TEXTCRAFT: A TEXT-NATIVE STRATEGY BENCHMARK AND TRAINING GROUND FOR LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Most language-model benchmarks are single-player: models are evaluated against fixed tasks that neither adapt as agents improve nor remain discriminative once performance saturates. Multi-agent competition offers a natural alternative, but existing environments are largely inherited from games designed for humans, where perception and interface complexity can obscure the language-model capabilities of interest. We introduce TEXTCRAFT, an open real-time strategy environment (tick-based, simultaneous-move) designed natively for language-model agents. Observations, communication, and actions are expressed entirely in text, while agents must simultaneously manage an economy, a technology tree, and military forces under partial observability across hundreds of sequential decisions. TEXTCRAFT supports both 1v1 competition and team-based 2v2 play, requiring long-horizon planning, decision-making under uncertainty, adaptation to an opponent, and coordination with a partner. Because difficulty is induced by other improving agents rather than a fixed answer key, TEXTCRAFT has no fixed saturation point and can remain discriminative as model capabilities advance. We release the environment, scripted baselines, and complete game trajectories from a range of open and closed models, enabling analysis of how language-model agents strategize, use partial information, adapt to opponents, and coordinate with teammates. TextCraft also serves as a testbed for improvement rather than measurement alone: With learned natural-language skills and no parameter updates, Claude Haiku 4.5 and Luna 5.6, two lower-ranked models on the leaderboard, achieve competitive win rates against stronger models at the final checkpoint under tournament configurations. Together, these results position TEXTCRAFT as a shared environment for measuring and improving long-horizon strategic and multi-agent behavior in language models.

All interactive replays and full traces available here:

https://textcraft-gxtmmvv2ca-uc.a.run.app/

1 INTRODUCTION

Most language-model benchmarks evaluate agents against fixed tasks that reveal relatively little about how models behave when the environment responds strategically to their decisions. In particular, they do not directly measure whether an agent can anticipate an opponent, adapt its strategy over hundreds of decisions, or coordinate with another agent under incomplete information. Moreover, as models improve, fixed benchmarks can saturate, limiting their ability to distinguish increasingly capable agents.

Competitive games offer a complementary evaluation paradigm where the opponent itself creates an adaptive source of difficulty. In this setting, head-to-head matches provide direct comparisons between models, and complete game trajectories show how an agent plans, adapts, and fails. However, existing strategy games were primarily designed for human players. Their complex interfaces and low-level mechanics can obscure the language-model capabilities we wish to study. Existing LLM interfaces to StarCraft II, for example, either delegate much of the tactical control to scripts or expose a low-level action space that is difficult for language models to manage (Ma et al., 2023; Shao et al., 2024; Li et al., 2024).

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

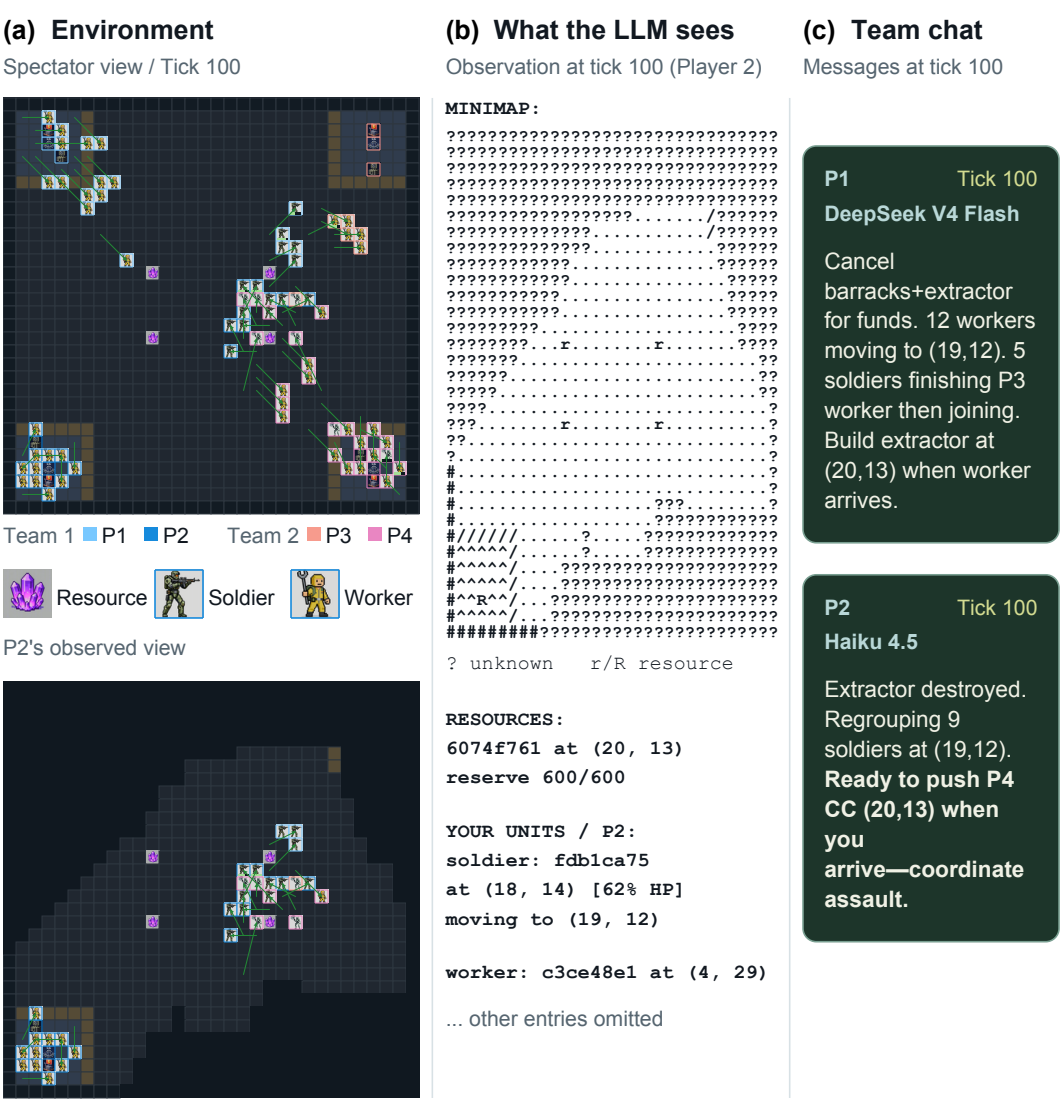


Figure 1: **TEXTCRAFT combines economy, combat, planning under partial observation and team coordination through text observations, language messages, and structured actions.** (a) The full spectator state and P2's observed view at tick 100, with sprite examples. (b) P2's exact ASCII terrain map and selected resource and unit entries; ? marks unknown terrain. Models receive text only; omitted entries are marked. (c) Team-chat messages at tick 100, in which Haiku 4.5 and DeepSeek V4 Flash coordinate an assault. Replays and full traces for all 3,615 games are available online.

We introduce TEXTCRAFT: an open, text-native real-time strategy environment designed to study long-horizon strategic reasoning in language-model agents. Players develop an economy, research technologies, and command armies on a 32×32 grid under fog of war. TEXTCRAFT provides players with a text observation at each game tick, and all players respond simultaneously with structured orders. Successful play requires balancing economic growth against immediate military threats, gathering information about an opponent, and adapting army composition and positioning as the game unfolds. In addition to 1v1 competition, TEXTCRAFT supports 2v2 matches in which allies have private observations and can coordinate through natural-language messages (Figure 1).

