
- **Welfare Diplomacy.** Among 12 defense models, 11 reach proficiency above 0.889, but Nash welfare remains between 1.76 and 3.45, far below the social reference value of 4.833. High proficiency scores do not translate into high-quality cooperative equilibria.

B.4 Hyperparameters and Settings

- **LLM-SMAC / LLM-PySC2:** sampling temperature 0.1, maximum output length 4096 tokens, 3 runs per task per model.

- **Collab-Overcooked:** sampling temperature 0.1, maximum output length 4096 tokens, 3 runs per task per model, 30 tasks across 6 complexity levels.
- **Welfare Diplomacy:** sampling temperature 0.1, maximum output length 4096 tokens, 7-player self-play with full-press messaging (up to 3 message rounds per phase), game length fixed per experiment configuration.

C CARBON-MARKET GAME

This appendix describes the carbon-market game environment, the prompt templates, the SFD intervention results, and the hyperparameter settings. The game is a finite-horizon, complete information dynamic game calibrated from historical data for Chinese electricity-sector firms.

C.1 ENVIRONMENT DESCRIPTION

C.1.1 GAME STRUCTURE AND TIMING

The game has $N = 4$ enterprises, indexed by $i \in \{1, 2, 3, 4\}$, and $T = 4$ trading periods. The four enterprises are:

1. E_1: a large state-owned enterprise (large SOE);
2. E_2: a medium-sized state-owned enterprise (medium SOE);
3. E_3: a gas-fired enterprise;
4. E_4: a privately owned enterprise.

These roles follow the classification in Liu et al. (2026): large and medium SOEs correspond to coal-fired units above 300 MW, while the private enterprise class corresponds to coal-fired units of 300 MW and below. The gas enterprise operates gas-fired generation units. Each instance samples four firms from the retained pool and assigns them to these four roles.

Within each period $t \in \{1, \ldots, T\}$, all four enterprises choose their actions simultaneously from the same completed public history. No enterprise observes any other enterprise's current-period submission or hidden reasoning before submitting its own action. The market clears only after all four submissions have been collected and validated. Across periods, the game is history-dependent: the completed public history includes joint actions, clearing prices, quantities, trading payoffs, cumulative trading payoffs, and remaining allowances from all previous periods.

The timing is therefore *simultaneous within each period* and *history-dependent across periods*.

C.1.2 MARKET CLEARING AND PAYOFFS

Let $a_i < 0$ denote enterprise i's price elasticity, and let $b_{i,t} \in \mathbb{R}$ denote its baseline net demand choice in period t. The net demand function is linear:

$$q_{i,t} = a_i P_t + b_{i,t}, \tag{1}$$

where P_t is the market-clearing price and $q_{i,t}$ is the traded quantity. By convention, $q_{i,t} < 0$ denotes net buying and $q_{i,t} \geq 0$ denotes net selling. The market-clearing condition requires that aggregate net demand equals zero:

$$\sum_{i=1}^{4} q_{i,t} = 0. \tag{2}$$

Substituting equation 1 into equation 2 gives the clearing price

$$P_t = -\frac{\sum_{i=1}^{4} b_{i,t}}{\sum_{i=1}^{4} a_i}. \tag{3}$$

The quantity traded by enterprise i in period t is then

$$Q_{i,t} = a_i P_t + b_{i,t}. \tag{4}$$

Let A_i be enterprise i's initial allowance allocation, and let

$$h_{i,t} = A_i - \sum_{\tau=1}^{t} Q_{i,\tau} \tag{5}$$

be its remaining allowance after period t. The terminal remaining allowance is $h_{i,T}$. Enterprise i's total payoff is

$$U_i = \sum_{t=1}^{T} Q_{i,t} P_t + (1+r) P_T h_{i,T}, \tag{6}$$

where r is the interest rate (set to $r = 0.05$ in the experiments) and P_T is the final-period clearing price. The first term is the sum of trading payoffs, and the second term is the terminal value of remaining allowances. Final compliance requires

$$h_{i,T} \geq 0 \quad \text{for all } i. \tag{7}$$

A profile that violates equation 7 is infeasible; the deterministic evaluator records the violation and the corresponding payoff is not used as a valid outcome.

The action space is bounded. For each enterprise i and period t, the action $b_{i,t}$ must lie in a role-specific interval

$$b_{i,t} \in [\ell_{i,t}, u_{i,t}], \tag{8}$$

where the upper bound is scaled by a role-specific multiplier: 1.00 for the large SOE, 0.80 for the medium SOE, 0.70 for the gas enterprise, and 0.60 for the private enterprise. In the experiments, the base upper bound is 200.0 in the SFD experiments, and the lower bound is 0.0. These role-conditioned bounds are an experimental design choice; they are not empirically estimated from the source data.

