Automated Human Strategic Behavior Modeling via Large Language Models (Supplemental Material)

Contents

A LLM-Generated Contents for Discovered Models

In Section 5 of the main text, we present mathematical formulations of models discovered by our framework **AutoBM**. In this section, we further demonstrate the contents directly generated by the LLM as both a complement and an illustration of our representation $\mathcal{M}$.

A.1 Ultimatum Game

We adopted a JSON-style format to implement the representation $\mathcal{M}$, which directly corresponds to the definition in the Section 4 of the main text.

```
policy = {
   "proposer_strategy": "Pr(a|o) propto exp(utility(a, o))",
   "parameters": [
      {"name": "self_interest_weight", "description": "Scaling factor for personal
          gain preference"},
      {"name": "responder_pressure_base", "description": "Base magnitude of
          responder acceptance anticipation"},
      {"name": "responder_sensitivity", "description": "Steepness of responder
          acceptance curve"},
      {"name": "fairness_weight", "description": "Strength of fairness
          consideration"},
      {"name": "fairness_decay", "description": "Rate of fairness preference
          attenuation"},
      {"name": "resource_sensitivity", "description": "Exponent for
          resource-dependent fairness scaling"}
   ],
   "constants": [
      {"name": "split_point", "description": "Equitable division reference",
          "type": "float", "value": 0.5}
```

```
  ],
  "intermediate_terms": [
     {
        "name": "self_interest_component",
        "description": "Proposer’s personal gain incentive",
        "computation": "self_interest_weight * (1 - a) * o"
     },
     {
        "name": "responder_acceptance_component",
        "description": "Anticipated responder acceptance likelihood",
        "computation": "responder_pressure_base / (1 +
            exp(-responder_sensitivity*(a - split_point)))"
     },
     {
        "name": "fairness_influence_component",
        "description": "Context-dependent equity consideration",
        "computation": "fairness_weight * (1 - |a - split_point|**fairness_decay) *
            (ln(1 + o))**resource_sensitivity"
     },
     {
        "name": "utility",
        "description": "Total decision utility combining all factors",
        "computation": "self_interest_component + responder_acceptance_component +
            fairness_influence_component"
     }
  ]
}
```

For consistency with our expressions in Sections 3-4, we adopted distinct symbols to reformulate the aforementioned content into mathematical formulations.

A.2 Repeated Rock-Paper-Scissors Game

In the original representation, we permit the LLM to generate pseudocode for illustrating computational processes for flexibility.

```
policy = {
    "human_strategy": "Pr(a_t | history) propto exp(utility)",
    "parameters": [
        {"name": "recency_weight", "description": "Scaling factor for action
            recency preference"},
        {"name": "discount_factor", "description": "Exponential decay rate for
            historical actions"},
        {"name": "repetition_penalty", "description": "Strength of penalty for
            repeated actions"},
        {"name": "outcome_influence", "description": "Weight of outcome-based
            action preference"}
    ],
    "constants": [
        {"name": "memory_window", "description": "Lookback period for recent
            actions", "type": "int", "value": 0}
    ],
    "intermediate_terms": [
        {
            "name": "action_recency_score",
            "description": "Exponentially weighted recent action frequency",
            "computation": "recency_weight * sum(discount_factor**(current_step -
                tau - 1) for tau in [0, s-1] if a_tau == a_s)"
        },
        {
            "name": "repetition_penalty_term",
            "description": "Penalty for frequent action repetition",
            "computation": "(repetition_penalty / memory_window) * count(a_tau ==
                a_s for tau in [s-memory_window, s-1])"
        },
        {
            "name": "outcome_association_score",
            "description": "Winning outcome-based action preference",
            "computation": "(outcome_influence / memory_window) * sum(1 for tau in
                [s-memory_window, s-1] if a_tau == a_s and v_tau == 1)"
        },
        {
            "name": "utility",
            "description": "Combined decision utility",
            "computation": "action_recency_score - repetition_penalty_term +
                outcome_association_score"
        }
    ]
}
```