We designed TEXTCRAFT as an experimental platform. All models use a common, instrumented agent scaffold whose affordances, including extended thinking, persistent scratchpads, history, and communication, can be varied. A standardized tournament protocol enables direct model comparisons

(§4), while complete game trajectories support behavioral analysis beyond win rates (§4.2). Because gameplay is text-native and computationally lightweight, hundreds of matches between frontier models are practical. We use this platform to investigate four questions: **(1)** How well do current models perform at long-horizon strategy? **(2)** Which agent affordances improve performance? **(3)** Can experience be distilled into reusable strategies without updating model weights? **(4)** And when does communication improve cooperation?

Contributions. Our contributions are fourfold.

1. **A competitive benchmark for strategic language agents.** We develop a text-native strategy environment with economic planning, tactical combat, partial observability, and team coordination. It includes five two-player maps, four scripted opponents, a standardized agent scaffold, and a side-swapped tournament protocol (§3–§4). We release the environment and complete game trajectories.
2. **An empirical study of strategic capabilities.** We evaluate 18 models in a 900-game tournament, measuring playing strength, inference cost, and strategic behavior (§4.2). Controlled comparisons show that the highest tested thinking level is selected for 14 of 18 models, whereas persistent scratchpads are selected for only three (Appendices B.1 and B.2), suggesting that higher test-time compute works better than remembering previous intentions. Behavioral analysis identifies distinct patterns of economic development, army composition, and combat associated with successful play (§4.2).
3. **Strategic skill learning without gradient updates.** We introduce a method that distills game experience into natural-language strategy handbooks. Learned natural-language skills against bots improve Haiku's and Luna's win rates by at least 40 percentage points against stronger, previously unseen opponents, Sonnet and Sol, without weight updates. (§5).
4. **An evaluation of language-mediated cooperation.** We study 72 mixed-model 2v2 games in which one team can communicate and the other cannot. With private observations and otherwise matched settings, communication improves outcomes for some model combinations but degrades them for others, revealing that access to a communication channel does not by itself ensure effective coordination (§6).

2 Related Work

Games for language agents. Text worlds, board and card games, and social-deduction benchmarks test reasoning and interaction in game settings (Wu et al., 2023; Paglieri et al., 2024; Hu et al., 2025; Costarelli et al., 2024; Guertler et al., 2025; Xu et al., 2023; Light et al., 2023; Yao et al., 2025). These benchmarks span single-agent tasks and multiplayer games, including games with hidden information and simultaneous actions. StarCraft II (Vinyals et al., 2017; Samvelyan et al., 2019; Vinyals et al., 2019) and MicroRTS (Ontañón, 2013) pose complex real-time challenges. Prior LLM interfaces differ in the granularity of control they expose: TextStarCraft II delegates micro-control to scripts (Ma et al., 2023), while SwarmBrain combines LLM-directed group commands with rule-based unit behavior (Shao et al., 2024). LLM-PySC2 instead exposes the full low-level action space, including fine-grained unit operations (Li et al., 2024). TextCraft gives the language model explicit control over per-unit target selection and grid-level movement through a compact textual interface, while the engine handles movement and combat execution. This design makes tactical decisions directly attributable to the agent while simplifying the interface through which those decisions are expressed. The StarCraft II studies emphasize built-in-AI and task-specific evaluations, although TextStarCraft II also reports human matches (Ma et al., 2023); our evaluation centers on competition between language-model agents. TextCraft combines three properties in a text-native RTS: moves are simultaneous and tick-based, so deliberation competes with the orders a model must emit in the same turn; the game is partially observable with a long horizon; and its 2v2 mode combines cooperation and competition in the same match. Its head-to-head evaluation follows the arena paradigm (Chiang et al., 2024; Elo, 1978; Bradley & Terry, 1952).

Reasoning, memory, and learning. Chain-of-thought prompting can improve static reasoning tasks (Wei et al., 2022), although deliberation also has a cost in agentic settings (Cuadron et al., 2025). Agent memories and skill libraries preserve useful experience across decisions or episodes (Shinn et al., 2023; Park et al., 2023; Wang et al., 2023; Zhao et al., 2023; Wang et al., 2024). We separately

test thinking and within-game scratchpads. A related line uses language models as optimizers over prompts and programs (Yang et al., 2023; Fernando et al., 2023; Yuksekgonul et al., 2024; Madaan et al., 2023), including coding agents that improve their own scaffolds or discover algorithms (Zhang et al., 2025; Novikov et al., 2025). Related implementation tools include coding agents such as Codex CLI (OpenAI, 2025) and packaged agent instructions such as Agent Skills (Anthropic, 2025). Inspired by reusable behavior handbooks (Didolkar et al., 2025), our skill teacher reads complete game records and rewrites a natural-language strategy handbook, with no weight updates.

Cooperation and communication. Hanabi studies coordination under hidden information (Bard et al., 2020), while Overcooked studies coordination with human partners (Carroll et al., 2019). Diplomacy combines communication and strategic competition (Meta Fundamental AI Research Diplomacy Team (FAIR) et al., 2022); emergent communication also has a long history in multi-agent learning (Lazaridou & Baroni, 2020). TEXTCRAFT brings free-text team communication into a competitive strategy game where allies must coordinate despite seeing different parts of the map, letting us measure when communication pays for its bandwidth.

3 THE TEXTCRAFT ENVIRONMENT

TEXTCRAFT is a text-native strategy game on a 32×32 grid for two players or two teams of two. It probes whether LLMs can reason strategically: weighing trade-offs, adapting to new information, and responding to an opponent's shifting tactics; under resource constraints and incomplete information.

Observations, orders, and objectives. Players compete to destroy the opposing team's command centres. At each discrete *tick*, every player receives a structured text observation and submits a JSON array of orders. The world advances after all players have acted. Orders persist until completed or replaced, so a player need not repeat every command at every tick. Observations include an ASCII map, visible units and buildings, resources, production queues, and feedback on earlier actions.

The action space covers resource gathering, construction, repair, unit production, research, movement, and combat. The engine resolves movement and damage, but it does not choose a player's strategy. In particular, units only attack explicitly assigned targets. Players must decide where to move, which enemies to engage, and when to revise ongoing orders. Games end when one side loses all command centres or surrenders, or when the tick limit is reached (which results in a draw).

Balancing economy with survival. Workers gather finite resources, build structures, and repair damage. More workers, better infrastructure, and new resource sites raise future income, but every investment delays military production. Resource depletion and travel distance also change which expansions are worth capturing and protecting. Because gathering, construction, and combat proceed together, players must sustain an economy without leaving themselves exposed.

Partial observability in fog of war. Players see only within their units' and buildings' vision ranges, a restriction known as *fog of war*. Unexplored terrain is unknown, and previously observed enemy entities remain as timestamped records that may no longer describe the current state. A player therefore rarely knows the opponent's current army or intentions. Scouting can reveal enemy production, new expansions, and approaching forces, but scouts may be lost. Players must decide when to trust stale records, act on incomplete evidence, and revise their plans as the map becomes clearer.

Unit composition and tactics. Three unit types have cyclic combat advantages. They also differ in range, mobility, damage, and durability. Terrain limits movement and vision, elevation improves combat effectiveness, and random damage adds uncertainty to engagements. Victory therefore depends on more than producing units: players must choose compositions, position forces, and assign targets while continuing to manage their economy.

Cooperation with allies. In 2v2 games, allies control separate economies and armies. Each sees only its own observation. Optional team chat lets allies share sightings and propose joint actions, but it does not automatically share vision or unit control. Cooperation therefore tests whether players can communicate relevant information and coordinate actions across separate resources and viewpoints.

Agent scaffold. All models share the same scaffold for rules, prompts, observations, and action feedback. Thinking settings, a persistent model-written scratchpad, history, team chat, skill hand-

books (§5), and decision frequency can be varied for controlled comparisons. Every match records observations, orders, messages, and game states.

