Self-Consistency Enhanced Chess Agents for Full Chess Games

Abstract

Current LLM approaches rely on single-model prompting for next-move prediction. We evaluate self-consistency (SC) and multi-agent debate (MAD) frameworks—shown to enhance long-horizon reasoning—on chess puzzles and full games with explicit multi-move planning. SC substantially outperforms MAD (54% vs. 46% puzzle accuracy) with lower token costs. A round-robin tournament shows model quality trumps parameter count, reproducing GPT-3.5-Turbo-Instruct's unusual chess strength. Yet interpretability reveals only 12% board state grounding despite strong moves, suggesting pattern-matching over explicit reasoning. Our results show structured frameworks with planning enable competitive play, while questioning the link between move quality and true understanding.

1 Introduction

Chess serves as a gold standard for evaluating reasoning and planning in AI systems. Specialized engines like AlphaZero, Leela Chess Zero, and Stockfish have achieved superhuman performance for decades, yet millions of people continue to play, creating persistent research questions about human-like play, long-term strategic coherence, and model performance across architectures.

While recent work demonstrates transformer-based models achieving grandmaster-level chess performance (Ruoss et al., 2024; Zhang et al., 2025), most approaches focus on single move generation or traditional search augmentation. Key gaps remain in enhancing reasoning frameworks with paradigms like self-consistency (SC) and multi-agent debate (MAD) in long-horizon chess reasoning, systematic evaluation across open-source and proprietary models, and whether strong move selection reflects genuine board understanding or pattern matching.

We evaluate self-consistency (SC) and multi-agent debate (MAD) frameworks across 11 models spanning GPT, Llama, Qwen, Mistral, Deepseek, Gemma, and Arcee families. Contrary to expectations, we find SC substantially outperforms MAD while being more token-efficient. A round-robin tournament reveals model quality trumps parameter count, reproducing observations that GPT-3.5-Turbo-Instruct exhibits unusually strong chess performance. Interpretability analysis reveals models achieve strong moves with only 12% factual board grounding, suggesting pattern-matching dominates explicit reasoning.

2 Related Work

Reasoning Enhancement Frameworks

Chain-of-Thought prompting elicits explicit intermediate reasoning steps and has been shown to improve reasoning for complex questions by decomposing problems (Wei et al., 2023). Building on this, self-consistency aggregates multiple independently sampled reasoning traces and selects the most consistent answer, leading to gains in robustness and accuracy (Wang et al., 2023). Multi-agent debate further extends these ideas by allowing multiple agents to argue for competing thoughts, which combats fixation on wrong answers (Du et al., 2023; Liang et al., 2024).

LLM Chess Agents

Several recent works have adapted large language models to play chess, including Maia for human-like educational play (Tang et al., 2024). One line of work fine-tunes transformers directly on complete game records, showing that language-only models can reach strong amateur or master-level strength without external search (Ruoss et al., 2024; Zhang et al., 2025). Outside of formal publications, we systematically reproduce empirical results showing GPT-3.5-Turbo-Instruct remains unusually strong at chess—even against newer models—relative to a wide range of open-source and proprietary models of similar size, which hallucinate or degrade quickly (Carlini, 2023; Dynomight, 2023).

Evaluation Methodologies

For LLM-based agents, recent benchmarks such as Kaggle Game Arena use simulated tournaments together with Bradley–Terry style models to estimate relative skill from paired comparisons, yielding stable rankings even with incomplete match schedules (Kaggle, 2024; Bradley and Terry, 1952).

3 Methodology

3.1 Data Sources

We sample 1,000 puzzles across rating ranges of 600-2400 from the public Lichess database[1], which contains more than 5.4 million rated and tagged chess puzzles. Each puzzle contains a FEN (board state), solution moves in UCI format, player rating, popularity, and theme annotations. Puzzles are preferred for evaluation since they have a well-defined correct move, mitigating ambiguity during evaluation.