A.3 Continuous Double Auctions

```
policy = {
    "buyer_strategy": "Pr(a | H, Q, A, P, t) propto exp(utility)",
    "parameters": [
        {"name": "profit_weight", "description": "Scaling factor for profit-seeking
            tendency"},
        {"name": "anchoring_strength", "description": "Influence of reference price
            alignment"},
        {"name": "history_alignment_factor", "description": "Weight for bid history
            consistency"},
        {"name": "liquidity_matching_weight", "description": "Strength of order
            book matching incentive"},
        {"name": "spread_sensitivity", "description": "Modulation effect from
            bid-ask spread"},
        {"name": "urgency_acceleration", "description": "Time-dependent bidding
            aggressiveness"}
    ],
    "constants": [
        {"name": "price_memory_window", "description": "Lookback period for price
            history", "type": "int", "value": 3},
        {"name": "volatility_weight", "description": "Fixed weight in history
            alignment kernel", "type": "float", "value": 0.1},
        {"name": "base_urgency_exponent", "description": "Minimum time sensitivity
            coefficient", "type": "float", "value": 1.3}
    ],
    "intermediate_terms": [
        {
            "name": "market_volatility",
            "description": "Price fluctuation measure from recent transactions",
            "computation": "std(P[-price_memory_window:]) /
                mean(P[-price_memory_window:])"
        },
        {
            "name": "market_liquidity",
            "description": "Normalized active ask order count",
            "computation": "min(1, len(A) / 10)"
        },
        {
            "name": "effective_spread",
            "description": "Bid-ask spread magnitude",
            "computation": "abs(min(e.price for e in A) - max(b.price for b in Q if
                not b.from_current))"
        },
        {
            "name": "reference_price",
            "description": "Weighted market price benchmark",
            "computation": "0.7 * (P[-3] + 1.5*P[-2] + 2*P[-1])/(1+1.5+2) + 0.3 *
                min(e.price for e in A)"
        },
        {
            "name": "profit_component",
            "description": "Profit-seeking incentive",
            "computation": "profit_weight * (1 - a)"
        },
        {
            "name": "price_anchoring_component",
            "description": "Alignment with reference price",
            "computation": "anchoring_strength * exp(-abs(a - reference_price))"
        },
        {
            "name": "history_alignment_component",
            "description": "Consistency with recent bidding patterns",
            "computation": "history_alignment_factor * sum(exp(-abs(a -
                h.price)/(volatility_weight + volatility_weight*market_volatility))
                for h in H[-price_memory_window:])"
        },
        {
            "name": "liquidity_matching_component",
            "description": "Order book compatibility score",
            "computation": "liquidity_matching_weight * ((count(e in A where e.price
                <= a)/len(A)) - (count(b in Q where b.price > a and not
                b.from_current)/len(Q))) * exp(-spread_sensitivity *
                effective_spread) * (1 + market_liquidity)"
        },
        {
            "name": "urgency_component",
            "description": "Time-dependent bid acceleration",
            "computation": "urgency_acceleration * a * (current_time /
                150)**(base_urgency_exponent - 0.3*market_liquidity)"
        },
        {
            "name": "utility",
            "description": "Total decision utility",
            "computation": "profit_component + price_anchoring_component +
                history_alignment_component + liquidity_matching_component +
                urgency_component"
        }
    ]
}
```

B Baseline Details

This section unpacks the definitions and design rationale of human-crafted baselines that are excised from Section 5 due to space constrains.

B.1 Ultimatum Game

Quantal Response Equilibrium (QRE) This model posits that participants exhibit bounded rationality, selecting actions stochastically with probabilities contingent upon their expected utilities. The model's formal construction is articulated as following Equations (1)-(2):

- The responder's probability of accepting an offer is modeled as:
$$P(\text{accept}|a, o) \propto \theta_2 \cdot oa. \tag{1}$$
Here, oa denotes the responder's payoff upon accepting offer a.
- The proposer's expected utility, $u_{\boldsymbol{\theta}}(a, o)$, for offering a share a, is defined as their reservation utility, $o(1-a)$, multiplied by the responder's acceptance probability:
$$u_{\boldsymbol{\theta}}(a, o) = o(1-a) \cdot P(\text{accept}|a, o). \tag{2}$$

- The probability, $\pi_{\boldsymbol{\theta}}(a|o)$, that the proposer selects a particular share a, is proportional to the exponentiated expected utility associated with that action:

$$\pi_{\boldsymbol{\theta}}(a|o) \propto \exp(\theta_1 \cdot u_{\boldsymbol{\theta}}(a,o)). \tag{3}$$

The model aims to encapsulate the stochasticity or "noise", indicative of bounded rationality.