Maps and scripted opponents. The environment includes five symmetric 1v1 maps that vary terrain and resource placement, three of which also have 2v2 versions. It also includes four scripted opponents (Random, Economy, Rush, and Defensive) that provide reference matches and training experience. Full game rules and map details appear in Appendix A.

4 Evaluating Strategic Performance

We evaluate how effectively language models play TextCraft and whether additional configuration (reasoning and persistent notes) improves their performance. Given that the performance depends on both a model and its configuration, our main evaluation has two stages: select the best configuration for each model, then compare the selected agents in a tournament. Separate experiments test skill learning and team communication.

4.1 Evaluation Protocol

Common scaffold and configuration selection. All 18 models receive the same rules, observation format, action schema, and feedback. The default prompt contains the current observation, not the complete history of earlier observations and orders. An optional scratchpad lets an agent retain its own notes between ticks.

We first compare each model's available thinking settings in head-to-head elimination matches, with scratchpads disabled (310 games). We then compare scratchpads on and off at the selected thinking level (180 games). This procedure makes configuration selection affordable, but it is sequential: we do not evaluate every thinking–scratchpad combination. Moreover, thinking labels differ across providers and do not imply equal compute budgets. The highest tested thinking setting is selected for 14 of 18 models, whereas a scratchpad is selected for only three. In these comparisons, greater reasoning effort is favoured more consistently than persistent notes. Table 9 lists the selected configurations; complete brackets appear in Appendices B.1 and B.2.

Tournament. The selected configurations compete in ten Swiss rounds. Models are paired by accumulated tournament points without repeated opponents. Each pairing plays ten games: one on each of the five symmetric maps from both starting positions. Player-relative coordinates and seat swaps reduce positional effects. Every model faces ten opponents and plays 100 games, producing 900 games overall. Matches have a 250-tick limit; wins earn one point and draws half a point.

4.2 Leaderboard and Behavioral Analysis

We rank the selected configurations by tournament record and Bradley–Terry rating (Bradley & Terry, 1952), which estimates relative strength while accounting for opponents faced, and pair each rating with the monetary cost per player tick (Figure 2). Costs combine recorded charges with estimates where necessary, and inference latency is excluded from game outcomes. The resulting leaderboard therefore compares models under their *selected scaffolds*, not under identical inference budgets. We then analyze economic, military, exploration, and order-level traces to characterize how models play. Rating calculations, cost accounting, complete records, and pairwise results appear in Appendix B.

Competitive performance and cost. GPT-6 Astra leads the 18-model comparison by Bradley–Terry rating, winning 96 of its 100 games. Fable 5.1, Opus 5, and Gemini 3.7 Flash follow in that order, with 82, 76, and 66 wins, respectively (Figure 2). Astra also has the highest mean cost, at $0.083 per player tick. Gemini 3.7 Flash, at $0.011 per tick, roughly a third of Opus 5's cost and a seventh of Astra's, ranks fourth and offers the best performance-cost trade-off. GLM 5.3 Flash has the lowest measured cost ($0.0019 per tick) yet ranks twelfth, ahead of six more expensive models, including GPT-5.6 Terra and DeepSeek V4 Pro.

Model behavioral analysis. Different models follow several distinct behavior profiles (Figure 3). We cluster (k-means, $k = 10$) 1,800 player-games using full-game summary and temporal features across several vital factors such as economy, production, composition, army, positioning, etc. and visualize this data in 2-D space with t-SNE (van der Maaten & Hinton, 2008). Scripted bots mostly lie outside the LLM reference range (Figure 14).

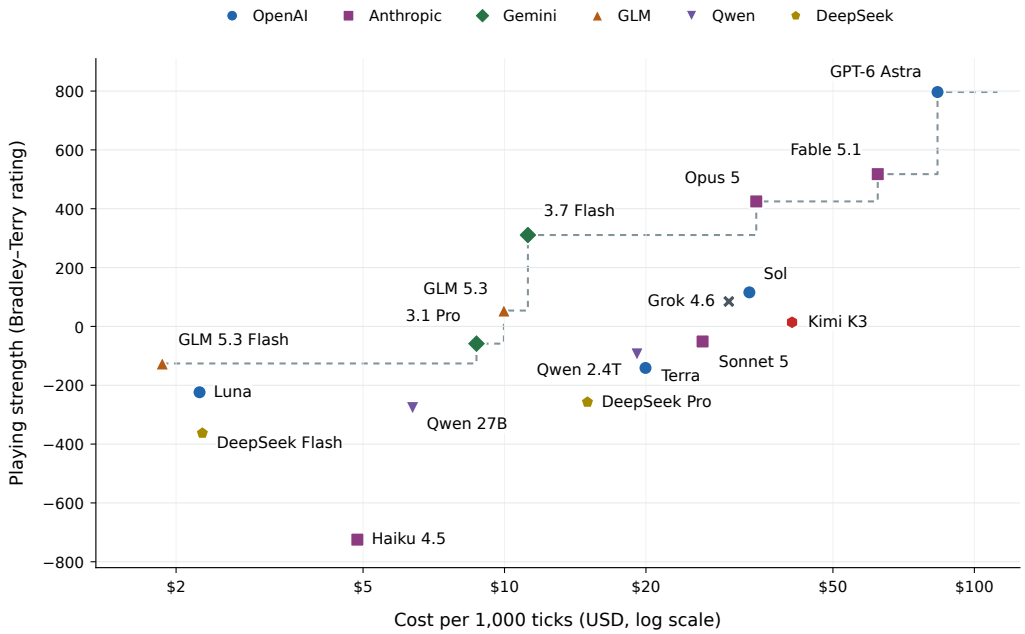


Figure 2: **Competitive performance and cost.** Zero-centered Bradley–Terry rating versus mean cost per 1,000 player ticks across 900 games, with 100 games per model. The cost axis is logarithmic; colors and shapes identify model families. The dashed staircase marks the observed Pareto frontier: the highest rating available at each cost budget.

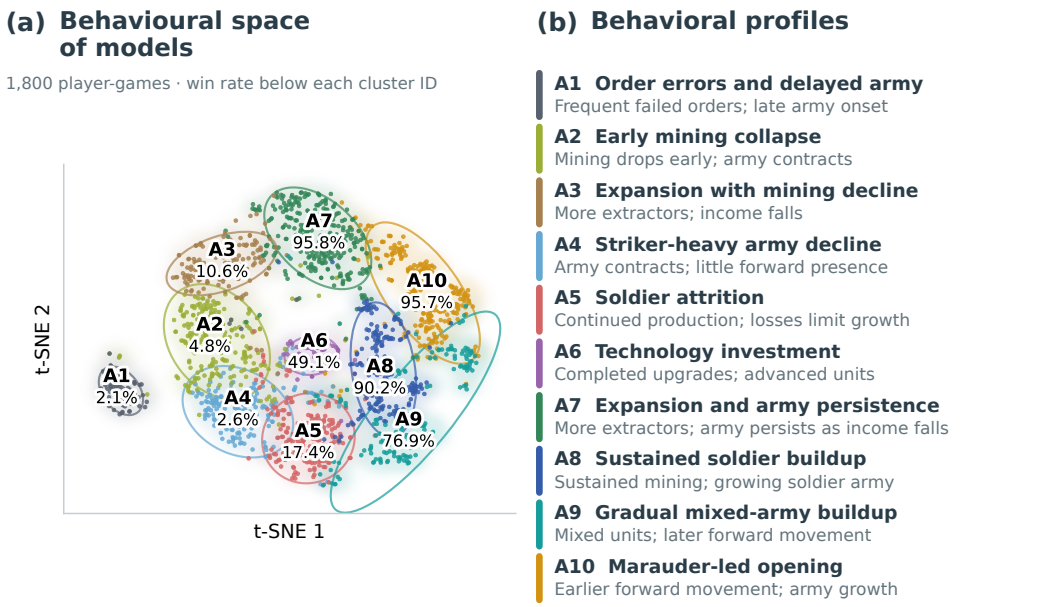
This yields ten behavioral profiles whose win rates vary, and models differ in how often they play each one (Figure 3b,c). Fable spreads most of its games across three profiles (A7–A9), whereas Astra plays 93% of its games in one, a marauder-led opening with sustained forward pressure (A10), and wins all of them. Opponents try to counter with strikers,[1] but the counter works only when well executed (Appendix E.4). Astra's four losses all begin with a similar opening: in three, Fable executes the counter well and breaks Astra's pressure (ending in A3 or A9); in the fourth, Gemini 3.7 Flash wins the marauder exchanges with high ground and focused fire (A2). The results thus show several routes to winning and no single optimal strategy, consistent with a balanced game, though other effective strategies may remain unexplored.