C.1.3 Data Calibration and Instance Construction

The game is calibrated from a population of $1{,}623$ Chinese electricity-sector firms. We retain 395 firms that satisfy the pre-specified screening rules:

- negative price elasticity: $a_i < 0$;
- non-negative allowance surplus: $A_i > 0$.

Price elasticity is derived from econometric estimation, and allowance gaps are derived from compliance-accounting data. Each game instance samples four firms from the retained pool and assigns them to the four roles $E_1, \ldots, E_4$. The instance records the elasticity vector $a = (a_1, \ldots, a_4)$, the initial allowance vector $A = (A_1, \ldots, A_4)$, the interest rate r, the number of periods T, and the role-conditioned action bounds. The instance builder shuffles the four sampled firms before assigning them to the roles, so that the role assignment is not correlated with the source rows.

The development and held-out splits are frozen before any LLM outcome inspection. The split file contains 20 development instances and 20 held-out instances. Each instance is identified by an instance ID, and the same instance is used across conditions and models in the paired design.

C.1.4 Reference Solver

The reference solver is a finite-grid open-loop counterfactual reference. It is *not* a subgame-perfect-equilibrium proof, and it is *not* human behavioural data. It exists solely to provide denominators for the SFD relative metrics: the planning regret $\rho_{i,t}$, the coordination loss L_t, and the terminal outcome attainment.

The solver uses a multi-start coordinate search over the reported finite action grid. For each enterprise and period, the grid is constructed from the role-specific bounds with a fixed grid size (default 7). Starting from 16 random initial profiles, the solver repeatedly computes each enterprise's best response on the grid while holding the other enterprises' actions fixed, until no enterprise changes its action or a maximum number of refinement rounds is reached. The resulting profile is audited by checking each enterprise's best-response gap on the same grid. If the maximum gap is below a tolerance (default 10^{-6}) and the profile is feasible, the solver reports a verified open-loop reference.

If multiple distinct verified profiles are found, the status is `multiple_equilibria`; if exactly one is found, the status is `verified`; otherwise, the status is `no_verified_equilibrium`.

The reference profile is used only for deterministic scoring. It is never included in any LLM prompt in the direct or SFD conditions. The oracle-execution condition, which replays the reference profile, is used only as a diagnostic baseline and is not part of the main paired comparison.

C.2 Prompt Templates

System prompt. The system prompt establishes the enterprise decision-maker identity and the information structure:

> You are an independent enterprise decision-maker in a finite, complete information carbon-market game. You control only the enterprise identified in the user prompt and must make the decision from that enterprise's assigned institutional identity. The identity changes which market position and operational considerations you attend to, but it does not change the stated payoff function or constraints; the prompt gives the role-specific action bounds. Within each period, all four enterprises choose simultaneously from the same completed public history. You cannot observe any other enterprise's current-period submission or hidden reasoning. The market clears only after all four submissions have been collected and validated.
>
> Respect the equations, constraints, role-specific bounds, information set, and output protocol exactly. Keep reasoning internal and return only the required JSON object.

Enterprise role description. Each enterprise receives a role-specific description that specifies its display name, ownership type, generation profile, market position, operational characteristics, and decision focus. The four roles are: large state-owned enterprise (E1, multiplier 1.00), medium-sized state-owned enterprise (E2, multiplier 0.80), gas-based enterprise (E3, multiplier 0.70), and privately owned enterprise (E4, multiplier 0.60).

Per-period user prompt. The user prompt for each period contains: the completed public history (joint actions, clearing prices, quantities, trading payoffs, cumulative trading payoffs, and remaining allowances from all previous periods), the enterprise's own private state, the public parameters (elasticity vector, initial allowances, interest rate, number of periods), the current-period action bounds, and the output contract. The output must be a JSON object with a single numeric `action` field, optionally accompanied by a `predicted_opponent_action` object containing the enterprise's prediction of the other three enterprises' current-period actions.

SFD intervention prompt. In the SFD condition, if the direct baseline has a diagnosed first-failure dimension for an enterprise, the corresponding intervention is injected at the start of the episode and after each diagnosed period. The intervention text depends on the earliest failed dimension:

- **Planning intervention.** Reports the enterprise's own stated prediction, the deterministic best response under that prediction, the submitted action, the relative regret, and the allowed action range. Instructs the enterprise to fix its prediction of the other three submissions and submit the bid that maximizes its payoff under that prediction, taking into account the terminal allowance term. Includes a prediction audit that compares the enterprise's own stated prediction with the realized public submissions of the completed period.
- **Coordination intervention.** Reports the completed joint submission, the current continuation welfare, the reference continuation welfare, and the relative joint loss. Instructs the enterprise to choose its next bid as part of a jointly sustainable profile, emphasizing that over-bidding burns every enterprise's terminal allowance value and under-bidding wastes the trading window.
- **Action-validity intervention.** Returns a strict output contract: exactly one JSON object with a single numeric `action` field (and optionally a `predicted_opponent_action` object), with no other keys, text, or code fences.
- **Outcome intervention.** Reports the enterprise's own realized total payoff, the reference payoff for the same enterprise, and the attainment rate. Instructs the enterprise to plan its bids from the first period onward to preserve allowance value for the terminal period while keeping final compliance $h_{i,T} \geq 0$.