Equity, Reciprocity, and Competition (ERC) This model posits that proposers, while maximizing self-interest, also consider the fairness of the allocation. Their utility function $u_{\boldsymbol{\theta}}(a,o)$ is defined as Equation (4):

$$u_{\boldsymbol{\theta}}(a,o) = \theta_1 o(1-a) - \theta_2 \left(a - \frac{1}{2}\right)^2. \tag{4}$$

The components are interpreted as follows:

- The first term, $\theta_1 o(1-a)$, represents the proposer's self-interest motivation: a smaller proposed share a increases the proposer's retained portion $o(1-a)$ and thus this utility component. θ_1 is the weight or sensitivity to self-interest.
- The second term, $-\theta_2(a-\frac{1}{2})^2$, signifies an aversion to deviations from an equal split: this term becomes negative when the proposed share a deviates from the equal division point ($a = 0.5$), and the negative utility increases with the magnitude of the deviation. θ_2 represents the weight of fairness considerations or the degree of aversion to inequity.

Inequity Aversion (IA) This model, similar to **ERC** in its focus on fairness, distinguishes between types of inequity—disadvantageous and advantageous—assigning different aversion weights. The utility function $u_{\boldsymbol{\theta}}(a,o)$ is defined as Equation(5):

$$u_{\boldsymbol{\theta}}(a,o) = \theta_1 o(1-a) - \theta_2 \max(o(2a-1),0) - \theta_3 \max(o(1-2a),0). \tag{5}$$

The components are interpreted as follows:

- The first term, $\theta_1 o(1-a)$, again represents the proposer's self-interest motivation.
- The second term, $-\theta_2 \max(o(2a-1),0)$, captures aversion to disadvantageous inequity (i.e., when the proposer receives less). The extent of this aversion is measured by θ_2 (≥ 0).
- The third term, $-\theta_3 \max(o(1-2a),0)$, denotes aversion to advantageous inequity (i.e., when the proposer receives more). The discomfort from this type of inequity is measured by θ_3 (≥ 0).

B.2 Repeated Rock-Paper-Scissors Game

Subgame Perfect Nash Equilibrium (SPNE) In this equilibrium, players select actions uniformly at random. The choice probability for any action $a_s \in \{\text{Rock, Paper, Scissors}\}$ is given by Equation (6):

$$\pi_{\boldsymbol{\theta}}(a_s|\boldsymbol{o}_s) = \frac{1}{3}, \quad \forall a_s \in \{\text{Rock, Paper, Scissors}\}. \tag{6}$$

Under this strategy, each of the three actions is selected with an equal and fixed probability of $1/3$ in each round, representing a theoretical equilibrium in the repeated Rock-Paper-Scissors stage game.

Regret Matching (RM) Players dynamically adjust action probabilities based on cumulative regret. Let $R_s(a)$ denote the cumulative regret for action a at round s, calculated from the historical information $\boldsymbol{o}_s$ (which includes the actions and payoffs of all players up to round $s-1$). The probability of selecting action a_s from the action set A is given by Equation (7):

$$\pi_{\boldsymbol{\theta}}(a_s|\boldsymbol{o}_s) = \begin{cases} \frac{\max(0,R_s(a_s))}{\sum_{a' \in A}\max(0,R_s(a'))}, & \text{if } \sum_{a' \in A}\max(0,R_s(a')) > 0; \\ \frac{1}{3}, & \text{otherwise.} \end{cases} \tag{7}$$

This strategy dictates that the probability of choosing an action a_s is proportional to its positive cumulative regret $R_s(a_s)$, which quantifies the potential payoff foregone by not selecting that action in previous rounds. If the sum of positive regrets is zero, actions are selected with a uniform probability (here, $1/3$, implying a three-action set).