3.2 Models

We include 11 models from GPT, Deepseek, Mistral, Llama, Qwen, Gemma, and Arcee families (see Appendix A.1 for details). Among these, we primarily use GPT-3.5-Turbo-Instruct due to its robust performance and lower operational cost, serving as our main baseline for chess puzzle solving and move prediction. Since the open-source models are accessed through a third-party provider[2], which may apply quantization or other deployment-time modifications, we also evaluate additional closed-source GPT models from the same family (GPT-4o-mini and GPT-4.1-mini) to rule out provider-specific artifacts as the cause of the unusually large performance gap between GPT-3.5-Turbo-Instruct and open-source models.

3.3 Experimental Setup

Each model receives a board position immediately after the opponent's move and is tasked with predicting the next best move. The model predicts a move, and if the correct move is played, the following move is automatically played. The puzzle is marked as solved if all moves in the sequence are correctly predicted by the model. Conversely, if the model predicts an incorrect move at any point, the puzzle is marked as unsolved for that model.

We implement two primary multi-agent frameworks:

1. **Self-Consistency (SC)**: Three LLMs operate in parallel, blind to each other, prompted with divergent play-styles (aggressive, positional, neutral). Majority vote selects the move; Aggressive $>$ Positional $>$ Neutral priority for ties.

2. **Multi-Agent Debate (MAD)**: Two models (affirmative/negative) alternate proposing/critiquing moves for up to two rounds, followed by judge LLM selection. Early agreement truncates for efficiency.

Illegal moves are automatically filtered using the Python Chess API. If no legal move is extractable, the attempt is marked incorrect.

3.4 Full Game Evaluation

To evaluate chess agents in full games, single-agent, SC, and MAD setups play against Stockfish (strong baseline) at fixed depths 1-7, which searches 1-7 moves ahead. Using identical move generation prompts as puzzles ensures fair comparison across settings. Models receive three attempts to generate a legal move before disqualification.

Relative strength is assessed via round-robin tournament: each model plays every other 10 times (5 as white, 5 as black), including Stockfish at depth 5 baseline. Bradley-Terry estimation provides robust skill rankings from paired comparisons.

3.5 Planning Framework

To improve long-term coherence in full-game settings, we equip agents with an explicit planning module. At a high level, the agent generates multiple plans at each turn and uses the first moves to guide its next action. The agent follows one of the generated plans for as long as the opponent's responses remain consistent with it, and regenerates new plans when the game diverges. This approach applies to both single-agent play and our self-consistency setup. Implementation details are provided in Appendix A.4.

[1]https://database.lichess.org/#puzzles
[2]https://anannas.ai/dashboard

3.6 Metrics

Accuracy is reported both per-move (move accuracy) and by-puzzle (puzzle solved if every move is correct). Token costs, error rates, and error types (illegal/mismatch) are logged for all models and paradigms. We track illegal moves and inaccurate moves with respect to puzzle ELO, move number, and model size.

3.7 Interpretability

We evaluate the factual grounding of model-generated reasoning (here, GPT-3.5-turbo-instruct) to measure how closely the agent's explanations reflect the true board state. Our analysis addresses three questions: (1) Do reasoning patterns correlate with move correctness in chess puzzles? (2) How accurately do agents describe the board state? (3) Do agents execute the plans they express?

For 50 randomly sampled puzzles (SC agents solved 56% correctly), the model is explicitly instructed to articulate its reasoning before selecting the next move. Similar to Nguyen et al. (2024), we extract atomic factual claims from these reasoning traces (150 samples) using a regex + GPT-4o parser, targeting only claims that can be objectively validated against a single board state, such as piece locations (e.g., "knight on d3"), legal attacks/defenses, checks, captures, etc. Each claim is then verified against the current board state using *python-chess*, and only fully interpretable propositions are scored. We also check whether the model's stated plans match their executed moves, and assess correlations between grounding quality and puzzle/move accuracy.

4 Results

4.1 Puzzles

The GPT-3.5-Turbo-Instruct model utilizing SC achieves the highest puzzle accuracy at 54% and move accuracy at 52%, outperforming its single model and MAD counterparts, including newer models like (GPT-4.1-Mini and GPT-4o-Mini). (Table 3). Open source models performed poorly, with Deepseek-v3 with SC achieving only 8% accuracy.

