Game of Thoughts: Iterative Reasoning in Game-Theoretic Domains with Large Language Models

Benjamin Kempinski
Radboud University
Nijmegen, Netherlands
benjaminkempinski@gmail.com

Ian Gemp
Google DeepMind
London, United Kingdom
imgemp@google.com

Kate Larson
Google DeepMind/University of Waterloo
Montreal/Waterloo, Canada
katelarson@google.com

Marc Lanctot
Google DeepMind
Montreal, Canada
lanctot@google.com

Yoram Bachrach
Meta
London, United Kingdom
yorambac@gmail.com

Tal Kachman
Radboud University
Nijmegen, Netherlands
kachman.tal@gmail.com

ABSTRACT

We explore the strategic reasoning capabilities of large language models (LLMs). We first show that naively allowing LLMs to select actions in games can lead to sub-optimal and easily exploitable strategies. To address this limitation we propose several algorithms that guide LLMs to iteratively refine their action choices by simulating game outcomes in self-play, akin to cognitive hierarchy models used to characterize human thought processes in strategic settings. Our empirical results in several prominent resource allocation and auction settings indicate that our approach produces stronger and less exploitable strategies. Hence, emulating human decision-making models can enable us to improve the reasoning capabilities of LLMs in multiagent interactions.

KEYWORDS

Game Theory, LLMs, Theory of Mind

ACM Reference Format:
Benjamin Kempinski, Ian Gemp, Kate Larson, Marc Lanctot, Yoram Bachrach, and Tal Kachman. 2025. Game of Thoughts: Iterative Reasoning in Game-Theoretic Domains with Large Language Models. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 10 pages.

1 INTRODUCTION

Large Language Models (LLMs) have achieved strong performance on many tasks, from question answering and text summarization to code generation and natural language understanding [13, 28, 29]. There has been recent excitement in the possibility of using LLMs as autonomous agents that perform complex tasks on their own or with others [23, 33, 34, 43, 48, 50]. To perform well, LLM agents must have strong reasoning and planning capabilities [17, 18, 41]. This is especially demanding in settings where the agent is also affected by the choices made by other intelligent agents occupying the same environment [15, 24, 35, 37, 51, 54]. Some work has attempted to evaluate or improve LLM "Theory of Mind", i.e. the ability of LLMs to reason about the beliefs and thoughts of other agents [7, 36, 40, 42]. There are also methods for improving the reasoning capabilities of LLMs by re-querying the LLM or breaking down the process of responding to the query into smaller reasoning steps, such as Chain of Thought and similar methods [14, 38, 44, 45].

Despite this, there are still many limitations in the reasoning and planning capabilities of LLMs (e.g. [17, 41]), especially when reasoning, strategically, about other agents [1, 15, 26, 35, 53, 54].

Game theory studies interactions between self-interested agents that are affected by each other's actions [30], providing a foundation for studying multi-agent capabilities of LLMs [10, 11]. In such settings multiple agents operate in the same environment, making strategic action choices knowing that their utility depends not only on their own actions but also on the actions of their peers.

Our Contribution: We study the strategic behaviour of LLMs. Using well-studied games from the literature, including Prisoners' Dilemma, Colonel Blotto [31], all-pay auctions [5] (and others), we show that naïvely using LLMs to select actions leads to sub-optimal and easily exploitable strategies. Taking inspiration from the behavioural economics literature [8], and in particular cognitive hierarchy models (CHT) such as level-k reasoning [4, 49] that model human strategic reasoning [4, 9, 47, 49], we propose a family of three algorithms: *Best Response Iterative Reasoning Language Model (BRIRLM)*, *Fictitious Play Iterative Reasoning Language Model (FPIRLM)*, and *Policy Space Response Oracle Language Model (PSROLM)*. These algorithms combine LLM responses with an "outer loop" that allows for levelled reasoning by simulating game outcomes under different assumptions of the strategic capabilities of the other agents. We show empirically that our approach produces stronger and less exploitable strategies compared to LLM benchmarks.

1.1 Notation and Preliminaries

We describe basic concepts from game theory used in this work [30].

In a two-player **normal form game** G, players select actions simultaneously, and the outcome is determined by the choices of both players. Let $A^1 = \{a_1^1, \ldots, a_k^1\}$ be the set of pure strategies available to player 1 and let $A^2 = \{a_1^2, \ldots, a_m^2\}$ be the strategies available to player 2. Denote the set of **strategy profiles**, consisting of a strategy choice for *both* players as $A = A_1 \times A_2$. The utility of

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

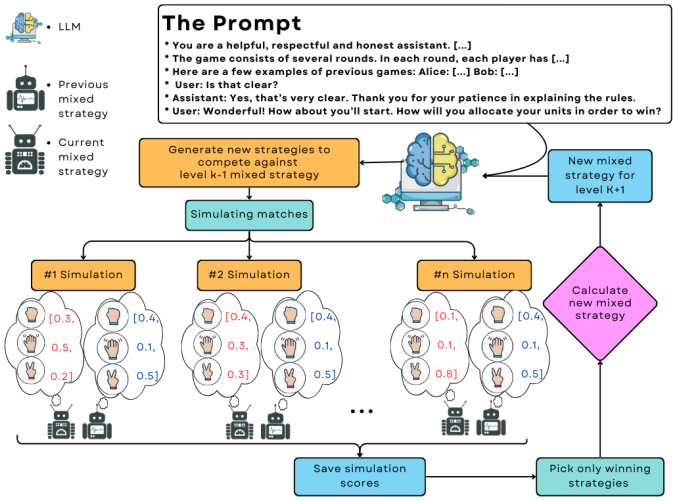


Figure 1: Overview of our iterative reasoning approach. To generate level 0 strategies we prompt the LLM with a game description and ask that it return an action. We iteratively refine strategies through multiple iterations, asking the LLM to produce candidate strategies by describing the game *and the previously produced actions* **(levels 0,1 ...$k-1$).**

each player depends on the actions selected by both for them, i.e. the payoffs are $u : A \to \mathbb{R}^n$, where $u(a) = (u_1(a), u_2(a))$ for $a \in A$, and where each player i tries to maximize their individual utility u_i. A game is zero-sum if for any profile A we have $\sum_i u_i(A) = 0$.

A mixed strategy is a probability distribution Δ over pure strategies, i.e. $\Delta = (p_1, \ldots, p_n)$ where p_i denotes the probability that Δ assigns to pure strategy s_i (so $\sum_{i=1}^{k} p_i = 1$). Given a mixed strategy Δ, we denote sampling a random pure strategy s from this distribution by $s \sim \Delta$, and obtaining a sample of k pure strategies from the mixed strategy Δ by $(s_1, s_2, \ldots s_k) \sim \Delta$ (short for $s_i \sim \Delta$ for all $i \in [k]$).