Fictitious Play (FP) In Fictitious Play, players form beliefs about an opponent's actions based on their historical frequencies and then select actions to maximize the expected utility against these beliefs. Let $N_{s-1}(a_{\text{opp}})$ be the count of an opponent's action $a_{\text{opp}} \in A$ (where A is the action set) observed up to round $s-1$, derived from the history $\boldsymbol{o}_s$. The player's belief about the opponent's current action is $b_s(a_{\text{opp}}) = N_{s-1}(a_{\text{opp}})/(s-1)$ (for $s > 1$). The expected utility for the player choosing action a_s, given the history $\boldsymbol{o}_s$, is denoted as $U_{\text{exp}}(a_s|\boldsymbol{o}_s) = \sum_{a_{\text{opp}} \in A} b_s(a_{\text{opp}}) u(a_s, a_{\text{opp}})$, where $u(a_s, a_{\text{opp}})$ is the payoff to the player when they choose a_s and the opponent chooses a_{opp}. The action selection probability is then given by Equation (8):

$$\pi_{\boldsymbol{\theta}}(a_s|\boldsymbol{o}_s) \propto \exp(\theta \cdot U_{\text{exp}}(a_s|\boldsymbol{o}_s)). \tag{8}$$

The parameter θ serves as a "temperature" that controls the sensitivity of action selection to the calculated expected utilities, influencing how deterministically the best response is chosen.

Win-Stay, Lose-Shift (WSLS) The Win-Stay, Lose-Shift strategy involves repeating an action if it was successful in the previous round and switching to a different action if it was not. Let a_{s-1} be the player's action in the previous round and v_{s-1} its outcome ($v_{s-1} = 1$ indicating a win, $v_{s-1} \neq 1$ for a non-win), both derived from the history $\boldsymbol{o}_s$. The probabilistic policy $\pi_{\boldsymbol{\theta}}(a_s|\boldsymbol{o}_s)$ for choosing the current action a_s is defined as:

$$\pi_{\boldsymbol{\theta}}(a_s|\boldsymbol{o}_s) = \begin{cases} p_{\text{stay}}, & \text{if } v_{s-1} = 1 \text{ and } a_s = a_{s-1}; \\ \frac{1}{2}(1 - p_{\text{stay}}), & \text{if } v_{s-1} = 1 \text{ and } a_s \neq a_{s-1}; \\ 1 - p_{\text{shift}}, & \text{if } v_{s-1} \neq 1 \text{ and } a_s = a_{s-1}; \\ \frac{1}{2} p_{\text{shift}}, & \text{if } v_{s-1} \neq 1 \text{ and } a_s \neq a_{s-1}. \end{cases} \tag{9}$$

This is a simple heuristic where the decision primarily depends on the most recent action and its outcome. The parameters p_{stay} and p_{shift} (components of $\boldsymbol{\theta}$) dictate the probabilities of repeating a_{s-1} or switching to a specific alternative action $a_s \neq a_{s-1}$, contingent on the outcome v_{s-1}.

B.3 Continuous Double Auctions

Belief-Based Bidding (BB) In the Belief-Based Bidding model, buyers aim to maximize expected utility using beliefs about bid acceptance probabilities, derived from the order book state $\boldsymbol{o}_\tau$. The belief $q(a_\tau|\boldsymbol{o}_\tau)$ that a buyer's bid a_τ at time τ will be accepted is estimated proportionally by Equation (10):

$$q(a_\tau|\boldsymbol{o}_\tau) \propto \frac{\sum_{p^i \in \boldsymbol{p}_\tau} \mathbf{1}_{\{p^i < a_\tau\}} + \sum_{a^i \in \mathcal{E}_\tau} \mathbf{1}_{\{a^i < a_\tau\}}}{\sum_{p^i \in \boldsymbol{p}_\tau} \mathbf{1}_{\{p^i < a_\tau\}} + \sum_{a^i \in \mathcal{E}_\tau} \mathbf{1}_{\{a^i < a_\tau\}} + \sum_{b^i \in \mathcal{B}_\tau} \mathbf{1}_{\{(b^i > a_\tau) \wedge (b^i \notin \boldsymbol{p}_\tau)\}}}. \tag{10}$$