Furthermore, GPT-3.5-Turbo-Instruct with SC outperforms the single model on puzzles rated between 1000 and 1799, demonstrating its ability to enhance move prediction reliability in moderately challenging scenarios.

Contrary to the findings reported in (Du et al., 2023), we observe that the MAD debate approach identifies the optimal move for puzzles less frequently than SC but outperforms a single model. No frameworks were able to solve the hardest puzzles from 2200+.

Model Size	**No Legal Move**	**Move Mismatch**
<10B	415 (69.2%)	185 (30.8%)
10-50B	45 (31.5%)	98 (68.5%)
50-100B	92 (32.3%)	193 (67.7%)
100-200B	35 (44.9%)	43 (55.1%)
200B+	114 (30.7%)	257 (69.3%)

Table 1: Error type distribution by model size (Chi-square test: p<0.001***).

4.2 Full Games with Planning

Enabling planning significantly improves full-game performance. Without planning, agents consistently lost to Stockfish at depth 5, and games tended to be longer and less stable. With planning, self-consistent agents defeated Stockfish at depth 5 in a 67-move game using two 4-ply plans per agent, and at depth 7 (ELO around 2033, roughly the top 1% of players) (Ferreira, 2013), with 41.2% and 36.4% of their moves, respectively, drawn from pre-generated continuations.

We also find most successful planned moves appear in the later stages of the game, where the reduced branching factor makes it more likely that the opponent follows one of the anticipated lines. This informed our adaptive planning strategy, which reduced token usage in the following games by more than one-third. Full game details, token usage metrics, and the implementation of adaptive planning are provided in Appendix A.1 and A.4.

4.3 Error Analysis

Comprehensive error analysis reveals a strong correlation between model size and performance metrics. Larger models tend to produce fewer errors, exhibiting a Pearson correlation coefficient of 0.525 (p = 0.0049) with puzzle accuracy. Similarly, puzzle rating negatively correlates with accuracy (r = -0.380, p = 0.0064), confirming that higher-rated puzzles present increased difficulty for language models. In terms of the type of errors made, smaller models more frequently fail by generating “no legal moves," whereas larger models exhibit a higher proportion of “move mismatch" errors (Table 1).

	Single	**SC**	**MAD**
Prompt Tokens	291.8	1224.1	2060.4
Completion Tokens	253.3	1180.9	1045.5
Total Tokens	545.1	2405.0	3105.9
Prompt Ratio	1.00	4.19	7.06
Completion Ratio	1.00	4.66	4.13
Total Ratio	1.00	4.41	5.70

Table 2: Token usage by paradigm (puzzle moves)

4.4 Token Usage and Efficiency

Both SC and MAD paradigms require substantially more prompt and completion tokens than the single model baseline. While SC outputs approximately 4.15 times more tokens than the single model, MAD requires 6.19 times as many tokens (Table 2).

4.5 Interpretability Analysis

Factual grounding is consistently low. GPT-3.5-Turbo-Instruct achieves a mean factual claim accuracy of 12.09% across all evaluated puzzles and agents. Model-generated reasoning frequently displays temporal confusion, as they reference incorrect board states or nonexistent pieces/threats. Plan-move alignment is near-zero: models almost never reliably describe the moves they ultimately play.

Grounding quality does not correlate with performance. Correlation between factual accuracy and puzzle and move correctness were negligible (Pearson correlation coefficient r = 0.0318 (claim accuracy vs. puzzle solved), r = 0.0335 (claim accuracy vs. move accuracy)). In the SC setting, the selected agent's reasoning was no more grounded than that of unselected ones, which indicates that majority-voted moves do not stem from more accurate internal representations.

However, since the claim extraction process cannot always parse the reasoning faithfully or interpret the model's statements in context, a more definitive assessment would require manually reviewing each reasoning trace and verifying all claims against the board.

5 Discussion

Interestingly, while SC consistently outperforms both single-model and MAD approaches across most puzzle difficulties, MAD shows a unique advantage in the 2000–2199 rating bin, the most challenging puzzles. This suggests debate mechanisms may offer benefits for particularly complex positions requiring deeper strategic synthesis, though at substantially higher costs.