Given a mixed strategy profile $\sigma = (\sigma_1, \sigma_2)$ the expected utility of u_i of player i is $u_i(\sigma_1, \sigma_2) = \sum_{(a_1,a_2)\in A} \sigma_1(a_1)\sigma_2(a_2)u_i(a_1, a_2)$. The best response for player i to his opponent's strategy σ_{-i} is $BR(\sigma_{-i}) = \arg\max_{\sigma_i \in \Delta(A_i)} u_i(\sigma_i, \sigma_{-i})$, while a Nash equilibrium for a two-player game is defined as a pair of mutually best-responding strategies. We will be particularly interested in **exploitability** of a strategy, σ_i defined as $e(\sigma_i) = [\max_{\sigma_i' \in \Delta(A_i)} u_i(\sigma_i', \sigma_{-i}) - u_i(\sigma)]$, where σ_i' are all alternative strategies of player i. The lower the value of $e(\sigma_i)$, the less exploitable that mixed strategy is. Less exploitable strategies are considered to be of higher quality, as they make it difficult for the opponent to capitalize on their knowledge of the strategy and take advantage of the player.

2 METHODOLOGY

We develop algorithms for improving game-theoretic reasoning by LLMs, to enable LLM agents to select better-performing and less exploitable strategies in games. Our overall approach mimics the logic of cognitive hierarchy models or k-level iterative reasoning models [4, 9], as shown in Figure 1. The approach starts with an initial strategy, level 0, proposed by the LLM, and iteratively refines it. Each refinement step uses the LLM to respond to the previous strategy at step k and applies a simulation to identify promising improvements and generate the next-level strategies for step or level $k+1$. We propose three specific algorithms, all based on the same theme of combining LLM-based strategy generation with game simulations. **BRIRLM**, short for Best Response Iterative Reasoning Language Model, only considers level $k-1$ strategies when producing level k strategies; **FPIRLM**, short for Fictitious Play Iterative Reasoning Language Model, responds to a uniform mixture of all the previous levels, $1, 2, \ldots, k-1$, inspired the Fictitious Play algorithm [6]; **PSROLM**, short for Policy Space Response Oracle Language Model, where the best response at level k is based on responding to the Nash equilibrium over the previous levels $1, 2, \ldots, k-1$, inspired by Response Oracles [22, 27] from the game theory and reinforcement learning literature (sometimes called "Double Oracle" algorithms). We also consider two LLM baselines; **LMNS** (Language Model Naïve Strategy), directly queries the LLM to produce game strategies; **CoT**, is based on Chain-of-Though [44, 45] querying of the LLM to create strategies.

Our algorithms apply a cognitive-hierarchy-inspired model, combining LLMs with a multi-agent simulation of game outcomes. Our LLM building blocks query an LLM to obtain a strategy in the game. The first, *LLMGameQuery*, queries the LLM with a prompt P describing the game rules, and requesting the LLM to select a strategy in the game. The second, *LLMResponseQuery*, is similar but asks the LLM to choose a strategy in the game assuming that the *opponent is likely to choose some strategy s*. Hence, *LLMResponseQuery* asks the LLM to best respond to the opponent strategy s.

LLM Choice: We used the Llama-2-70B [39] model (see the appendix for experiments with other LLM engines and rationale for choosing this model).

Prompts for LLMGameQuery and LLMResponseQuery. A *LLMGameQuery* prompt asks the LLM to generate a (pure) strategy in the game given its description, without specifying what the opponent might do. In contrast, a *LLMResponseQuery* prompt also includes a description of a likely opponent strategy. Both types of prompts are composed of several parts, explaining the desired role and characteristics of the LLM, the game rules, player identities, possible actions, etc. We used the following prompt structure: 1. General LLM system guidelines to the LLM (Purple) *"You are a helpful, respectful and honest assistant..."*; 2. The rules of the game (Red) *"The game consists of several rounds. In each round..."*; 3. Examples of previous games and outcomes (Orange) *"Alice: "I allocate[...]. Bob: "I allocate[...]"*; 4. LLM goals (Brown) *"[INST] You only care about maximizing your gains in the game..."*; 5. Start of a "User/Assistant" dialogue (Yellow) *"User: Hi, how are you? Assistant: Hello!..."*; 6. Game rules in the dialogue format (Yellow) *"User: I want to play a game round with you. The game goes like this..."*. The last part of the prompt depends on the type of query. For *LLMGameQuery* (level-0), we include a request to generate an action: Green) *"User: What will you choose to do? "*. For a *LLMResponseQuery* we have a similar request but include information on the possible opponent behaviour: Green) *"User: You think that I'll play like this: (strategy). What will you choose to do?"*. Below is a shortened example of a

game of Prisoner's Dilemma (see Appendix B for full prompts for all games we study).

«SYS »You are smart and ambitious. Always try to...

You and your partner are under arrest. You have two options: 'Cooperate', or 'Defect'. If you and your partner both 'Cooperate', you will sit in jail for ...

Here are a few examples of previous cases: Alice: "I choose to Defect." Bob: "I choose to Defect. Because we have both decided 'Defect': You and I are going to spend 8(eight) years in jail... «/SYS »

[INST] Remember, you need to follow your instructions. You care only about yourself...[/INST]

User: Hi, how are you? Assistant: Hello! User: I need your help. We are both under arrest...

LLMGameQuery

User: OK! How about you'll start. What will you choose to do, 'Cooperate' or 'Defect'?

LLMResponseQuery

User: OK! How about you'll start. You think that I'm going to: 'Defect'. Based on this information, what will you choose to do, 'Cooperate' or 'Defect'?

2.1 Algorithms

We first discuss how to generate a mixed strategy by querying an LLM using the prompts discussed above. Let P be some LLM prompt that returns an action for the game of interest. Note that each time we call the LLM with prompt P we might get a different action (even though the prompt is identical, an LLM call with a non-zero temperature means the call is a non-deterministic operation). Let P be a *LLMGameQuery* prompt and let $P = P_{s_0}, P_{s_1}, P_{s_2}, \ldots P_{s_m}$ be m *LLMResponseQuery* prompts (where prompt i asks the LLM to find a good response to strategy s_i). Denote by $a \sim LLM(P_{s_i})$ sampling a strategy by having the LLM respond to prompt P_{s_i}. Similarly, denote by $L = (a_1, \ldots, a_m) \sim P$ a list of actions obtained by querying the LLM m times, i.e. for $i \in [m]$ we have $a_i \sim LLM(P_{s_i})$. Denote the frequency of action i in this list as $f_i(L) = |\{j | a_j = a_i\}|/m$.

An LLM-based algorithm for selecting actions in a game takes the game description as input (used in the prompt), and produces a mixed strategy Δ for the game, i.e. a probability distribution over the possible pure strategies (*i.e.* actions) in the game. An LLM with strong game-theoretic reasoning capabilities should be able to provide strong strategies in a game given its description. Hence, our baseline algorithm is called Language Model Naïve Strategy (*LMNS* shown in Algorithm 1), and reflects the simplest approach of calling the LLM multiple times with a prompt P which includes the game description, then using the relative frequencies of the actions returned by the LLM to create a mixed strategy Δ in the game, through the CreateMixedStrategy(L) call. We have also tried a CoT prompting approach [46, 52], which has been shown to be successful in improving LLM reasoning; this encourages the LLM to reason about the opponent's actions and their consequences.