The numerator sums counts of existing orders (e.g., asks from sets $\boldsymbol{p}_\tau$ and $\mathcal{E}_\tau$) priced below the bid a_τ. The denominator augments this sum with counts of certain competing orders (from set $\mathcal{B}_\tau$, priced above a_τ and not in $\boldsymbol{p}_\tau$). The expected utility of submitting bid a_τ is $U(a_\tau|\boldsymbol{o}_\tau) = q(a_\tau|\boldsymbol{o}_\tau) \cdot (1 - a_\tau)$. The bidding policy $\pi(a_\tau|\boldsymbol{o}_\tau)$ for selecting a_τ is then modeled as proportional to its exponentiated expected utility (Equation (11)):

$$\pi(a_\tau|\boldsymbol{o}_\tau) \propto \exp(\theta \cdot U(a_\tau|\boldsymbol{o}_\tau)), \tag{11}$$

where θ is a parameter controlling sensitivity to expected utility differences (rationality/temperature).

Time-Dependent Bidding (TDB) Time-Dependent Bidding is a strategy where a participant's bid, a_τ, is adjusted according to the progression of time τ. A common heuristic involves increasing the bid as the deadline or the conclusion of a round approaches.

$$a_\tau = \text{sigmoid}(\alpha + \beta \cdot e^{\gamma\tau}). \tag{12}$$

In Equation (12), α, β, and γ are parameters that shape the bidding trajectory over time.

C Prompt

In this section, we present the prompts employed in AutoBM. These prompts integrate stage-specific templates with task-specific contents.

C.1 Templates

We distinguish component origins by employing fixed-width typography (`{components}`) for predefined elements and blue coloration ({components}) for framework-generated content during runtime.

Representation Template We utilize the following template to guide LLM to generate the model representation that matches our definition.

Representation Template (referred as `{repr_format}`)

```
policy = {
   "{role}_strategy": "Pr(a|{information_structure}) = ...", # Mathematical expression or
        pseudocode
   "parameters": [
      {"name": "param1", "description": ""},
      # ...other parameters
   ],
   "constants": [
      {"name": "constant1", "description": "", type: "int", value: 0},
      # ...other constants
   ],
   "intermediate_terms": [
      {
         "name": "term1",
         "description": "",
         "computation": , #Mathematical expression or pseudocode
      },
      # ...other intermediate terms
   ]
}
```

- **Parameters:** Prespecified quantities that define a model's structural assumptions and requiring explicit learning from data using gradient descending.
- **Constants:** SIMPLE fixed values that are NOT subject to learning or optimization, serving as reference points or prior assumptions or hyperparameters. If not neccessary, do not make float value as constants, leave them in parameters.
- **Intermediate terms:** Dynamically generated transient variables derived during computational processes, which serve as critical feature design elements for capturing system behaviors.

Code Template We present the template architecture for PyTorch module generation that enables automation of model training, validation, and testing pipelines.

Code Generation Template

```
# [STRICT IMPLEMENTATION BEGIN]
import torch
import torch.nn as nn

class GameModel(nn.Module):
    def __init__(self, {extra_settings}):
        super().__init__()
        # Initialization of trainable parameters
        # self.parameter1 = nn.Parameter(torch.tensor(0.5))

        # Register constants
        # self.register_buffer("constant1", torch.tensor(0.5))

    def forward(self, {inputs}):
        """
        Compute the action probability distribution given the {inputs}.

        Args:
            {inputs} (torch.Tensor):
```

```
            Information structure/Dataset structure description.
            Shape: [batch_size, {input_size}]

        Returns:
            torch.Tensor:
            Probability distribution over possible actions.
            Each row sums to 1 after normalization.
            Shape: [batch_size, n_actions]
        """
    # other helper functions can be defined here

# [STRICT IMPLEMENTATION END]
```

Judgment Template We use this template to prompt LLM to generate judgment for each model.