Unlike prior debate work evaluating single midgame moves (Du et al., 2023), our diverse puzzle set spans all game phases and multi-move sequences, likely explaining SC's superiority in broader chess contexts.

Given the high overhead and marginal improvement that MAD offers over the single model, we recommend using SC for high-level chess play.

Full-game evaluations against Stockfish at multiple depths show that planning substantially improves performance, and even shallow lookahead can impose useful structure on long-term decision making. Planning also stabilizes move selection by grounding the current move within a longer anticipated sequence. Although planning is expensive up front, surviving plans amortize this cost, i.e., moves within an active plan no longer require regeneration. Games with planning consistently perform better and reach terminal positions in fewer plies. Furthermore, by enabling planning adaptively (when the branching factor is small), we significantly reduce overall token usage while retaining the benefits of structured lookahead.

Interpretability reveals agents achieve strong moves with only 12% factual board grounding, showing frequent temporal confusion and near-zero plan-move alignment. This aligns with evidence of LLM state-tracking failures (Hwang et al., 2025; Nguyen et al., 2024), suggesting chess performance stems primarily from pattern-matching rather than explicit reasoning.

6 Conclusion

Among 11 evaluated models spanning major families, GPT-3.5-Turbo-Instruct stands out for chess performance (54% puzzle accuracy with SC), while open-source models struggle despite larger scales. Self-consistency outperforms multi-agent debate with better efficiency, and planning enables master-level play (defeating Stockfish depth 7, ELO 2033). Yet interpretability reveals only 12% board state grounding despite effective moves, suggesting pattern-matching over explicit reasoning. This paper provides a systematic evaluation of reasoning strategies across model families, but understanding LLM reasoning in chess remains an open challenge.

References

R. A. Bradley and M. E. Terry. 1952. Rank analysis of incomplete block designs: The method of paired comparisons. *Biometrika*, 39(3–4):324–345.

Nicholas Carlini. 2023. Playing chess with large language models. `https://nicholas.carlini.com/writing/2023/chess-llm.html`. Accessed: 2025-10-15.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *Preprint*, arXiv:2305.14325.

Dynomight. 2023. More on chess and language models. `https://dynomight.net/more-chess/`. Accessed: 2025-10-15.

Diogo R. Ferreira. 2013. The impact of the search depth on chess playing strength. *ICGA Journal*, 36:67 – 80.

Dongyoon Hwang, Hojoon Lee, Jaegul Choo, Dongmin Park, and Jongho Park. 2025. Can large language models develop strategic reasoning? post-training insights from learning chess. *Preprint*, arXiv:2507.00726.

Kaggle. 2024. Kaggle game arena: Chess benchmark. `https://www.kaggle.com/game-arena`. Accessed: 2025-12-07.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. *Preprint*, arXiv:2305.19118.

Minh-Vuong Nguyen, Linhao Luo, Fatemeh Shiri, Dinh Phung, Yuan-Fang Li, Thuy-Trang Vu, and Gholamreza Haffari. 2024. Direct evaluation of chain-of-thought in multi-hop reasoning with knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2862–2883, Bangkok, Thailand. Association for Computational Linguistics.

Anian Ruoss, Grégoire Delétang, Sourabh Medapati, Jordi Grau-Moya, Li Kevin Wenliang, Elliot Catt, John Reid, and Tim Genewein. 2024. Grandmaster-level chess without search. *CoRR*, abs/2402.04494.

Zhenwei Tang, Difan Jiao, Reid McIlroy-Young, Jon Kleinberg, Siddhartha Sen, and Ashton Anderson. 2024. Maia-2: A unified model for human-ai alignment in chess. In *Advances in Neural Information Processing Systems*, volume 37, pages 20919–20944. Curran Associates, Inc.

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. *Preprint*, arXiv:2203.11171.

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. *Preprint*, arXiv:2201.11903.

Yinqi Zhang, Xintian Han, Haolong Li, Kedi Chen, and Shaohui Lin. 2025. Complete chess games enable llm become a chess master. *Preprint*, arXiv:2501.17186.