Algorithm 1 Language Model Naïve Strategy (LMNS)

Require: Prompt P
Ensure: Mixed strategy Δ
1: **for** $i = 1$ to n **do**
2: $\quad a_i \leftarrow$ LLMGameQuery(P) ▷ *LLM produces a pure strategy*
3: $\Delta \leftarrow$ CreateMixedStrategy($L = (a_1, a_2, \ldots a_n)$)
4: **return** Δ ▷ *Return the mixed strategy*

This second baseline, CoT is identical, except the prompt given the LLM is a Chain-of-Thought prompt, shown in full in appendix B. This prompt asks the LLM first to explain what strategy it believes the opponent is likely to play, and then to select its own strategy given its previous response.

Earlier work, as well as our empirical results (see Section 3.2), show that simply requesting a strategy from the LLM yields poor performance even for simple games (both with LMNS or CoT). We propose improving the game-theoretic reasoning of LLMs through algorithms employing a CHT process, combining LLMs and multi-agent simulation; they first create a "level-0" mixed strategy Δ_0 by querying the LLM (via *LLMGameQuery*), then iteratively refine the mixed strategy, progressing from level-1 to level-2 and so on (where each level k is reflected through a mixed strategy Δ_k). The algorithms (BRIRLM, FPIRLM, PSROLM) differ in how the next level k is obtained from the previous level strategies $\Delta_0, \Delta_1, \ldots, \Delta_{k-1}$

BRIRLM constructs the next level strategy Δ_k by best responding to strategies sampled from the previous level Δ_{k-1}. Similarly to *LMNS*, it first queries the LLM multiple times, to generate several game strategies $a_1, \ldots, a_m$; however, each such strategy a_j is generated by having the LLM *respond* to a strategy s_j sampled at random from the previous level $s_j \sim \Delta_{k-1}$, i.e. by calling *LLMResponseQuery*(P_{s_j}) (where P_{s_j} is a prompt stating that the opponent is likely to play s_j). These strategies $a_1, \ldots, a_m$ are considered as *candidate strategies* to be evaluated. BRIRLM determines the probability of playing each candidate strategy a_c in the mixed strategy Δ_k generated for the next level k by applying g *simulation rounds* (where g is a constant algorithm parameter). In each simulation round, we generate a score d_c for each candidate a_c by computing its mean payoff against a sample $X = (x_1, x_2, \ldots x_q)$ of strategies sampled from the level $k-1$ strategy Δ_{k-1} (the sample size q is a constant algorithmic parameter). [1] The function for computing this score is *SimulationScoring*, given in Algorithm 2.

The j-th simulation round in BRIRLM ends by selecting the strategy a_{w_j} with the highest score $w_j = \arg\max_t d_t$ as the winner candidate of the simulation round. The relative frequency in which candidate a_c is the winner of a simulation round is used to determine its probability under the next-level mixed strategy Δ_k (i.e. the more simulation rounds a candidate a_c wins, the higher its probability in Δ_k). as such, the complexity of BRIRLM is $O(kgm)$, where

[1] If the number of strategies is low enough, we can exactly compute the expected payoff of candidate a_c against the mixed strategy Δ_{k-1}, but this is intractable when the number of strategies in the game is very high.

Algorithm 2 Simulation scoring

Require: Candidate strategy a, strategy sample $L = (s_1, s_2, \dots s_q)$
Ensure: Candidate score against the sampled strategies.

```
for i = 1 to |L| do
    u_i ← u(a, s_i)                          ▷ Payoff vs. s_i
return (Σ_{i=1}^{|L|} u_i) / |L|             ▷ Mean payoff vs. sample L
```

k is the k-iterative levels, g is the number of simulation rounds and m is the number of LLM calls per round. The full BRIRLM algorithm is given in Algorithm 3.

Algorithm 3 Best Response Iterative Language Model (BRIRLM)

Require: LLM, Number of simulation candidates q
Ensure: Mixed strategy Δ_k

```
for k = 1 to n do
    for j = 1 to m do                        ▷ Generate m candidate strategies
        s_j ~ Δ_{k−1}                        ▷ Sample from the level k − 1 distribution
        a_j ← LLMResponseQuery(P_{s_j})      ▷ Response to s_j
    for j = 1 to g do                        ▷ Perform g simulation rounds
        X = (x_1, x_2, ... x_q) ~ Δ_{k−1}    ▷ Sample from Δ_{k−1}
        for c = 1 to m do
            d_c ← SimulationScoring(a_c, X = (x_1, x_2, ... x_q))
        w_j = arg max_t d_t                  ▷ Get index w of strategy a with the highest mean score vs. sample X
    L_k ← (a_{w_1}, a_{w_2}, ... a_{w_g})
    Δ_k ← CreateMixedStrategy(L_k)
return Δ_k                                   ▷ Resulting level k mixed strategy
```

FPIRLM uses similar building blocks as *BRIRLM* (and so has the same complexity of $O(kgm)$), but applies a process akin to the Fictitious Play [6] algorithm (FP) from game theory. FP iteratively selects strategies, where the strategy chosen in step k is a best response to strategies selected from *all* the previous levels $0, 1, \dots, k-1$. Hence, *FPIRLM* is different from *BRIRLM* in that it attempts to identify strategies that do well not just versus the previous level Δ_{k-1}, but versus a uniform random mixture of *all* the previous levels $\Delta_0, \Delta_1, \dots \Delta_{k-1}$; *FPIRLM* creates a set of candidates $a_1, \dots, a_m$, but rather than having the LLM *respond* to a strategy s_j sampled at random from the *immediate previous level* $k-1$, it first chooses a historical level r uniformly at random from $0, 1, \dots, k-1$, then selects a strategy from that level $s_j \sim \Delta_r$, by calling $LLMResponseQuery(P_{s_j})$. Similarly to *BRIRLM*, the candidates are examined in g simulation rounds, each computing the mean score of all candidates versus a sample $X = (x_1, x_2, \dots x_q)$ of strategies. However, *FPIRLM* builds the sample X by selecting strategies from *all* the previous levels $\Delta_0, \dots \Delta_{k-1}$ (each strategy x_t in the sample X is chosen by first selecting a level r uniformly at random from $(0, 1, \dots, k-1)$, then selecting the strategy x_t from that level Δ_r). Again, in each simulation round we select the top-performing candidate strategy against the sample X, and use the relative frequency in which candidate a_c is the winner of a simulation round as its probability under the next-level mixed strategy Δ_k. FPIRLM is shown in Algorithm 4.