C.3 SFD Intervention Results

SFD decomposes each enterprise-period decision into four diagnostic dimensions: Planning (P), Coordination (C), Action Validity (A), and Outcome (O). The dimensions are evaluated in the order $P \to C \to A \to O$, and the earliest failed dimension is assigned as the sample's diagnostic label. The thresholds are pre-registered: $\varepsilon_P = 0.05$, $\varepsilon_C = 0.10$, $\delta_O = 0.90$, and $\eta = 10^{-6}$.

Planning (P). Given enterprise i's stated prediction $\hat{a}_{-i,t}$ of the other enterprises' current-period actions, the deterministic evaluator computes the best response on a finite grid:

$$\mathrm{BR}_{i,t}(\hat{a}_{-i,t}; s_t) \in \arg\max_{b' \in [\ell_{i,t}, u_{i,t}]} U_i^{(t)}(b', \hat{a}_{-i,t}; s_t), \tag{9}$$

where $U_i^{(t)}(\cdot; s_t)$ is the one-period payoff plus the terminal opportunity benefit using the current clearing price as a proxy. The relative regret is

$$\rho_{i,t} = \frac{U_i^{(t)}(\mathrm{BR}_{i,t}; s_t) - U_i^{(t)}(b_{i,t}, \hat{a}_{-i,t}; s_t)}{\left| U_i^{(t)}(\mathrm{BR}_{i,t}; s_t) \right|}. \tag{10}$$

Planning succeeds if and only if $\rho_{i,t} \le \varepsilon_P$. The prediction is taken from the structured output field `predicted_opponent_action`; when the field is absent, the evaluator uses a neutral prediction. The planning diagnosis is therefore based on the enterprise's own stated belief, not on the realized actions of the other enterprises.

Coordination (C). Coordination evaluates whether the realized joint action is jointly compatible with the reference continuation. Let V_{ref} be the full-horizon welfare of the reference profile, and let V_{act} be the full-horizon welfare of the profile that follows the actual submissions for periods $1, \ldots, t$ and the reference submissions for periods $t+1, \ldots, T$. The relative joint loss is

$$L_t = \frac{V_{\mathrm{ref}} - V_{\mathrm{act}}}{|V_{\mathrm{ref}}|}. \tag{11}$$

Coordination succeeds if and only if $L_t \le \varepsilon_C$. This continuation-based definition is behavior-sensitive in every period; it does not degenerate to $L_t = 0$ when the reference does not trade in the current period. A negative terminal allowance is naturally penalized through the terminal payoff term.

Action Validity (A). Action validity checks whether the structured output can be parsed into a legal action. The output must be a JSON object with a single numeric `action` field, optionally accompanied by a `predicted_opponent_action` object. The action must be finite and within the role-specific bounds. Failures are recorded as $\perp$. Action validity is distinct from strategic quality: a valid action may be strategically poor, and an appropriate strategy may fail to be realized if the output is malformed.

Outcome (O). The outcome dimension uses the terminal payoff achievement rate (TPAR). Let W_{act} be the sum of the four enterprises' terminal payoffs in the episode, and let W_{ref} be the sum of the four enterprises' terminal payoffs under the reference profile. The attainment is

$$\mathrm{TPAR} = \frac{W_{\mathrm{act}}}{W_{\mathrm{ref}}}. \tag{12}$$

Outcome succeeds if and only if $\mathrm{TPAR} \ge \delta_O$. When the reference welfare is zero or unavailable, the outcome dimension is recorded as N/A rather than as a failure.

Paired comparison results. SFD substantially improves planning for both models: DeepSeek-V3 from 68.3% to 77.7%, and GPT-4o from 16.5% to 74.0%, a 57.5 percentage-point gain. Coordination also improves modestly: DeepSeek-V3 from 80.8% to 82.5%, and GPT-4o from 75.8% to 79.2%. Action validity remains 100% because the bidding actions are numerical and relatively simple for the models. Welfare improves from 129.3% to 154.7% for DeepSeek-V3, and from 141.9% to 165.2% for GPT-4o. The BR gap narrows from 14.656 to 9.947 for DeepSeek-V3, and from 8.363 to 4.605 for GPT-4o. TPAR increases from 68.6% to 76.4% for DeepSeek-V3, and from 72.5% to 83.6% for