A Appendix

A.1 Models

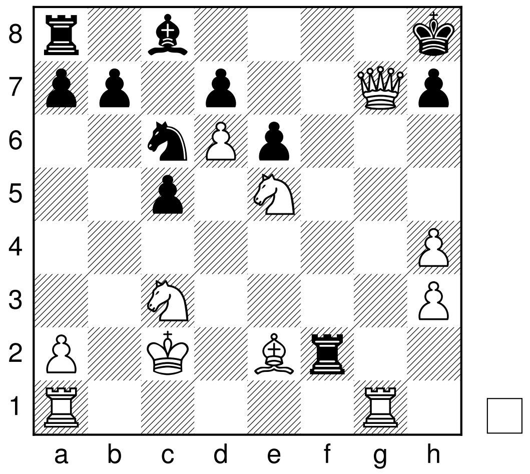
The GPT models are reached with the OpenAI API, and the open source models are reached with the Anannas API. It could be possible that Anannas has quantized the models, leading to worse performance. However, even within OpenAI models, GPT-3.5-Turbo-Instruct, and older, deprecated model, still seems to outperform newer models in chess.

- **GPT-3.5-Turbo-Instruct**
- **GPT-4o-mini**
- **GPT-4.1-mini**
- **deepseek-ai/deepseek-v3** (67B)
- **mistralai/mistral-small-24b-instruct-2501** (24B)
- **meta-llama/llama-3.3-70b-instruct** (70B)
- **meta-llama/llama-3.1-8b-instruct** (8B)
- **qwen/qwen3-235b-a22b-instruct-2507** (235B)
- **arcee-ai/afm-4.5b** (4.5B)
- **google/gemma-2-2b-it** (2B)
- **together/google-gemma-3n-E4B-it** (4B)

A.2 Full Game PGNs

A.2.1 Game 1 (Single model vs Stockfish at depth 1 - LLM wins

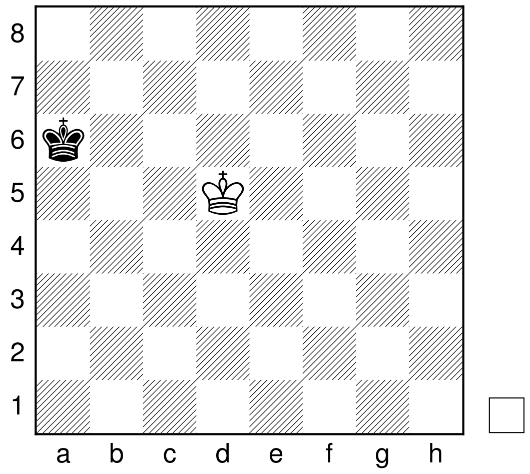
1 e4 c5 2 ♘f3 e6 3 d4 ♘f6 4 ♘c3 ♗e7 5 e5 ♘g4 6 h3 ♘h6 7 ♗×h6 g×h6 8 d5 O-O 9 d6 ♗h4 10 g3 ♕b6 11 g×h4 ♔h8 12 ♕d2 f6 13 ♕×h6 ♖f7 14 ♖g1 ♕×b2 15 ♔d2 f×e5 16 ♘×e5 ♕×c2+ 17 ♔×c2 ♖×f2+ 18 ♗e2 ♘c6 19 ♕g7# Z1-0