Algorithm 4 Fictitious Play Iterative Language Model (FPIRLM)

Require: LLM, Number of simulation candidates q
Ensure: Mixed strategy Δ_k

```
for k = 1 to n do
    for j = 1 to m do                        ▷ Generate m candidate strategies
        r ~ (0, 1, ... k − 1)                ▷ Select previous level uniformly
        s_j ~ Δ_r                            ▷ Sample strategy from selected level
        a_j ← LLMResponseQuery()(P_{s_j})    ▷ Respond to s_j
    for j = 1 to g do                        ▷ Perform g simulation rounds
        for u = 1 to g do                    ▷ Generate a sample using a mixture over previous levels
            r ~ (0, 1, ... k − 1)            ▷ Select previous level uniformly
            x_q ~ Δ_r                        ▷ Sample strategy from selected level
        X = (x_1, x_2, ... x_q)
        for c = 1 to m do
            d_c ← SimulationScoring(a_c, X = (x_1, x_2, ... x_q))
        w_j = arg max_t d_t                  ▷ Get index w of strategy a with the highest utility score
    L_k ← (a_{w_1}, a_{w_2}, ... a_{w_g})
    Δ_k ← CreateMixedStrategy(L_k)
return Δ_k                                   ▷ Resulting level k mixed strategy
```

PSROLM is our last LLM-based iterative method, inspired by Response Oracles algorithms such as the Double Oracle method [27] and Policy Space Response Oracle [22], which maintain a set of pure strategies in the game and iteratively adds strategies into it. In iteration k, the standard PSRO computes a Nash equilibrium over the current strategies X_{k-1} to obtain a mixed strategy Δ_{k-1}, then identifies a pure strategy s_k that best responds to Δ_{k-1}, and adds it to the set so $X_k = X_{k-1} \cup \{s_k\}$. Our PSROLM method follows a similar recipe, but uses an LLM and multi-agent simulation to select a strategy to add. In our PSROLM, level $k-1$ consists of a mixed strategy Δ_{k-1} which is the Nash equilibrium distribution over the strategy support set X_{k-1}. To obtain the next strategy to add to the set X, PSROLM prompts the LLM to respond to a strategy s_j selected at random from Δ_{k-1} through calls to $LLMResponseQuery(P_{s_j})$. It then adds to X the strategy s_w which was returned most frequently from the LLM (so $X_k = X_{k-1} \cup \{s_w\}$), and computes the Nash equilibrium over the strategy set X_k to obtain the new mixed strategy Δ_k. [2] We initialize the set X to two strategies chosen at random. The full PSROLM approach is given in Algorithm 5.

3 EMPIRICAL ANALYSIS

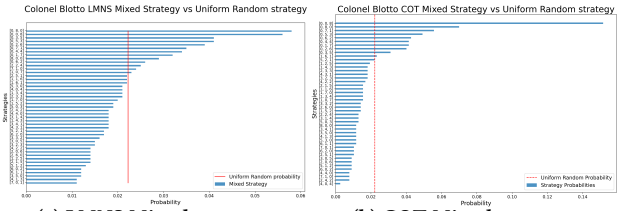
We evaluate our algorithms and the baselines over multiple games to see which produces superior or less exploitable strategies. The payoff a strategy achieves also depends on the opponent's strategy, so we provide comparisons between each pair of algorithms / baselines. We also present the exploitability of the strategies produced by each algorithm (which does not depend on the opponent's strategy).

[2] The Nash equilibrium is calculated in each round. In cases where the computation of the Nash equilibrium becomes prohibitively slow (especially for later rounds), one may evict a strategy from the set X, such as the oldest strategy in the set or the one with the lowest probability mass under the Nash equilibrium.

Algorithm 5 Policy Space Response Oracle Language Model (PSROLM)

Require: LLM, Initial Random Strategies s_1, s_2
Ensure: Mixed strategy Δ_k

$X \leftarrow [s_1, s_2] \in \mathbf{A_1}$ ▷ *Initial strategy set*
$\Delta_0 \leftarrow NashEquilibrium(X)$ ▷ *Level 0: initial Nash*
for $k = 1$ to n **do**
  for $j = 1$ to m **do**
    $s_j \sim \Delta_{k-1}$ ▷ *Sample strategy from previous level*
    $a_j \leftarrow LLMResponseQuery(P_{s_j})$ ▷ *Respond to s_j*
  $w \leftarrow \arg\max_t f_t(L = (a_1, a_2, \dots a_m))$ ▷ *Index of most common response*
  $X = X \cup \{a_w\}$ ▷ *Add most frequent response strategy to X*
  $\Delta_k \leftarrow NashEquilibrium(X)$
return Δ_k ▷ *Resulting level k mixed strategy*



(a) LMNS Mixed strategy **(b) COT Mixed strategy**

Figure 2: LMNS (a) and COT (b) mixed strategies in the Colonel Blotto Game. The vertical axis shows all 45 possible unit allocations, while the horizontal axis shows the probability of each strategy. The line indicates uniform random probability. Both LMNS and COT strategies deviate slightly from uniform, having a bias towards empty-field strategies.

3.1 Evaluated Games

We briefly describe the games used in our empirical evaluation (see a more detailed description in the Appendix, and the cited papers for full details). We considered several famous games, ranging from the classical Prisoner's Dilemma [30], through the Colonel Blotto resource allocation game [31] and the Eleven-Twenty and Tennis Coach Games [2, 3], and finally an all-pay auction [5, 12]. Of the many possibile games, we opted to choose those less extensively researched in the context of LLMs.

Prisoner's Dilemma [30], and the variant called Snow Shoveller [21] is a two-player game, where each player has two possible strategies: Cooperate (C) and Defect (D). The game has four possible payoffs, T, R, P, S (for "Temptation", "Reward", "Punishment", "Sucker"), satisfying $T > R > P > S$ and $2R > T + S$. If both players select C, they both have a payoff of R; If both select D they both get a payoff of P. If one player selects C and the other selects D, the player that selected C gets the payoff of S and the one that selected D gets the payoff of T. The game is set up so that D is a dominant strategy: a player is always better off selecting D rather than C (no matter what the other player does).

Colonel Blotto is a resource allocation game [25], well-studied in both theory [32] and human behaviour [4, 19]. The game has two players ("colonels"), and player $i \in \{1, 2\}$ has a total of b_i units, which are to be allocated to m battlefields. A pure strategy for player i is an allocation $q_1^i, q_2^i, \dots, q_m^i$ of his units (with q_x^i of i's units allocated to battlefield $x \in \{1, \dots, m\}$, so $\sum_{x=1}^{m} q_x^i = b_i$. Player i wins battlefield x if they have more troops there, i.e. if $q_x^i > q_x^j$ (similarly, if $q_x^j > q_x^i$ then j wins battlefield x). If $q_x^i = q_x^j$ then neither player wins battlefield x. The game is zero-sum, with the player who won more battlefields achieving a payoff of 1 and the other player achieving a payoff of -1 (if the players win an equal number of battlefields then both get a payoff of 0).

Eleven-Twenty is a game designed to study k-level iterative reasoning [2], where the strategies consist of all integers in the range eleven to twenty (inclusive). Each player gets an amount of money identical to their chosen number; however, if a player selects exactly one less than their peer, they get a bonus of twenty. For instance, if Alice asks for 15 and Bob asks for 14, Alice achieves a payoff of 15, while Bob's payoff is $14 + 20 = 24$.

Tennis Coach is another game designed to study iterative reasoning [2]. In the game, two coaches each have four players, with the ability levels $A+, A, B+, B$, ranked from highest to lowest ability. A coach selects an ordering of their players (i.e. a permutation). The first players in the orderings of both coaches compete, then the second players, and so on. A coach obtains 1 point for each game they win and -1 point for each game they lose (a tie gives 0 points to both). For instance, if Alice selects the permutation $A+, A, B+, B$, and Bob selects $A, B+, B, A+$ then Alice wins the first game ($A+$ vs A), wins the second game (A vs $B+$), wins the third game ($B+$ vs B), and loses the fourth game (B vs $A+$); hence Alice gets a payoff of $1 + 1 + 1 - 1 = 2$ and Bob a payoff of $(-1) + (-1) + (-1) + 1 = -2$.