Judgment Template

{scene_desc}
Here is a probabilistic policy model for human players.
{Model representation}
The model has tuned the parameter on the dataset of different states and human players' actions. The following are the validation results of the model on the validation set.
Validation Outcome
{Validation results}
Analysis Task

- Focus on term-level and structure-level analysis of the model. The value of parameters cannot be manually changed, just view it as references.
- **Categorize loss distribution patterns**: Identify game scenarios and participant behaviors where prediction loss is systematically higher vs. lower. Consider structural factors.
- **Analyze predictive power of model terms**: Evaluate how specific terms in the strategy function correlate with accuracy. Distinguish between terms that consistently reduce loss vs. those introducing noise or misalignment with game incentives.
- **Synthesize structural interactions**: Propose hypotheses explaining why certain term architectures succeed/fail in particular information environments or strategic landscapes.
- **Evaluation and Suggestions**: Based on the above analysis, provide a comprehensive evaluation of the current model and offer concrete suggestions to enhance its performance.

Output Requirements

1. Provide a clear, step-by-step analysis.
2. Offer concrete, actionable improvement suggestions—no high-level platitudes or vague statements.
3. End with a "Conclusion" section of no more than five concise sentences, each stating a top insight or recommendation.
4. Use this Conclusion template:
   Conclusion:
   - {Most critical analysis finding/suggestion 1}
   - {Most critical analysis finding/suggestion 2}
   - ...
   - {Most critical analysis finding/suggestion 5}

Initialization Template Here we present the prompt template used in the Initialization stage.

Templates for Initialization Stage

Strategic Environment Description
`{scene_desc}`
Task Specification
`{task_desc}`
Output Format Requirements
`{repr_format}`

The placeholders used in this prompt are defined as follows:

- `{scene_desc}`: A natural-language description of the strategic environment specifying how participants interact and their information structures.
- `{task_desc}`: An instruction specifying the target role for strategy analysis.
- `{repr_format}`: The representation template $\mathcal{M}$ with some extra explanations and constrains. Presented in Paragraph C.1

The first two are task-specific and will be addressed in subsequent subsections.

Improvement templates Here we present the prompt template used in the Improvement strategy of the Expansion stage.

Templates for Improvement Strategy

Strategic Environment Description
`{scene_desc}`
Here are some models aiming to predict the behavior in the game.
{models}
Now analyze these models . Your task is to develop a better-performing model by utilizing the insights from these models.
Guidelines:

- You may utilize your knowledge of the game theory and strategic behavior principles.
- Analyze each model and the judgment of the model.
- Considering the models' structures and the terms used, which ones are the most powerful, which ones are the weakest?
- If the term is powerful, how can it be further improved? If the term is weak, how can it be removed or replaced?
- Are there other aspects that previous models didn't consider? What new mechanisms/structures/features can be introduced to improve the model?
- How to more effectively combine the terms? How to more effectively exploit the information structure.
- Keep the model clean and interpretable.
- Follow the guidelines and think step by step

Output format (possible chain-of-thought and reasoning steps)
`{repr_format}`

{models} contains multiple model samples, which are automatically acquired and assembled during the **AutoBM**'s operation. Each sample consists a representation, a validation result and a judgment.

Model Sample Template

Model Description:
{The representation of the model}
Validation Result:
{The validation result}
Judgments and suggestions:
{Judgments generated by LLM}

Exploration templates Here we present the prompt template used in the Exploration strategy of the Expansion stage.

Templates for Exploration Strategy

Strategic Environment Description
`{scene_desc}`
Here are some models aiming to predict the behavior in the game.
{models}
Now analyze these models. Your task is to generate a different model in terms of approach to explore other possibilities of solving the problem.
Guidlines:

- You may utilize your knowledge of the game theory and strategic behavior principles.
- Check each model and the parameters/constants/intermediate terms of the model.
- What part of the information provided is not used/well exploit in the model?
- Are there other completely different aspects that previous models didn't consider?
- What new mechanisms/structures/features can be
- Keep the model clean and interpretable.
- Follow the guidelines and think step by step.

Output format (possible chain-of-thought and reasoning steps)
`{repr_format}`

C.2 Task-Specific Contents

Strategic Environment Description Referred to as `{scene_desc}`, this is a natural-language definition of the strategic interactive environment, detailing participants' interaction mechanisms and the information observed by different agents.

Ultimatum Game

The ultimatum game is a sequential economic experiment where:

1. A total amount o is given to a proposer.
2. The proposer makes an offer $a \in [0, 1]$ to a responder.
3. The responder can either accept or reject the offer.
4. If the responder accepts, both players receive their respective shares ($(1 - a) \cdot o$ for proposer and $a \cdot o$ for responder);
5. if rejected, both players receive nothing.