A.2.2 Game 2 (SC vs Stockfish at depth 4) - Draw

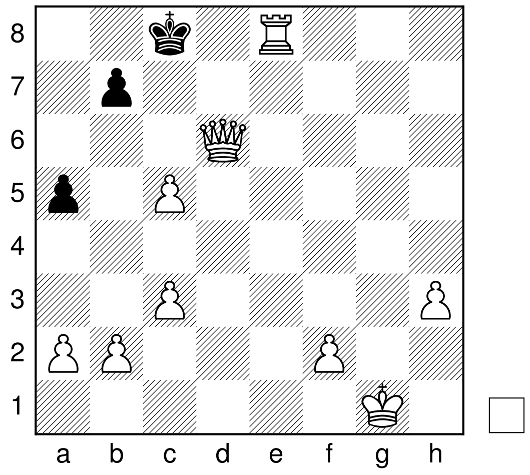
1 e4 ♘c6 2 d4 d5 3 ♘c3 d×e4 4 d5 ♘b8 5 ♘×e4 ♗f5 6 ♘g3 ♗c8 7 ♘f3 c6 8 ♗c4 g6 9 O-O ♘f6 10 d×c6 ♘×c6 11 ♕×d8+ ♘×d8 12 ♘e5 ♘d7 13 ♗b5 a6 14 ♗×d7+ ♗×d7 15 ♘×d7 ♔×d7 16 ♖d1+ ♔c8 17 ♘e4 ♗g7 18 ♗g5 ♗×b2 19 ♗×e7 ♗×a1 20 ♘d6+ ♔c7 21 ♖×a1 ♘c6 22 ♘×f7 ♖he8 23 ♗d6+ ♔d7 24 ♖d1 b5 25 ♗c5+ ♔e6 26 ♘g5+ ♔f5 27 ♖d5+ ♖e5 28 g4+ ♔×g5 29 ♗e3+ ♔h4 30 ♖×e5 ♘×e5 31 ♔g2 ♘×g4 32 ♗f4 ♔h5 33 ♔g3 ♖c8 34 h3 ♘e3 35 ♗×e3 ♖c4 36 a3 ♖c3 37 ♔f4 g5+ 38 ♔f5 a5 39 ♗×g5 a4 40 ♗e7 ♖f3+ 41 ♔e4 ♖×h3 42 f4 ♖c3 43 f5 ♖×c2 44 f6 ♔g6 45 ♔e5 h5 46 ♔e6 ♖e2+ 47 ♔d7 ♔f7 48 ♔c6 ♖×e7 49 f×e7 ♔×e7 50 ♔×b5 ♔e6 51 ♔×a4 ♔d7 52 ♔b5 ♔d8 53 a4 ♔c7 54 a5 ♔b7 55 a6+ ♔b8 56 ♔c4 h4 57 ♔d3 h3 58 ♔e2 h2 59 ♔f2 h1♕ 60 a7+ ♔×a7 61 ♔e3 ♕h2 62 ♔d4 ♕c7 63 ♔d5 ♕a5+ 64 ♔c6 ♕d8 65 ♔c5 ♕e7+ 66 ♔c6 ♕d8 67 ♔c5 ♕h4 68 ♔d5

♕h2 69 ♔c6 ♕h3 70 ♔d5 ♕h6 71 ♔c5 ♔b7 72 ♔d5 ♕h1+ 73 ♔c5 ♕h6 74 ♔d4 ♕h8+ 75 ♔d5 ♔a6 76 ♔c6 ♕d8 77 ♔c5 ♕a5+ 78 ♔c6 ♕c3+ 79 ♔d5 ♕d2+ 80 ♔c4 ♕g2 81 ♔d4 ♕g7+ 82 ♔c5 ♕g2 83 ♔d6 ♕g3+ 84 ♔c5 ♕g7 85 ♔d6 ♕g2 86 ♔c5 ♕g1+ 87 ♔c6 ♕g2+ 88 ♔c7 ♕g3+ 89 ♔c6 ♕g2+ 90 ♔c5 ♕c2+ 91 ♔d4 ♕c8 92 ♔d5 ♕g8+ 93 ♔c6 ♕d8 94 ♔c5 ♕g5+ 95 ♔c6 ♕h5 96 ♔d6 ♔a5 97 ♔c6 ♕f5 98 ♔d6 ♕h5 99 ♔c6 ♕f5 100 ♔d6 ♕f2 101 ♔c6 ♕b6+ 102 ♔d5 ♕b2 103 ♔c4 ♕c2+ 104 ♔d5 ♕h2 105 ♔c4 ♔a6 106 ♔c5 ♕g2 107 ♔c4 ♕b7 108 ♔c5 ♕h7 109 ♔c4 ♕d7 110 ♔c5 ♕d5+ 111 ♔×d5 Z1/2-1/2