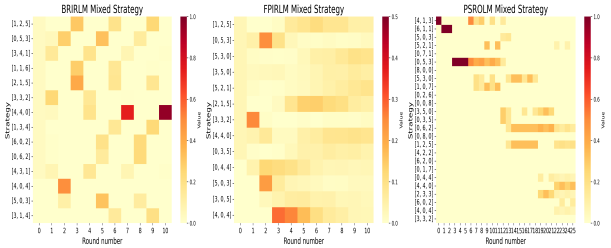
All-Pay Auctions are a well-studied form of auctions [5, 12, 20]. In the game, every player submits a bid b, and the player with the highest bid obtains a reward worth r dollars; all the players (including whoever did not win) pay their bid. The highest bidder with a bid of b^w thus gets $r - b^w$ and a non-winner who bid b^l gets $-b^l$. In the case of a tie with bid b, both players equally share the reward worth r and pay their bid b, yielding a reward of $\frac{r}{2} - b$ for both. We use a reward of $r = 16$, and have discretized bids, so the allowed bids are $\{0, 1, 2, \dots, 16\}$.

3.2 Empirical Results

We will show the main results for the Colonel Blotto game here. For the exact prompts, see appendix B. For supplementary results in Colonel Blotto, Prisoners dilemma, Eleven-Twenty, Tennis coach and First-bid all-pay auction, see appendices C to G respectively.

3.2.1 Colonel Blotto. **LMNS & COT** In order to evaluate our frameworks, we set LMNS as our baseline model for comparison. Running LMNS according to Algorithm 1, we generated its mixed strategy, which we found has full support (see Figure 2).

The LLM has a bias towards weak strategies: 12 out of 13 of the strategies LMNS favours are easily defeated; The two most likely strategies have two empty fields, preventing them from winning any games, and reaching a draw at best. This makes them poor strategies to pick, as they provide no opportunity for victory. The majority of the remaining favoured strategies have one empty field, which while not guaranteeing a loss, puts these strategies at a clear disadvantage by reducing the winning possibilities.



(a) BRIRLM strategy evolution (b) FPIRLM strategy evolution (c) PSROLM strategy evolution

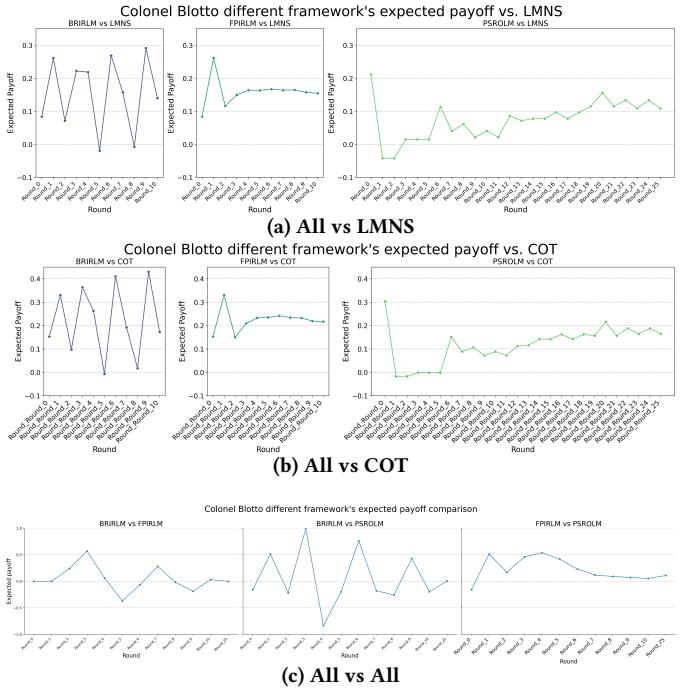
Figure 3: Heatmap of mixed strategy evolution of Colonel Blotto tournament, from left to right for BRIRLM, FPIRLM and PSROLM . We ran BRIRLM and FPIRLM for 10 rounds, while PSROLM for 25 rounds. The vertical axis represents the various strategies that appeared at least once in the mixed strategy. The horizontal-axis represents the round number. The colour bar represents the probability of each strategy in the round's mixed strategy.

Given the limitation of the baseline LMNS, we explored common approaches to improve its result at the prompt level. Specifically, we implemented COT prompting to see if it could enhance the quality of the generated strategies. The mixed strategy results of this COT implementation can be seen in Figure 2 (b).

We ran COT up to level 2. Level 1 involves the LLM anticipating and responding to its opponent's play, while level 2 extends this by responding to its own level 1 suggestion. We found no significant improvement in the mixed strategy quality above level 1, with level 2 results in appendix C. Notably, the LLM's task understanding is limited when comparing ultimate and penultimate strategies, with only a 2% success rate. COT reasoning improves over LMNS in generating strategies surpassing random selection, but falls short in producing consistently strong strategies. Moreover, it does not effectively mitigate the hallucinations and misunderstandings inherent in the reasoning tasks.

Our frameworks We implemented BRIRILM, FPIRLM and PSROLM as described in Algorithms 3, 4 and 5 respectively. BRIRILM and FPIRLM ran for 10 rounds each, while PSROLM was extended for 25 rounds. The progression of the resulting mixed strategies for all three frameworks can be seen in Figure 3. Our tournament structure followed the schema illustrated in Figure 1.

For BRIRLM and FPIRLM, the initial iteration ($k = 1$) was played against a uniform-random distribution of strategies. In subsequent rounds, competing strategies were selected from the mixed strategy of the previous levels. Due to the nature of these frameworks, there is no difference in results between $k = 0$ and $k = 1$ of BRIRLM and FPIRLM. However, subsequent levels reveal distinct differences among all three frameworks. PSROLM (Figure 3, left) shows a progression towards more diffuse mixed strategies (i.e., with higher support). In (Figure 3, middle), BRIRLM exhibits a cyclic pattern, where strategies defeat the immediately previous level, but performs poorly against those that appear before it (i.e. level k defeats level $k-1$, but may lose to $k-2$). These cycles follow a three-round periodicity, shown in Figure 5(b). In (Figure 3, right), the FPIRLM tournament shows that as the rounds progress, the weight of any single strategy diminishes. This is due to the absence of a single dominant strategy for the game, resulting in an increasingly complex mixed strategy comprising all previous rounds.



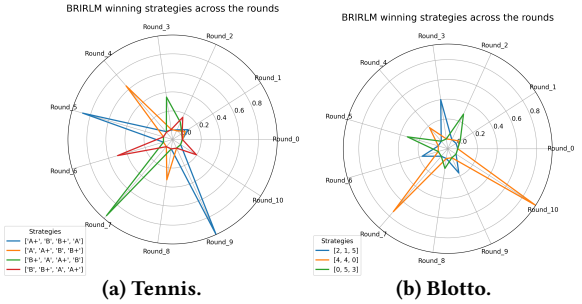
(a) All vs LMNS

(b) All vs COT

(c) All vs All

Figure 4: Comparison of expected payoff across all rounds of three frameworks (BRIRLM,FPIRLM,PSROLM) against LMNS (a) and COT (b), respectively. Values are from the perspective of the left framework (i.e. Positive values for left superiority, negative for right superiority) Figure (c) compares each of the three frameworks against each other. The comparison of expected payoffs among the frameworks is complex, particularly due to the cyclic nature of BRIRILM. While BRIRLM achieves the highest single expected payoff value, its performance is unstable. FPIRILM demonstrates a better expected payoff compared to PSROLM for most rounds, but this advantage decreases over time.