5 Skill Learning Without Weight Updates

TextCraft also lets us study how agents improve from experience. We test whether a language model can learn to play better from its own games, without weight updates, with the help of a stronger LLM teacher. The teacher reads the student's games and distills them into natural-language *skills*, which the student receives in its prompt in later games. The student learns these skills against scripted bots, and we evaluate whether they transfer to stronger language-model opponents that it never trained against.

Distilling games into a handbook. The student plays the four scripted bots in rotation, one game against each per generation, and GPT-5.6 Sol serves as the teacher. We use Haiku and Luna as students; they finish 18th and 14th of 18 in the tournament. After every generation, the teacher reads all of the student's games so far, anonymized and unordered, and writes a *skill handbook* from scratch. This handbook contains at most eight skills, each a named piece of natural-language advice with a trigger and an action. For example, Luna's `skill_opening_script` says to build an extractor on the nearest resource, keep producing and harvesting workers, and add a barracks at about eight workers (Appendix C.5). We rebuild rather than revise the handbook because a teacher that revises its previous handbook anchors on the latest games and regresses after early gains (Appendix C.1). The student receives the handbook with the rules, and it does use the skills, e.g: "Following `skill_opening_economy`, I need to reach 7–8 workers".

[1] Strikers deal +20% damage to marauders; marauders deal −20% to strikers.



(c) Where each model plays

Game share (%) · ordered by rank

		A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
01	GPT-6 Astra		1	1				1		4	93
02	Claude Fable 5.1		3	12	5			23	19	26	12
03	Claude Opus 5		5	8	10			25	4	18	30
04	Gemini 3.7 Flash		5	6	17	1		22		16	33
05	GPT-5.6 Sol		14	8	20	3		20		19	16
06	Grok 4.6		11	17	16	4		9	5	31	7
07	GLM 5.3		18	4	18	13		18	22	5	2
08	Kimi K3		11	13	21	8	13	15	16	2	1
09	Claude Sonnet 5		15	12	3	23	1	17	29		
10	Gemini 3.1 Pro		15	10	13	15	5	17	7	9	9
11	Qwen3.8-2.4T	1	16	10	15	11		15	21	1	10
12	GLM 5.3 Flash		13	16	10	13	15	20	10	2	1
13	GPT-5.6 Terra		15	9	14	3	13	23		6	17
14	GPT-5.6 Luna		32	14	8	9	5	17	10	1	4
15	DeepSeek V4 Pro	1	26	8	13	12	1	8	18	13	
16	Qwen3.8-27B		12	7	4	30		22	25		
17	DeepSeek V4 Flash	3	19	6	5	31		12	8	16	
18	Claude Haiku 4.5	92				8					

Figure 3: **Behavioral analysis across the leaderboard.** **(a)** t-SNE analysis on leaderboard games, colored by clusters fitted to full-game summary and temporal features. Percentages indicate cluster win rates. **(b)** Behavioral profiles describe common behavior patterns of clustered groups. **(c)** Behavioral profile distribution (%) of each model's 100 games; blank cells denote zero.

Both students improve against stronger models. Each student trains against the bots for twelve generations; we freeze its handbooks at G3, G6, G9, and G12 and play twenty games per checkpoint against Sonnet 5, Sol, and Opus 5, none of which receives a handbook (Figure 4, Appendix C.2). Evaluation games never reach the teacher. Both students improve most against the mid-range models. Haiku, which lost all ten of its tournament games against Sonnet, wins 95% of its G12 games against Sonnet without thinking, 55% against Sonnet with thinking, and 45–55% against Sol at G9–G12. Luna rises from 15% to 55–75% against Sonnet at G9–G12 and from 0% to 60% against Sol at G12. Against Opus, Haiku never wins and Luna wins two games.

Skills exploit the habits of mid-range models. A few skills can beat models far above the student on the leaderboard because mid-range models play with fixed habits and adapt little. Sonnet opens with soldiers in almost every game and keeps producing them, so a student that counters this habit

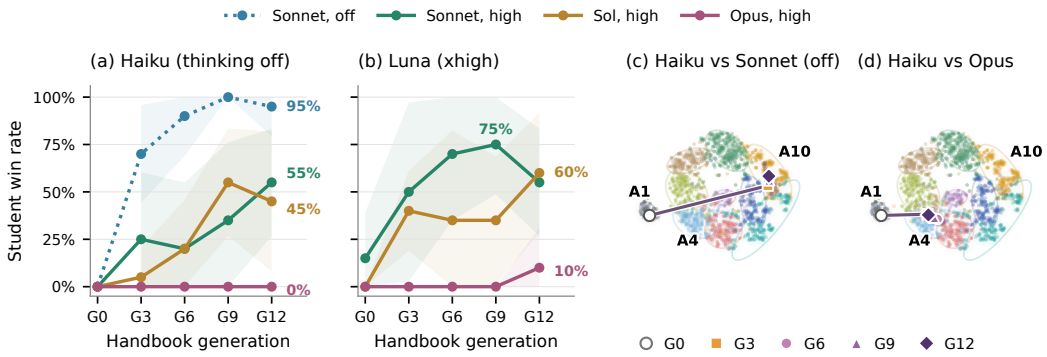


Figure 4: **Two students take different lessons from the same curriculum.** (a) Haiku with thinking off and (b) Luna with xhigh reasoning against Sonnet 5, Sol, and Opus 5 with high reasoning, plus Sonnet without thinking for Haiku (dotted); twenty games per checkpoint, shading ± 1 SD across ten seed pairs. Both students use the configuration that won their tournament selection, Haiku without thinking and Luna at xhigh reasoning; enabling Haiku's thinking changes its Sonnet results by at most one game (Appendix C.3). (c–d) Haiku's games in the frozen Swiss embedding against different opponents; connected large markers are checkpoint geometric medians. Profiles describe realized behavior.

from the start keeps its advantage. The teacher records whatever beat the training opponent, so the training bot decides which counter the student learns (Appendix C.4). The soldier-only Defensive bot teaches marauders, which outrange and beat soldiers, and this lesson transfers: trained on that bot alone, Luna wins 63 of 80 games against a panel of eight LLMs. The marauder-only Economy bot teaches strikers. Luna eventually wins every training game against that bot but only 23 of 80 panel games, because the counter does not match what the LLM opponents field.

Two students take different lessons. Haiku's handbook converges on the marauder lesson, and its games shift accordingly: without a handbook, most fall in A1; with one, they move into A10, the marauder-led opening that Astra uses throughout the tournament, now without the failed orders. Luna's orders fail far less often, so its teacher spends fewer skills on order legality and formatting, as it must for Haiku, and more on strategy. Luna's handbook conditions production on the enemy army it observes, strikers against marauders and marauders against soldiers, and Luna follows it, answering Opus's marauders with strikers.