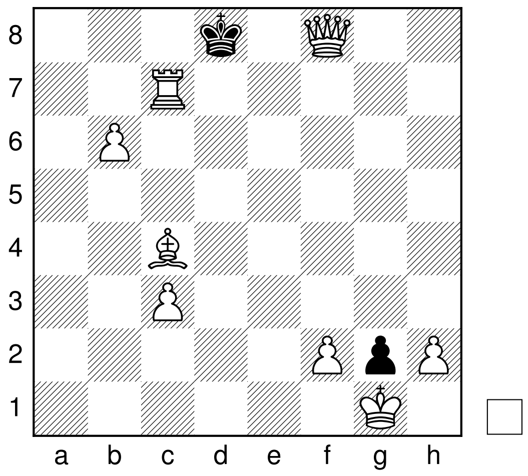
A.2.3 Game 3 (SC with planning vs Stockfish at depth 5) - LLM wins

1 e4 e5 2 ♘f3 ♘f6 3 ♘×e5 d6 4 ♘f3 ♘×e4 5 d4 ♗g4 6 ♗d3 ♕e7 7 O-O ♘c6 8 ♖e1 f5 9 ♘c3 O-O-O 10 ♘×e4 f×e4 11 ♗×e4 d5 12 ♗f5+ ♗×f5 13 ♖×e7 ♗×e7 14 c3 ♖hf8 15 ♗g5 ♗g4 16 ♗×e7 ♘×e7 17 ♕e2 ♘f5 18 ♕e6+ ♔b8 19 ♘e5 ♖d6 20 ♘d7+ ♖×d7 21 ♕×d7 a6 22 ♕×d5 h5 23 h3 ♗×h3 24 g×h3 ♘h4 25 ♕×h5 ♔c8 26 ♕×h4 c5 27 d×c5 g5 28 ♕×g5 ♖f7 29 ♕g8+ ♔c7 30 ♕×f7+ ♔c6 31 ♕e6+ ♔c7 32 ♕d6+ ♔c8 33 ♖e1 a5 34 ♖e8# Z1-0



A.2.4 Game 4 (SC with planning vs Stockfish at depth 7) - LLM wins

1 e4 e5 2 ♘f3 ♘f6 3 ♘×e5 ♘×e4 4 ♕e2 f6 5 ♕h5+ ♔e7 6 ♕f7+ ♔d6 7 ♘c4+ ♔c6 8 ♘a5+ ♔d6 9 d3 c6 10 ♗f4+ ♔c5 11 ♕c4+ ♔b6 12 ♕b3+ ♔×a5 13 d×e4 b5 14 a4 ♔b6 15 a×b5 c5 16 ♘c3 a5 17 b×a6+ ♔a7 18 ♘b5+ ♔b6 19 ♘d6+ ♔a7 20 ♘×c8+ ♕×c8 21 ♗×b8+ ♖×b8 22 ♕d5 c4 23 ♗×c4 ♖b4 24 b3 ♗c5 25 O-O ♖e8 26 c3 ♖b8 27 b4 ♗b6 28 ♖fd1 ♕c6 29 ♕×d7+ ♕c7 30 ♕×c7+ ♗×c7 31 ♖d7 ♔b6 32 a7 ♖a8 33 ♖a6+ ♔b7 34 b5 ♔c8 35 ♖×g7 f5 36 ♖c6 ♖e7 37 ♖×e7 ♖×a7 38 b6 ♔d8 39 ♖c×c7 ♖×c7 40 ♖×c7 h5 41 e×f5 h4 42 f6 h3 43 f7 h×g2 44 f8♕# Z1-0



A.3 Additional Figures

Model	Single	SC	MAD
Afm 4.5B	0.00	0.00	0.00
Deepseek V3	6.00	8.00	8.00
Gemma 2 2B It	0.00	0.00	0.00
Gpt 3.5 Turbo Instruct	*44.00*	**54.00**	46.00
Gpt 4.1 Mini	30.00	38.00	32.00
Gpt 4o Mini	14.00	20.00	16.00
Llama 3.1 8B Instruct	0.00	0.00	0.00
Llama 3.3 70B Instruct	2.00	4.00	2.00
Mistral Small 24B Instruct 250	4.00	4.00	6.00
Qwen3 235B A22B Instruct 2507	2.00	2.00	4.00
Google Gemma 3N E4B It	0.00	0.00	0.00

Table 3: Puzzle accuracy (%) by model and paradigm.