Repeated Rock-Paper-Scissors Game

The repeated RPS game is an iterative strategic interaction where:

1. Two players engage in multiple rounds (50) of the classic Rock-Paper-Scissors game.
2. In each round, players simultaneously choose an action $a \in \{\text{Rock, Paper, Scissors}\}$.
3. The outcome is determined by standard RPS rules: Rock beats Scissors, Scissors beats Paper, Paper beats Rock.
4. Players receive payoffs based on the outcome (typically $+1$ for win, 0 for tie, -1 for loss).
5. Both players observe the full history of previous actions before each subsequent round.
6. The game continues for the predetermined number of rounds (50), with cumulative payoffs tracked throughout.

Continuous Double Auctions

i. Core Definitions

1. **Participants**
   - **Buyers**: Let $\mathcal{B} = \{b_1, \ldots, b_{10}\}$ denote the set of buyers, each with a private reservation price β_i.
   - **Sellers**: Let $\mathcal{S} = \{s_1, \ldots, s_{10}\}$ denote the set of sellers, each with a private reservation price σ_j.

ii. Auction Process

1. **Quote Submission**
   - In each round $r \in \{1, \ldots, 10\}$, participants submit quotes during time $t \in [0, T]$ seconds, where $T = 150$.
   - **Buyer quotes**: $q_i^{r,k}(t_{i,k}) \in [0, \beta_i]$, where $q_i^{r,k}$ is the k-th bid by buyer b_i at time $t_{i,k}$ in round r. (Price is in monetary units; bids may be zero.)
   - **Seller quotes**: $e_j^{r,\ell}(t_{j,\ell}) \in [\sigma_j, M]$, where $e_j^{r,\ell}$ is the ℓ-th ask by seller s_j at time $t_{j,\ell}$ in round r, and M is a sufficiently large upper bound (e.g., $M = 99999$).
   - Each quote remains active for 10 seconds after submission; participants may update (overwrite) their quotes at any time before expiration.
2. **Order Book Dynamics**
   - **Active quotes** at time t in round r:
     - Buyer bids: $\mathcal{B}^r(t) = \{q_i^{r,k} \mid t_{i,k} \in [t-10, t]\}$.
     - Seller asks: $\mathcal{E}^r(t) = \{e_j^{r,\ell} \mid t_{j,\ell} \in [t-10, t]\}$.
   - **Sorted order books**:
     - Bids (from $\mathcal{B}^r(t)$): Descending list $\langle q_{(1)}^r, q_{(2)}^r, \ldots \rangle$ with $q_{(1)}^r \geq q_{(2)}^r \geq \ldots$.
     - Asks (from $\mathcal{E}^r(t)$): Ascending list $\langle e_{(1)}^r, e_{(2)}^r, \ldots \rangle$ with $e_{(1)}^r \leq e_{(2)}^r \leq \ldots$.
3. **Matching & Pricing**
   - **Trade trigger**: A match occurs if there exists any $q_i^{r,k} \geq e_j^{r,\ell}$ among current active quotes.
   - **Price**: Set by the earlier-submitted quote:

$$p_{ij}^r = \begin{cases} q_i^{r,k} & \text{if } t_{i,k} < t_{j,\ell} \\ e_j^{r,\ell} & \text{if } t_{j,\ell} < t_{i,k} \end{cases}$$

     In case $t_{i,k} = t_{j,\ell}$, resolve ties randomly or by participant index order.
   - **Post-trade**: Matched buyers and sellers are removed from the order book and cannot submit further quotes in round r.
4. **Round Termination**
   - The round ends at $t = T$ seconds or when all active buyers and sellers have traded.

Task Description Referred to as `{task_desc}`, this is the instruction that specifies target behavior.

Ultimatum Game

Both players are assumed to be bounded rational.
Your task is to design a model to explain the strategies of proposers in the game.
The model should provide the likelihood of proposers' offer based on the total amount. i.e. $Pr(a|o)$, where $a \in [0, 1]$

Repeated Rock-Paper-Scissors Game

Human players in repeated Rock-Paper-Scissors (RPS) games are assumed to operate under bounded rationality, influencing their choice strategies over successive rounds.
Your task is to design a probabilistic policy model to explain these adaptive player strategies.