Skill issue. A good opening only goes so far against a model that adapts. Opus with high reasoning beats every Haiku handbook, and Haiku's games against it drift into A4 (striker-heavy army decline), a profile that loses almost every tournament game in which it appears. These games start with the same marauder opening, but once Opus takes it apart, Haiku has no answer, and in most of them Opus wins with a marauder opening of its own. Luna's adaptive handbook does slightly better, winning the only two games any student takes from high-reasoning Opus, both decided well after the opening. A weak model can copy the strategy of the strongest players but not the execution that makes it work, a gap we call a "skill issue".

6 When Does Communication Improve Cooperation?

In a 2v2 game, two teams of two players compete to destroy each other's command centres. Allies control separate economies and armies and see only their own observations, but they can exchange natural-language messages through a team chat (Section 3). This setting lets us test a different agent affordance: communication. Allies may benefit from sharing enemy sightings or coordinating attacks, but composing, reading, and acting on messages also adds work to each decision.

Matched chat and silent teams. We pit a chat-enabled team against a silent team with the same two-model roster and individual settings. Both sides have private observations, and scratchpads are disabled. Messages sent on one tick arrive on the next; the chat-enabled team can see messages from the previous ten completed ticks. The three rosters are Haiku 4.5 with DeepSeek V4 Flash (beginner), Sonnet 5 with Gemini 3.1 Pro (intermediate), and GPT-6 Astra with Fable 5.1 (expert). These names identify experimental groups; rosters never play one another. Each roster plays 24 games across the three four-player maps, both left/right assignments, and four within-team player orders. Seat permutations matter because four-player maps lack player-relative coordinates.

Table 1: Chat-enabled team against a silent team with the same roster (24 games per roster).

Roster	Models	Wins	Losses	Draws
Beginner	**Haiku 4.5** + **DeepSeek V4 Flash**	12	12	0
Intermediate	**Sonnet 5** + **Gemini 3.1 Pro**	15	8	1
Expert	**GPT-6 Astra** + **Fable 5.1**	6	18	0

Chat can help, have no effect, or hurt. Chat wins 15 of 24 intermediate games, 12 of 24 beginner games, and 6 of 24 expert games (Table 1). The chatting expert team loses all eight King-of-the-Hill games. With 24 games per roster, these results show that the effect depends on the tested team, not that chat systematically helps or hurts at a given level of capability. Outcomes alone also do not say why, so we read the chat logs alongside replays and paired coordination measures (Appendix F.4).

Chat helps when it leads to timely support. Sonnet and Gemini Pro mostly talk about forces they already have. Requests become reinforcements, and teammates sometimes divide pressure between opponents rather than focus on one target. Compared with its silent counterpart, the chatting team spends more time forward and in joint combat. The beginner roster uses the same channel less reliably: Haiku reports reinforcements before they exist, and DeepSeek Flash holds position to wait for them. Chat improves the beginner team's army balance but not its forward presence or joint combat, and its record stays even.

Agreement can reinforce a costly plan. Astra and Fable exchange precise tactical information, but agreement can lock them into an expensive plan. They endorse saving for additional command centres while their armies fall behind; by tick 50, the chatting expert team has produced less military than its silent opponent in 22 of 24 games. Coordinated assaults can also leave an ally's base exposed. Their wins instead keep military production going, revise expensive plans, and combine pressure with home defense. What matters is therefore not whether teammates agree, but whether their agreement supports actions they can sustain.

7 Limitations

Fixed scaffold. All models play through the same scaffold, with the same rules, prompts, observation format, and action interface (Section 3). This isolates differences between models, but it means we do not study how the scaffold itself affects performance. Some models may play better with a different prompt, observation format, or memory design, so the leaderboard ranks model–scaffold pairs, and a different scaffold could reorder them.

Engine exploits. We do not know whether stronger models can find exploits in the game engine. If they do, competition between those models could be decided by who exploits the engine better rather than by strategic play. We release all replays and traces so that such behavior can be detected.

Capped test-time compute. For budget reasons, we cap generation at 10,240 tokens per call, with up to 4,096 more for action recovery, and omit thinking levels that exceed this budget in more than 5% of calls (Appendix B). Our experiments therefore do not test the high-compute regime. Because the highest tested thinking setting is selected for 14 of 18 models, this cap may disadvantage models that keep improving with more test-time compute.

8 Conclusion

TextCraft provides a text-native environment for studying how language-model agents compete, cooperate, and improve through experience. The observed counterplay is consistent with a balanced game with substantial strategic depth, rather than one universally dominant strategy. Success depends not only on choosing a plan, but on executing it: balancing investment, army production, positioning, attack timing, and defense, while gathering information under partial observability. Natural-language skills distilled from experience help weaker models adopt strategies that beat stronger ones, though not the execution of the strongest. Team play shows that communication is not automatically effective cooperation: agents can agree on a costly plan despite detailed tactical exchanges, while simple, accurate reports and timely requests for support can sustain joint action. We release the environment, trajectories, and training tools to support further study of these capabilities.

Reproducibility Statement

The environment, scaffold, scripted opponents, tournament runner, teacher harness, all experiment drivers, match logs, replays, learned handbooks, and analysis scripts are provided as anonymised supplementary material. We also provide a website for interactively exploring replays and all game traces. The teacher runs a pinned Codex CLI (0.146.1) in an isolated home directory (Codex harness is open source). Appendix A lists all game constants and Table 12 gives the costs for the four main experiments only.

Ethics Statement

This work involves no human subjects, human annotators, or personal data. The environment is a fictional strategy game, and all game content is original to this work. The main resource cost is LLM API usage and compute; Table 12 reports API costs for the four main experiments. The learned skills are natural-language game advice, and we do not foresee dual-use or other harmful applications.

The Use of Large Language Models

Large language models are the object of study in this work: they play every game, and a coding agent (OpenAI Codex CLI running GPT-5.6 Solin Codex CLI version 0.146.1) acts as the skill teacher; both are described in §3 and §5.

Beyond this, we used generative AI tools in the following ways. (i) *Software implementation*: coding agents (Cursor, Claude Code and OpenAI Codex CLI) were used to implement parts of the game engine and the experiment drivers, including the ablation experiments. (ii) *Analysis and figures*: the same agents wrote the analysis scripts and the code used to generate the figures in the paper. (iii) *Writing*: LLMs were used for grammar and style corrections of the manuscript text. The research idea, hypotheses, and experimental design are entirely our own, as is the interpretation of results. We have reviewed all AI-assisted work: LLM-generated code and analysis scripts were checked and tested by the authors, and reported results were verified against the underlying experimental outputs. We take full responsibility for the final content of this work.

References

Anthropic. Introducing agent skills. https://claude.com/blog/skills, 2025. Accessed 2026-09-26.

Nolan Bard, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. The Hanabi challenge: A new frontier for AI research. *Artificial Intelligence*, 280:103216, 2020.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Micah Carroll, Rohin Shah, Mark K. Ho, Thomas L. Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-AI coordination. *arXiv preprint arXiv:1910.05789*, 2019.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating LLMs by human preference. *arXiv preprint arXiv:2403.04132*, 2024.

Anthony Costarelli, Mat Allen, Roman Hauksson, Grace Sodunke, Suhas Hariharan, Carlson Cheng, Wenjie Li, Joshua Clymer, and Arjun Yadav. GameBench: Evaluating strategic reasoning abilities of LLM agents. *arXiv preprint arXiv:2406.06613*, 2024.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, et al. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*, 2025.

Aniket Didolkar, Nicolas Ballas, Sanjeev Arora, and Anirudh Goyal. Metacognitive reuse: Turning recurring LLM reasoning into concise behaviors. *arXiv preprint arXiv:2509.13237*, 2025. URL https://arxiv.org/abs/2509.13237.

Arpad E. Elo. *The Rating of Chessplayers, Past and Present*. Arco Publishing, 1978.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*, 2023.