Model	Single	SC	MAD
Afm 4.5B	0.80	0.80	0.80
Deepseek V3	10.40	10.40	12.80
Gemma 2 2B It	0.80	1.60	0.80
Gpt 3.5 Turbo Instruct	41.60	**52.00**	46.40
Gpt 4.1 Mini	37.60	*42.40*	35.20
Gpt 4o Mini	21.60	23.20	20.80
Llama 3.1 8B Instruct	1.60	0.00	2.40
Llama 3.3 70B Instruct	3.20	5.60	4.00
Mistral Small 24B Instruct 250	4.00	3.20	4.80
Qwen3 235B A22B Instruct 2507	5.60	5.60	10.40
Google Gemma 3N E4B It	0.00	1.60	0.00

Table 4: Move accuracy (%) by model and paradigm.

A.4 Move Extraction and Validation

Because LLMs often generate extraneous text with partial annotations or descriptive comments, we implement a parser to extract Standard Algebraic Notation (SAN) moves. The parser prioritizes:

1. explicit labeled predictions like "Predicted move SAN: Qe1#"
2. SAN immediately followed by game result (e.g. "Qe1# 0-1")
3. explicit numbered moves for the provided current_turn_number ("22..." or "22.")
4. SANs whose nearest preceding move-number $\leq$ current_turn_number
5. ellipsis-based heuristics and safe fallbacks

One thing we may try is using an LLM as a judge to extract the predicted moves. This can handle unexpected output formats, especially from models that are smaller and less predictable.

A.5 Detailed Implementation of Planning

LLMs often produce speculative multi-move continuations when asked for a single decision. Rather than discarding these, we treat them as short-horizon plans that the agent can follow as long as the game remains consistent with the predicted line. This mirrors human decision-making, where candidate lines and expected replies guide move selection. This section outlines how plans are generated, executed, configured in our experiments, and adapted to reduce token usage.

A.5.1 Plan Generation and Execution

The agents generate several independent multi-ply continuations at each decision point and select the move that appears most frequently as the first step of these plans. Although any individual plan may be overly optimistic or based on a misread of the position, agreement across multiple plans provides a more stable and reliable choice. Once the majority-selected first move is identified, the system may request additional continuations that all begin with that move. During the game, the agent follows an active plan until the sequence ends or becomes invalid due to an unforeseen opponent move, at which point a new batch of plans is generated from the current position.

Planning is integrated into the SC setup by having each agent contribute its own set of plans, resulting in a mix of tactical and strategic perspectives. The final move and continuation are determined by whichever move the agents implicitly agree upon.

A.5.2 Adaptive Planning

On analyzing multiple 2-plan and 4-plan full games against Stockfish at depths 5-7, we observe that planned moves are used far less frequently in the early game, where the branching factor is high and predicted opponent responses rarely materialize. As a result, early plans are often broken and provide little benefit relative to their token cost. To address this, we use an adaptive planning strategy: planning is disabled while the board contains at least a certain number of pieces and enabled once the piece count falls below this threshold. This concentrates planning in the parts of the game where continuations are more stable and the likelihood of multi-ply plan follow-through is higher.

In our experiments, adaptive planning reduces overall token usage by more than one-third while retaining the benefits of structured lookahead. The results across planning configurations are summarized in Table 5, which shows that adaptive planning remains competitive with the standard planning approach without excessive token overhead.

Setting	Plans	Stockfish depth	Length	% Planned	Tokens	Result
No Planning	0	5	294	0%	231041	0-1
Planning enabled	2 plans	5	71	41.7%	263411	1-0
Planning enabled	3 plans	7	99	26%	429019	1-0
Planning enabled	4 plans	7	83	33.3%	478143	1-0

Table 5: Comparison of adaptive planning-enabled and no-planning games against Stockfish, in terms of total tokens used. Plans use a 4-ply depth, with planning activated once the board contains fewer than 24 pieces.