The model should provide the likelihood of a player selecting an action $a_t \in \{\text{Rock, Paper, Scissors}\}$ based on the observed history of play $\boldsymbol{o}_t$, i.e., $P(a_t|\boldsymbol{o}_t)$.

Continuous Double Auctions

Information Structure

1. **Private Information**
   - Each participant knows only their own private reservation price: β_i for buyer b_i, or σ_j for seller s_j.
2. **Public Information**
   - **Order Book Display**: All participants have access to a real-time display of the active bid order book ($\mathcal{B}^r(t)$) and ask order book ($\mathcal{E}^r(t)$). These books show dynamically sorted lists of quotes. A participant's own active quotes are typically highlighted in their personal view of this display.
   - **Trade Price Broadcast**: When a trade occurs, its transaction price (p^r_{ij}) is publicly announced to all participants for a short duration (e.g., 3 seconds).
   - **Quote Validity and Expiration**: Active quotes are generally displayed with information about their remaining validity (e.g., via a countdown timer). Submitted quotes remain active for a specific period (e.g., 10 seconds) unless they are updated or result in a trade.

Task Specification
Your task is to design a model to explain the strategies of buyers in the auctions.
Buyer information data interface
For buyer b_i in round r at timestep τ:
Normalization Convention
For buyer b_i, define the normalization operation:

$$\text{normalize}_i(x) := \frac{x}{\beta_i}$$

This normalization applies to all price or quote values provided to buyer b_i's information interface, including those originating from the buyer themself or other market participants. All subsequent references to price-like data in the interface use this normalized scaling, consistent with the definition of $\boldsymbol{o}_\tau$.
Information Interface
The contextual information $\boldsymbol{o}_\tau = (\tau, \boldsymbol{h}_\tau, \boldsymbol{p}_\tau, \mathcal{E}_\tau, \mathcal{B}_\tau)$ available to buyer b_i (for modeling purposes) in round r at current timestep τ, with all price-related values normalized by β_i, comprises:

- **Current Timestep (τ)**: The current timestep τ (in seconds) within the ongoing round r.
- **Buyer's Historical Bids ($\boldsymbol{h}_\tau$)**: A record of all bids previously submitted by buyer b_i within the current round r up to timestep τ. Each entry in this history details:
  - The normalized price of the bid.
  - Its current status (active or expired), reflecting the public quote expiration rule (e.g., the 10-second rule).
- **Historical Transaction Prices ($\boldsymbol{p}_\tau$)**: A list of normalized transaction prices from all trades that have been publicly broadcast in the current round r up to timestep τ.
- **Active Ask Order Book ($\mathcal{E}_\tau$)**: Structured data derived from the public ask order book ($\mathcal{E}^r(\tau)$), presented as a list of active asks. Each entry provides its normalized ask price.
- **Active Bid Order Book ($\mathcal{B}_\tau$)**: Structured data derived from the public bid order book ($\mathcal{B}^r(\tau)$), presented as a list of active bids. Each entry provides:
  - The normalized bid price.
  - An indicator specifying if the bid originates from the current buyer b_i.

The model should provide the action probability function $\pi_{\boldsymbol{\theta}}(a_\tau|\boldsymbol{o}_\tau)$, where a_τ is the buyer's new bid (action) at timestep τ, normalized by their reservation price β_i.

D Experimental Details

Hardware Usage The training, validation, and testing for the strategy models were conducted locally on an Apple M3 chip. Due to limitations in local computational resources, large language models (LLMs) were accessed via vendor-provided APIs rather than utilizing locally deployed LLMs.

Details of Strategy Model Training For the optimization of trainable parameters via gradient descent, we employed the Adam optimizer. The learning rate was configured to 1e-3. Each model was subjected to a maximum of 200 training epochs. To prevent overfitting and reduce training time, an early stopping criterion was adopted, terminating the training process if the validation loss failed to decrease over a period of 10 successive epochs.

E Project Code Access

The project source code is available at the following link: https://github.com/Crescenx/autobm.

Guidance for project execution is provided in the accompanying `README.md` document.