Leon Guertler, Bobby Cheng, Simon Yu, Bo Liu, Leshem Choshen, and Cheston Tan. TextArena. *arXiv preprint arXiv:2504.11442*, 2025.

Lanxiang Hu, Mingjia Huo, Yuxuan Zhang, Haoyang Yu, Eric P. Xing, Ion Stoica, Tajana Rosing, Haojian Jin, and Hao Zhang. lmgame-bench: How good are LLMs at playing games? *arXiv preprint arXiv:2505.15146*, 2025.

Angeliki Lazaridou and Marco Baroni. Emergent multi-agent communication in the deep learning era. *arXiv preprint arXiv:2006.02419*, 2020.

Zongyuan Li, Yanan Ni, Runnan Qi, Lumin Jiang, Chang Lu, Xiaojie Xu, Xiangbei Liu, Pengfei Li, Yunzheng Guo, Zhe Ma, et al. LLM-PySC2: StarCraft II learning environment for large language models. *arXiv preprint arXiv:2411.05348*, 2024.

Jonathan Light, Min Cai, Sheng Shen, and Ziniu Hu. AvalonBench: Evaluating LLMs playing the game of Avalon. *arXiv preprint arXiv:2310.05036*, 2023.

Weiyu Ma, Qirui Mi, Yongcheng Zeng, Xue Yan, Yuqiao Wu, Runji Lin, Haifeng Zhang, and Jun Wang. Large language models play StarCraft II: Benchmarks and a chain of summarization approach. *arXiv preprint arXiv:2312.11865*, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

Meta Fundamental AI Research Diplomacy Team (FAIR), Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, et al. Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Santiago Ontañón. The combinatorial multi-armed bandit problem and its application to real-time strategy games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pp. 58–64, 2013.

OpenAI. OpenAI Codex CLI. https://github.com/openai/codex, 2025. Accessed 2026-09-26.

Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wołczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, et al. BALROG: Benchmarking agentic LLM and VLM reasoning on games. *arXiv preprint arXiv:2411.13543*, 2024.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft multi-agent challenge. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2019.

Xiao Shao, Weifu Jiang, Fei Zuo, and Mengqing Liu. SwarmBrain: Embodied agent for real-time strategy game StarCraft II via large language models. *arXiv preprint arXiv:2401.17749*, 2024.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL https://www.jmlr.org/papers/v9/vandermaaten08a.html.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. StarCraft II: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Yue Wu, Xuan Tang, Tom M. Mitchell, and Yuanzhi Li. SmartPlay: A benchmark for LLMs as intelligent agents. *arXiv preprint arXiv:2310.01557*, 2023.

Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. Exploring large language models for communication games: An empirical study on Werewolf. *arXiv preprint arXiv:2309.04658*, 2023.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

Jianzhu Yao, Kevin Wang, Ryan Hsieh, Haisu Zhou, Tianqing Zou, Zerui Cheng, Zhangyang Wang, and Pramod Viswanath. SPIN-Bench: How well do LLMs plan strategically and reason socially? *arXiv preprint arXiv:2503.12349*, 2025.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. TextGrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. *arXiv preprint arXiv:2308.10144*, 2023.

A GAME PARAMETERS

Win condition. All main evaluation maps use command-centre destruction: a player or team wins by eliminating all opposing command centres while retaining at least one of its own. In 2v2, this requires eliminating the command centres of both opposing players. Simultaneous elimination of both sides, or reaching the tick limit without a winner, results in a draw.

Table 2: Unit statistics. Cooldown is ticks between attacks. The tank deals 3×3 splash damage and cannot fire at range < 3.

Unit	HP	Dmg	Range	Speed	Vision	CD	Cost	Build	Tier
worker	25	–	–	2	2	–	25	2	0
scout	20	3	1	4	6	2	25	2	1
soldier	40	10	2	2	3	2	50	3	1
marauder	30	15	4	2	4	2	60	3	1
striker	35	8	1	3	3	2	40	2	1
enforcer	80	15	1	1	3	3	100	4	2
tank	45	12	6	1	5	4	150	5	2

Table 3: Damage modifiers among tier-1 infantry: attacker (row) $\rightarrow$ defender (column), applied to units only.

	vs. soldier	vs. marauder	vs. striker
soldier	$+0\%$	-40%	$+30\%$
marauder	$+40\%$	$+0\%$	-20%
striker	-30%	$+20\%$	$+0\%$

Table 4: Structures and upgrades. Upgrade/ability entries give resource cost and effect.

Structure	HP	Cost	Build	Vision	Prerequisite	Upgrades / ability (cost, effect)
command centre	500	400	20	4	–	–
extractor	150	100	5	2	–	–
barracks	200	150	8	3	–	–
tech lab	180	200	10	2	barracks	optics (100, +2 vision); plating (150, +15% HP)
machine shop	180	200	10	2	barracks	boosters (100, +1 speed); ordnance (150, +15% damage)
weapons lab	200	500	15	2	tech lab + machine shop	Orbital strike (300, 200 damage in a 3×3 area, 5-tick countdown, 30-tick cooldown)

Table 5: Other game constants and evaluation limits.

Parameter	Value
Damage variance	$\pm 25\%$
High-ground bonus	$+25\%$
Worker carry capacity	8
Starting resources	200
Starting workers	2 per command centre
Cancellation refund	75%
Worker repair	5 HP/tick at 1 resource/tick
Engine hard tick cap	500
Main evaluation tick limit	250

Table 6: Research times and application rules.

Upgrade	Research structure	Time (ticks)
Optics	Tech lab	10
Plating	Tech lab	12
Boosters	Machine shop	10
Ordnance	Machine shop	12
Plating applies only to newly produced units. Upgraded HP, base damage and cancellation refunds are truncated to integers.		

Distances. Movement permits eight directions, subject to terrain restrictions. Attack range and vision use Chebyshev distance, $d = \max(|\Delta x|, |\Delta y|)$, so an adjacent diagonal cell is distance 1.

Evaluation maps. Table 7 lists the initial resource reserves for each main evaluation map.

Table 7: Initial resource reserves for each main evaluation map. All maps are 32×32. Reserve entries give the number of nodes multiplied by resources per node; totals exclude players' starting resources.

Map	Format	Nodes	Reserves	Total resources
Simple	1v1	2	$2 \times 2{,}000$	4,000
Simple Mounds	1v1	2	$2 \times 2{,}000$	4,000
Bottleneck	1v1	6	$2 \times 2{,}000 + 4 \times 600$	6,400
King of the Hill	1v1	6	$2 \times 800 + 4 \times 600$	4,000
Simple Map Control	1v1	6	$2 \times 800 + 4 \times 600$	4,000
Simple	2v2	4	$4 \times 2{,}000$	8,000
King of the Hill	2v2	8	$4 \times 800 + 4 \times 600$	5,600
Simple Map Control	2v2	8	$4 \times 800 + 4 \times 600$	5,600

B SWISS LEADERBOARD EXPERIMENTS

Table 8 lists the model configurations considered for evaluation. For each model, we first use an elimination-bracket tournament to select a thinking variant. We then compare scratchpad on versus off at that selected thinking level. The resulting configuration is used in the Swiss Leaderboard comparison against other models.

Token budget and screening. We request an initial generation budget of 10,240 tokens per call. If a response reaches the cap without a complete action array, the scaffold makes an action-recovery request with up to 4,096 additional tokens, asking for final actions to reduce truncation-induced no-ops. During configuration screening, we omit from Table 8 thinking levels that exceed the initial budget in more than 5% of total calls, since they require a larger reasoning budget to be used meaningfully. This provides a common requested output allowance with limited recovery flexibility for a fairer comparison: a *high* reasoning setting in one model does not necessarily correspond to *high* in another.

Table 8: The 18 models evaluated in Thinking-Variant Selection, Scratchpad Selection, and Swiss Leaderboard, grouped by provisional size class. Thinking variants match Thinking-Variant Selection. Scratchpad Selection tests both on and off for every model. Model-name colors identify families.