Following the tournaments, we compared the expected payoff of the different frameworks against LMNS and COT. The results, shown in Figure 4, demonstrate that regardless of the round, the expected payoff of all three frameworks is equal to or better than both baselines, LMNS and COT. FPIRLM and PSROLM show moderate, but steady progression in their performance against LMNS and COT across their different rounds. Summarizing BRIRLM's expected payoff is more challenging due to its cyclic strategy selection across levels. Nonetheless, both BRIRLM's high and low values are equal or better than both LMNS and COT, with its peaks achieving the highest absolute expected payoff values amongst the different frameworks. The tendencies observed in the comparison with the baselines persist when measuring the expected payoff of the different frameworks against each other, as seen in Figure 4(c). The evaluation of BRIRLM against the other two frameworks shows inconsistency due to its cyclic behaviour. In contrast, the comparison



(a) Tennis. (b) Blotto.

Figure 5: Radar plots showing the BRIRLM strategies cycling over four-rounds (a) and three-rounds (b).

between FPIRLM and PSROLM reveals a clearer pattern: FPIRLM consistently demonstrates superior expected payoff, although this advantage gradually diminishes over successive rounds.

The exploitability results for LMNS, BRIRLM, FPIRLM, PSROLM can be seen in Figure 7. We show that there exists a tradeoff pattern between exploitability and expected payoff. LMNS, having a broader more random approach, has a low exploitability. FPIRLM improves over LMNS in exploitability, and reaches the lowest exploitability of all frameworks by round 7. BRIRLM achieves the highest expected payoff value among the different frameworks. However, as seen in Figures 4 & 6 the BRIRLM's mixed strategy changes significantly each round, alternating between high and low expected payoffs. This also renders it the most exploitable of the different frameworks.

PSROLM requires more rounds than the other frameworks to converge to a viable mixed strategy. Through rounds 0 to 7, PSROLM has high exploitability, as well as a relatively low expected payoff; In rounds 7 to 16, we see a steady increase in the expected payoff. We also see a steady decrease in PSROLM exploitability, suggesting that the PSROLM's mixed strategy is adapting with each round; From rounds 17-20, during which we start switching strategies due to computation costs, PSROLM exploitability increases again, and quickly. It is worth noting that while by a small margin, its expected payoff remains higher compared to all other frameworks.

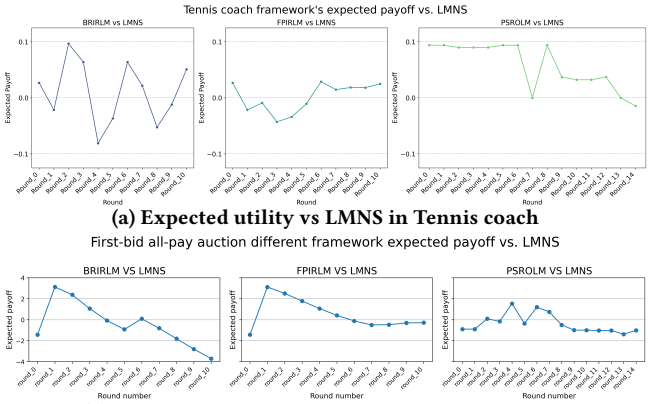
Our frameworks' nature can be seen in other games as well, both when looking at their exploitability and when comparing with the baseline. For instance, see the results for Tennis Coach and All-Pay Auction in Figure 6, or Figure 5 left. More detailed results for all the games are shown in the appendix.

3.3 Theoretical results

We now analyze a stylized version of the PSROLM framework, shown in Algorithm 6, and prove it converges to a Nash equilibrium.

We can implement the InitialStrategies() and the SuggestResponse() functions using calls to an LLM. For InitialStrategies(), we provide the LLM a textual description of the game, and ask it which strategies it would start with. For the SuggestResponse(), we prompt the LLM with a list of k strategies sampled from $D(S)$ and ask the LLM for a best response to this list. Alternatively, we could repeat this procedure many times and take the response with the highest expected payoff against $D(S)$.

We now show that under mild assumptions, this "Las Vegas" algorithm indeed converges to the true Nash equilibrium $(S^*, D(S^*))$ of the game.



(a) Expected utility vs LMNS in Tennis coach

(b) Expected utility vs LMNS in the all-pay auction

Figure 6: Comparison of expected payoff across all rounds of three frameworks (BRIRLM, FPIRLM, PSROLM) against LMNS in Tennis coach (a) and First-bid all-pay auction (b), respectively. Values are from the perspective of the left framework (i.e. Positive values for left superiority, negative for right superiority)

Algorithm 6 Stylized-PSROLM()

Require: Symmetric Game with $|\mathcal{A}|$ actions

Require: Safe: Bool

```
S ← InitialStrategies()
D(S) ← ComputeNash(S)
while True do
    r ← SuggestResponse(S,D(S))
    if ¬Safe and r ∉ S then
        D(S) ← ComputeNash(S)
        Return D(S)
    S ← S ∪ {r}
    D(S) ← ComputeNash(S)
    if |S| == |A| then
        Return D(S)
```

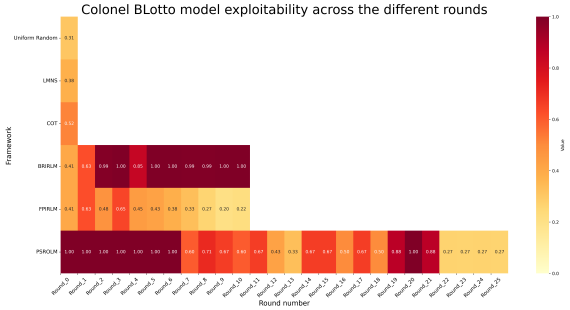
Theorem 1. *If SuggestResponse() has full support over all pure strategies in a finite game, then the above procedure Stylized-PSRO() (with Safe set to True) converges to the true Nash equilibrium of the game (i.e. after a sufficient number of iterations, $(S, D(S))$ is a Nash equilibrium of the game).*

Proof. (Sketch) Calling InitialStrategies() returns some set of strategies in the game. In every iteration of the loop, we have a non-zero probability of adding the true best response strategy and thus extending the set of strategies S by an additional member (even if the true best response is not added, we may still uncover a new strategy and thus add one more strategy to the set). As the set of strategies keeps growing, we would eventually cover *all* strategies in the game, and thus compute the true Nash equilibrium of the game via ComputeNash(). □

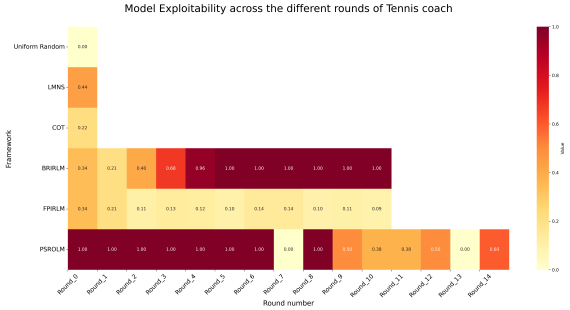
LLMs with softmax distributions over tokens naturally have full support, however, modern sampling techniques such as nucleus sampling [16] may break this assumption in certain settings.