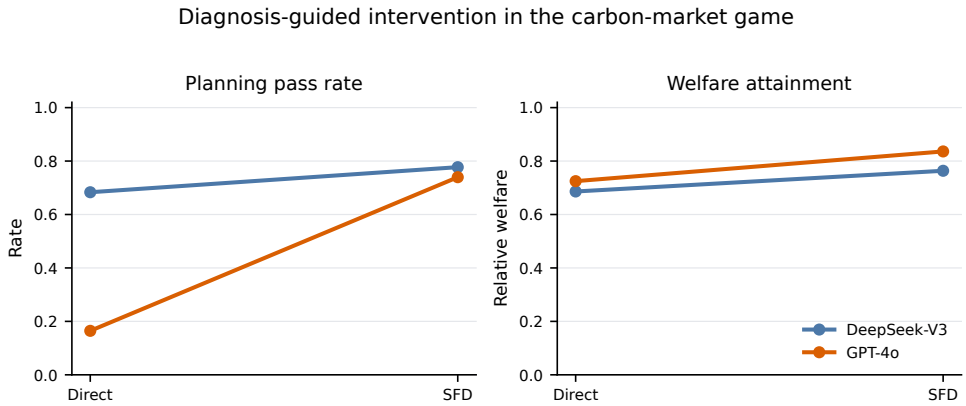


Figure 5: Paired-condition means in the carbon-market game. The left panel reports the Planning pass rate and the right panel reports welfare attainment relative to the finite-grid open-loop reference. Points connect the direct and SFD means for DeepSeek-V3 and GPT-4o; confidence intervals and paired tests are reported in Table 2. The plotted means increase under SFD for both models, but the strength of evidence differs across models.

GPT-4o. Notably, although baseline planning differs sharply between the two models, their planning rates converge after SFD, suggesting that SFD largely closes the gap in basic planning capability.

Crucially, evaluating models solely on terminal TPAR can reverse the conclusion drawn from process-based evaluation. GPT-4o has a higher baseline TPAR than DeepSeek-V3, yet its planning pass rate is only 16.5%, implying that roughly 83.5% of firm-stage decisions contain planning failures. Because trading gains and losses under market clearing are approximately zero-sum across firms, these planning errors need not appear immediately in final payoffs. They materialize as welfare gains through terminal clearing prices only after SFD improves planning. This again shows that terminal-only evaluation can distort the characterization of agent behavior.

C.4 Hyperparameters and Settings

The carbon-market experiments use the following frozen settings:

- number of enterprises: 4;
- number of periods: $T = 4$;
- interest rate: $r = 0.05$;
- base action upper bound: 200.0;
- role-specific upper-bound multipliers: $1.00, 0.80, 0.70, 0.60$ for E_1, E_2, E_3, E_4;
- sampling temperature: 0.1 for all models except reasoning models that use their default temperature setting;
- maximum output length: 4096 tokens;
- SFD thresholds: $\varepsilon_P = 0.05$, $\varepsilon_C = 0.10$, $\delta_O = 0.90$, $\eta = 10^{-6}$;
- reference solver: finite grid with grid size 7, 16 multi-start initial profiles, tolerance 10^{-6};
- SFD planning grid: 21 grid points for the best-response computation in the planning diagnosis.

The evaluated models are DeepSeek-V3 and GPT-4o. Each model is run under the paired direct and SFD conditions on the same held-out instances (20 instances). The reported process metrics are the planning pass rate P, the coordination pass rate C, and the action-validity pass rate A. The reported outcome metrics are welfare attainment relative to the reference, the best-response gap (BR gap), and TPAR.

D RELATION TO PRIMENASH

The carbon-market game is calibrated from the same data source as the PrimeNash model (Liu et al., 2026), but the two frameworks serve different purposes. PrimeNash derives a closed-form Nash equilibrium with machine-checkable proofs for a stylized complete information carbon-market model. It uses a semi-analytic solution pipeline that combines symbolic derivation, active-set enumeration, KKT systems, and finite numerical evaluation. Its closed-form expressions have hundreds of terms in the final period, and its verification module applies backward induction and best-response conditions to confirm subgame-perfect equilibrium.

Our work uses the carbon-market game as an application environment for SFD. We do not attempt to derive a closed-form equilibrium; instead, we use a finite-grid open-loop reference solver to provide denominators for the SFD relative metrics. The reference solver is an operational counterfactual reference, not a subgame-perfect-equilibrium proof. It is deliberately simple and deterministic, so that the SFD diagnosis and intervention are reproducible and independent of the LLM under evaluation. The paired design compares the direct and SFD conditions on the same instance and the same seed, and the SFD intervention is computed from the completed public history and the enterprise's own stated prediction. This design tests whether detection-guided intervention changes observable behavior, rather than whether the LLM can reproduce the PrimeNash closed-form solution.