Model	Weight access	Thinking variants	Scratchpad
Large *Approximately 1T+ parameters*			
GPT-6 Astra	Proprietary	Low, medium, high, xhigh	On / off
GPT-5.6 Sol	Proprietary	None, low, high, xhigh	On / off
Claude Opus 5	Proprietary	Disabled; adaptive low, high	On / off
Claude Fable 5.1	Proprietary	Adaptive low, medium, high	On / off
Grok 4.6	Proprietary	Low, medium	On / off
Qwen3.8-2.4T-A95B	Open-weight	Low, medium	On / off
Kimi K3	Open-weight	Low, high	On / off
Gemini 3.1 Pro	Proprietary	Low, high	On / off
DeepSeek V4 Pro	Open-weight	Off; low, high	On / off
Medium *100B–1T parameters*			
Gemini 3.7 Flash	Proprietary	Low, high	On / off
Claude Sonnet 5	Proprietary	Disabled; adaptive high	On / off
DeepSeek V4 Flash	Open-weight	Off; low, high	On / off
GLM 5.3	Open-weight	Low, high	On / off
GPT-5.6 Terra	Proprietary	None, low, high, xhigh	On / off
Small *Fewer than 100B parameters*			
GPT-5.6 Luna	Proprietary	None, low, high, xhigh	On / off
Claude Haiku 4.5	Proprietary	Off; on with a 1,024-token thinking budget	On / off
Qwen3.8-27B	Open-weight	Disabled, low, medium	On / off
GLM 5.3 Flash	Open-weight	Low, high	On / off

Thinking-Variant Selection and Scratchpad Selection use the 18 models in Table 8. Every played matchup contains ten games: one game on each of the five 1v1 maps from each engine seat, with a 250-tick cap. Observation history is one tick and order-history recap is disabled. Model-specific thinking controls are those listed in Table 8.

B.1 THINKING-VARIANT SELECTION

Thinking-Variant Selection is an ordered single-elimination comparison within each model. Its 49 variant entrants produce 31 ten-game matchups and 310 games in total. Scratchpad is disabled throughout. Lower-thinking variants occur earlier in each bracket; equal win counts advance the earlier entrant, and three-entrant brackets give the bottom entrant a first-round bye. This selects

a configuration for subsequent experiments but does not compare every pair of thinking levels or guarantee the globally best configuration.

The highest tested thinking level wins 14 of the 18 brackets. The exceptions are GPT-5.6 Sol (high, after a 5–5 tie), Gemini 3.1 Pro (low, after a 5–5 tie), DeepSeek V4 Pro (low, after a tied 5–5 final against high), and Haiku 4.5 (thinking off, 6–4 against its 1,024-token thinking variant). These results favor higher thinking in most of the tested brackets. Figures 5–6 show the match-level elimination brackets, including advancement, byes, and tied-win decisions. The underlying reports retain aggregate records and bootstrap intervals.

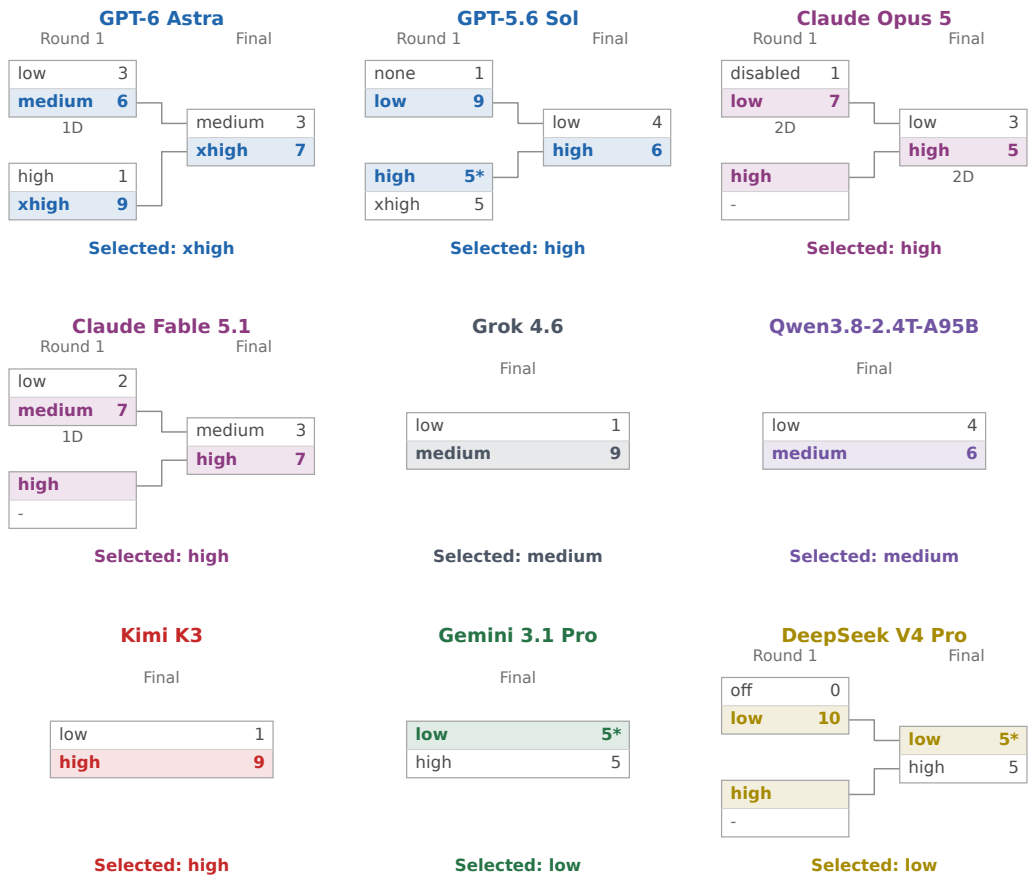


Figure 5: Thinking-Variant Selection: elimination brackets for the first nine models in Table 8. Each match contains ten games; row numbers are wins and D gives the number of draws when nonzero. Bold shaded rows advance; an asterisk marks advancement by the seeded tie-break after equal win counts. A − denotes a bye (no played games). Each panel names the selected variant.