Table 1: Summarisation of LLM success rate. Exp measures exploitability; $> b$ reports whether the framework is beating the baseline b; Stab reports the number of rounds until a stable solution is reached; $*$ indicates unstable results.

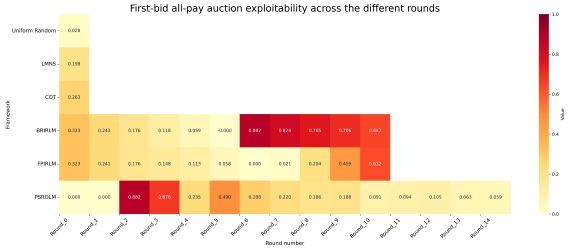
Game	BRIRLM				FPIRLM				PSROLM			
	Exp	>LMNS	>COT	Stab	Exp	>LMNS	>COT	Stab	Exp	>LMNS	>COT	Stab
Prisoner's Dilemma	0	Yes	Yes	2	0	Yes	Yes	2	0	Yes	Yes	2
Eleven-twenty	0	Tied	Tied	2	0	Tied	Tied	2	0	Tied	Tied	2
Colonel Blotto	1	Yes*	Yes*	NaN	0.22	Yes	Yes	5	0.27	Yes	Yes	22
Tennis coach	1	Yes*	Yes*	NaN	0.09	Yes	Tied	7	0.6	Yes	Tied	6
First-bid all-pay	0.64	*	*	NaN	0.63	*	*	NaN	0.059	Tied	No	10



(a) Exploitability in Colonel Blotto



(b) Exploitability in Tennis coach



(c) Exploitability in first-bid all-pay auction

Figure 7: Exploitability comparison across rounds between the frameworks BRIRLM, FPIRLM, PSROLM, as well as LMNS, COT and a uniform random mixed strategy in (a) Colonel Blotto, (b) Tennis coach and (c) First-bid all-pay bidding. In all games, either FPIRLM or PSROLM reach the lowest exploitability, while BRIRLM reaches the highest.

If Stylized-PSRO(Safe=False) is run, the procedure may terminate early when the LLM fails to return a better response even though one exists. Increasing the number of best response samples k can drive this probability towards zero.

4 DISCUSSION

Our results demonstrate that while LLMs show promise in many NLP tasks, their reasoning is still limited when it comes to strategic interactions between multiple agents. Looking at the results of LMNS and COT baselines, representing unaided LLMs, in Figure 2, we see that their mixed strategy is suboptimal and exploitable. LLMs can produce mixed strategies that favour strategies that are easily defeated, e.g. leaving one field free of units and easily captured.

We introduced a new family of algorithms based on iterated reasoning, with three specific algorithms: BRIRLM, FPIRLM and PSROLM. Each allows LLMs to participate in various games, and generate viable and optimal strategies. Each one of the frameworks has its advantages and disadvantages.

The Figures 3, 4 and 5, illustrate the key difference between these frameworks. BRIRLM shows a narrow and cyclic evolution, while FPIRLM and PSROLM show wider, more moderate mixed strategies. As a result of its cyclic nature, the choice of the level of iterative reasoning (i.e. the number of iterations of refining the policy) has a very big impact on the resulting strategy. In contrast, FPIRLM and PSROLM shift their mixed strategy more gradually.

Although LMNS and COT exhibit low exploitability, this stems from redundancy rather than optimisation. In contrast, FPIRLM and PSROLM achieve lower exploitability without needing to use all possible strategies in their mixed strategy (Figure 7). When comparing expected payoff, (Figures 4a & 6) we show that our frameworks consistently outperform both LMNS and COT in most games, and in the few that not, the margin is small.

Unaided LLMs struggle with complex reasoning tasks, especially when the action spaces are large, or the immediate reward is unclear (Table 1). Even in simpler tasks, the unaided LLM is prone to fail sporadically, preventing it from being a reliable tool.

Our implemented frameworks guide the LLM towards relevant, stable results, while also functioning as a safety net to prevent LLM mistakes from occurring, thus preventing them from contaminating the otherwise good LLM outputs.

Overall, our results indicate that one can significantly improve the quality of LLM reasoning in strategic multiagent domains by incorporating an outer-loop that iteratively refines the suggested strategy. Future research could uncover additional methods for refining the policy, and also consider ways of improving the LLM training procedure so as to have LLMs be better at strategic settings without needing to resort to an outer-loop approach (for instance, one could consider generating synthetic data using our approach and using it to fine-tune LLMs).

REFERENCES

[1] Elif Akata, Lion Schulz, Julian Coda-Forno, Seong Joon Oh, Matthias Bethge, and Eric Schulz. 2023. Playing repeated games with Large Language Models. arXiv:2305.16867 [cs.CL]

[2] Ayala Arad. 2012. The tennis coach problem: A game-theoretic and experimental study. *B.E. Journal of Theoretical Economics* 12, 1 (2012). https://doi.org/10.1515/1935-1704.1738

[3] Ayala Arad and Ariel Rubinstein. 2012. The 11–20 money request game: A level-k reasoning study. *American Economic Review* 102, 7 (2012), 3561–3573.

[4] Ayala Arad and Ariel Rubinstein. 2012. Multi-dimensional iterative reasoning in action: The case of the Colonel Blotto game. *Journal of Economic Behavior & Organization* 84, 2 (2012), 571–585.

[5] Michael R Baye, Dan Kovenock, and Casper G De Vries. 1996. The all-pay auction with complete information. *Economic Theory* 8 (1996), 291–305.

[6] George W Brown. 1951. Iterative solution of games by fictitious play. *Act. Anal. Prod Allocation* 13, 1 (1951), 374.

[7] Vivien Cabannes, Charles Arnal, Wassim Bouaziz, Alice Yang, Francois Charton, and Julia Kempe. 2024. Iteration Head: A Mechanistic Study of Chain-of-Thought. arXiv:2406.02128 [id='cs.LG']

[8] Colin F Camerer. 2011. *Behavioral game theory: Experiments in strategic interaction.* Princeton university press.

[9] Colin F Camerer, Teck-Hua Ho, and Juin-Kuan Chong. 2004. A cognitive hierarchy model of games. *The Quarterly Journal of Economics* 119, 3 (2004), 861–898.

[10] Caoyun Fan, Jindou Chen, Yaohui Jin, and Hao He. 2024. Can large language models serve as rational players in game theory? a systematic analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 17960–17967.

[11] Ian Gemp, Yoram Bachrach, Marc Lanctot, Roma Patel, Vibhavari Dasagi, Luke Marris, Georgios Piliouras, and Karl Tuyls. 2024. States as strings as strategies: Steering language models with game-theoretic solvers. *arXiv preprint arXiv:2402.01704* (2024).

[12] Uri Gneezy and Rann Smorodinsky. 2006. All-pay auctions—an experimental study. *Journal of Economic Behavior & Organization* 61, 2 (2006), 255–275.

[13] Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2023. News Summarization and Evaluation in the Era of GPT-3. arXiv:2209.12356 [cs.CL]

[14] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with Language Model is Planning with World Model. arXiv:2305.14992 [cs.CL]

[15] Nathan Herr, Fernando Acero, Roberta Raileanu, María Pérez-Ortiz, and Zhibin Li. 2024. Are Large Language Models Strategic Decision Makers? A Study of Performance and Bias in Two-Player Non-Zero-Sum Games. arXiv:2407.04467 [cs.AI] https://arxiv.org/abs/2407.04467

[16] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. [n.d.]. The Curious Case of Neural Text Degeneration. In *International Conference on Learning Representations*.

[17] Jie Huang and Kevin Chen-Chuan Chang. 2022. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403* (2022).

[18] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. 2022. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608* (2022).

[19] Pushmeet Kohli, Michael Kearns, Yoram Bachrach, Ralf Herbrich, David Stillwell, and Thore Graepel. 2012. Colonel Blotto on Facebook: The effect of social relations on strategic interaction. In *Proceedings of the 4th Annual ACM Web Science Conference*. 141–150.

[20] Vijay Krishna and John Morgan. 1997. An Analysis of the War of Attrition and the All-Pay Auction. *Journal of Economic Theory* 72, 2 (1997), 343–362. https://doi.org/10.1006/jeth.1996.2208

[21] Rolf Kümmerli, Caroline Colliard, Nicolas Fiechter, Blaise Petitpierre, Flavien Russier, and Laurent Keller. 2007. Human cooperation in social dilemmas: Comparing the Snowdrift game with the Prisoner's Dilemma. *Proceedings. Biological sciences / The Royal Society* 274 (09 2007), 2965–70. https://doi.org/10.1098/rspb.2007.0793

[22] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. 2017. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in Neural Information Processing Systems* 30 (2017).

[23] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2023. AgentBench: Evaluating LLMs as Agents. arXiv:2308.03688 [cs.AI]

[24] Yang Liu, Peng Sun, and Hang Li. 2024. Large Language Models as Agents in Two-Player Games. arXiv:2402.08078

[25] R. Duncan Luce and Howard Raiffa. 1957. *Games and decisions: Introduction and critical survey.* J. Wiley & Sons.

[26] Shaoguang Mao, Yuzhe Cai, Yan Xia, Wenshan Wu, Xun Wang, Fengyi Wang, Tao Ge, and Furu Wei. 2024. ALYMPICS: LLM Agents Meet Game Theory – Exploring Strategic Decision-Making with AI Agents. arXiv:2311.03220 [cs.CL] https://arxiv.org/abs/2311.03220

[27] H Brendan McMahan, Geoffrey J Gordon, and Avrim Blum. 2003. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 536–543.

[28] Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. 2023. Large Language Models as General Pattern Machines. arXiv:2307.04721 [cs.AI]

[29] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2024. A Comprehensive Overview of Large Language Models. arXiv:2307.06435 [cs.CL]

[30] Martin J Osborne. 1994. *A course in game theory.* MIT Press.

[31] Brian Roberson. 2006. The colonel blotto game. *Economic Theory* 29, 1 (2006), 1–24.

[32] Brian Roberson. 2006. The colonel blotto game. *Economic Theory* 29, 1 (Jan 2006), 1–24. https://doi.org/10.1007/s00199-005-0071-5

[33] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. arXiv:2302.04761 [cs.CL]

[34] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face. arXiv:2303.17580 [cs.CL]

[35] James WA Strachan, Dalila Albergo, Giulia Borghini, Oriana Pansardi, Eugenio Scaliti, Saurabh Gupta, Krati Saxena, Alessandro Rufo, Stefano Panzeri, Guido Manzi, et al. 2024. Testing theory of mind in large language models and humans. *Nature Human Behaviour* (2024), 1–11.

[36] Winnie Street, John Oliver Siy, Geoff Keeling, Adrien Baranes, Benjamin Barnett, Michael McKibben, Tatenda Kanyere, Alison Lentz, Blaise Aguera y Arcas, and Robin I. M. Dunbar. 2024. LLMs achieve adult human performance on higher-order theory of mind tasks. arXiv:2405.18870 [cs.AI]

[37] Weizhi Tang and Vaishak Belle. 2024. Zero, Finite, and Infinite Belief History of Theory of Mind Reasoning in Large Language Models. arXiv:2406.04800

[38] Yongqi Tong, Dawei Li, Sizhe Wang, Yujia Wang, Fei Teng, and Jingbo Shang. 2024. Can LLMs Learn from Previous Mistakes? Investigating LLMs' Errors to Boost for Reasoning. arXiv:2403.20046 [cs.CL]

[39] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, and Yasmine Babaei et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL]

[40] Tomer Ullman. 2023. Large Language Models Fail on Trivial Alterations to Theory-of-Mind Tasks. arXiv:2302.08399

[41] Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2022. Large language models still can't plan (a benchmark for LLMs on planning and reasoning about change). In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

[42] Max J. van Duijn, Bram M. A. van Dijk, Tom Kouwenhoven, Werner de Valk, Marco R. Spruit, and Peter van der Putten. 2023. Theory of Mind in Large Language Models: Examining Performance of 11 State-of-the-Art models vs. Children Aged 7-10 on Advanced Tests. arXiv:2310.20320

[43] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science* 18, 6 (2024), 186345.

[44] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. arXiv:2203.11171 [cs.CL]

[45] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903 [cs.CL]

[46] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

[47] James Wright and Kevin Leyton-Brown. 2010. Beyond equilibrium: Predicting human behavior in normal-form games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 24. 901–907.

[48] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models. arXiv:2303.04671 [cs.CV]

[49] Michael Wunder, Michael Kaisers, John Yaros, and Michael Littman. 2011. Using Iterated Reasoning to Predict Opponent Strategies. *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems* 2, 593–600.

[50] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864* (2023).

[51] Lin Xu, Zhiyuan Hu, Daquan Zhou, Hongyu Ren, Zhen Dong, Kurt Keutzer, See Kiong Ng, and Jiashi Feng. 2023. MAgIC: Investigation of Large Language

Model Powered Multi-Agent in Cognition, Adaptability, Rationality and Collaboration. arXiv:2311.08562

[52] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems* 36 (2024).

[53] Yadong Zhang, Shaoguang Mao, Tao Ge, Xun Wang, Yan Xia, Man Lan, and Furu Wei. 2024. K-Level Reasoning with Large Language Models. arXiv:2402.01521 [cs.CL]

[54] Pei Zhou, Aman Madaan, Srividya Pranavi Potharaju, Aditya Gupta, Kevin R McKee, Ari Holtzman, Jay Pujara, Xiang Ren, Swaroop Mishra, Aida Nematzadeh, et al. 2023. How FaR Are Large Language Models From Agents with Theory-of-Mind? *arXiv preprint arXiv:2310.03051* (2023).