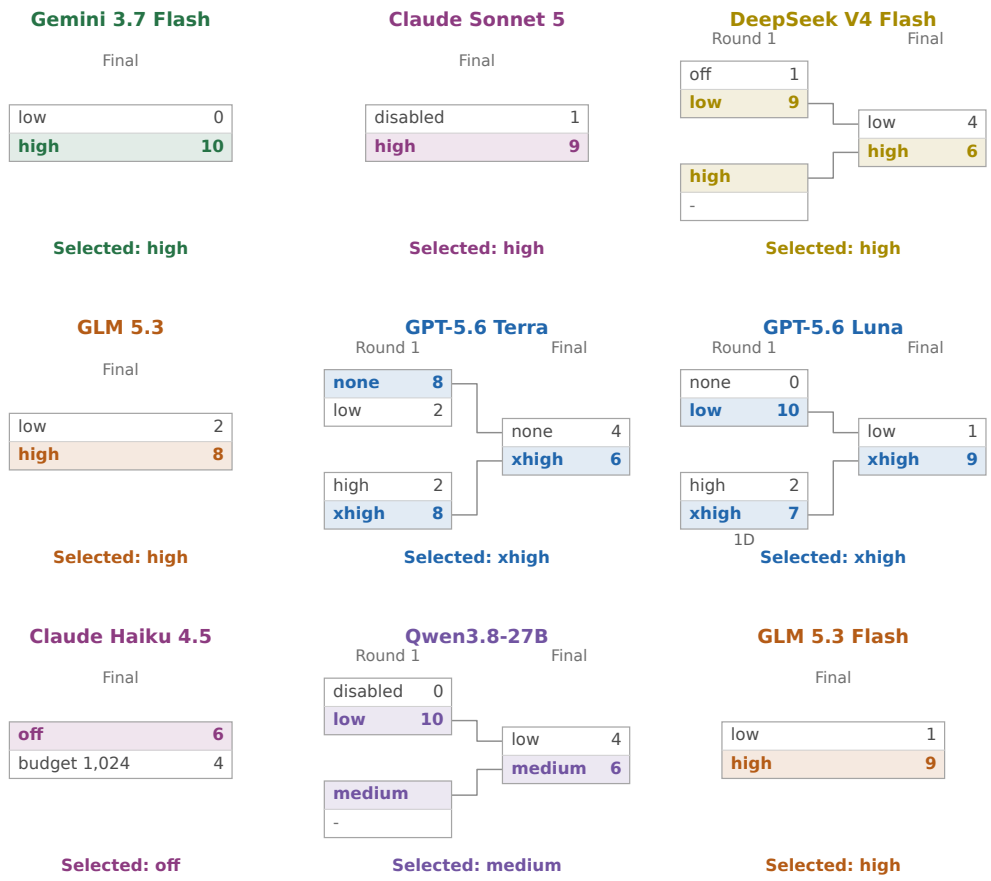


Figure 6: Thinking-Variant Selection: brackets for the remaining nine models, using the conventions of Figure 5; – denotes a bye (no played games). Haiku's enabled condition has a 1,024-token thinking budget; the other labels denote configured effort levels or disabled conditions.

B.2 Scratchpad Selection

Scratchpad Selection compares scratchpad on versus off at the thinking level selected for each model in Thinking-Variant Selection, keeping the remaining player configuration fixed. The 18 ten-game matchups contain 180 games. The no-scratchpad entrant has the top-line tie-break priority, although none of these completed matchups requires a tied-win decision.

Scratchpad wins only for Qwen3.8-2.4T-A95B (7–3–0), Kimi K3 (7–2–1), and GLM 5.3 Flash (7–3–0); the other 15 models select no scratchpad. Across all 180 games, scratchpad-on has 61 wins, 114 losses, and five draws (33.9% wins; 35.3% score when a draw counts one half). Figure 7 shows the model-level outcomes and thinking settings. These are ten-game selection comparisons, not claims that every individual difference is statistically resolved.

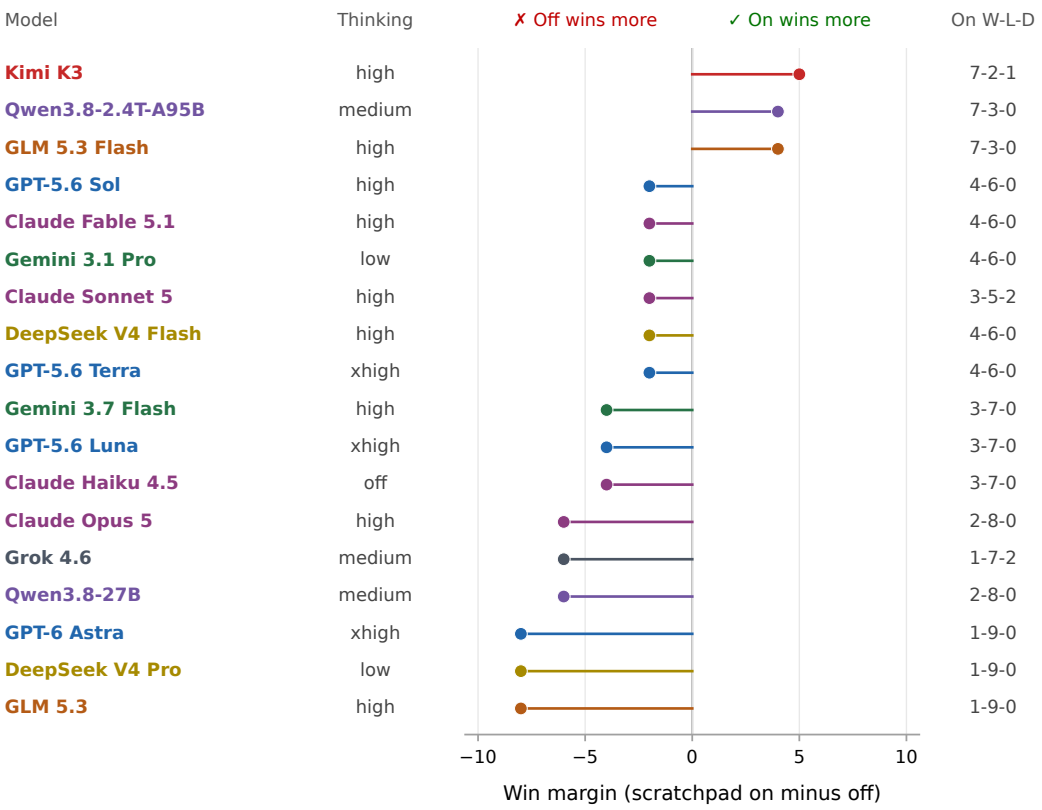


Figure 7: Scratchpad Selection: observed win margin for scratchpad-on versus scratchpad-off, ten games per model. Right of zero means more wins with scratchpad; left means more wins without it. The side with more wins is selected. The left-hand Thinking column gives the configured thinking level used for both conditions; the rightmost column gives scratchpad-on wins–losses–draws. Models are sorted by margin; colors identify model families. Draws contribute no win margin.

B.3 Swiss Leaderboard

Table 9 gives the final thinking and scratchpad configurations selected by the two stages and used in the ten-round Swiss Leaderboard tournament.

Table 9: Selected configurations for the Swiss Leaderboard. Scratchpad: ✓ enabled; ✗ disabled.

Model	Selected thinking	Scratchpad
GPT-6 Astra	xhigh	✗
GPT-5.6 Sol	high	✗
Claude Opus 5	high (adaptive)	✗
Claude Fable 5.1	high (adaptive)	✗
Grok 4.6	medium	✗
Qwen3.8-2.4T-A95B	medium	✓
Kimi K3	high	✓
Gemini 3.1 Pro	low	✗
DeepSeek V4 Pro	low	✗
Gemini 3.7 Flash	high	✗
Claude Sonnet 5	high (adaptive)	✗
DeepSeek V4 Flash	high	✗
GLM 5.3	high	✗
GPT-5.6 Terra	xhigh	✗
GPT-5.6 Luna	xhigh	✗
Claude Haiku 4.5	off	✗
Qwen3.8-27B	medium	✗
GLM 5.3 Flash	high	✓

Opponent (BT rank)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1 Astra		7	10	9	10	10	10	10	10	-	10	-	10	-	-	-	-	-
2 Fable 5.1	3		6	8	9	9	8	10	-	9	10	10	-	-	-	-	-	-
3 Opus 5	0	4		6	9	9	10	-	-	9	9	10	-	-	10	-	-	-
4 Gemini 3.7 Flash	1	2	4		7.5	8	9	7	-	8	-	-	-	10	10	-	-	-
5 Sol	0	1	1	2.5		6	-	7	9	5	-	-	8	-	-	-	-	10
6 Grok 4.6	0	1	1	2	4		-	-	7	9	-	-	8	8	-	8	-	-
7 GLM 5.3	0	2	0	1	-	-		4.5	7	-	-	10	-	6	-	-	10	10
8 Kimi K3	0	0	-	3	3	-	5.5		6	-	7	5	7	-	-	-	-	10
9 Sonnet 5	0	-	-	-	1	3	3	4		-	7	5.5	-	-	-	9	9	10
10 Gemini 3.1 Pro	-	1	1	2	5	1	-	-	-		-	-	4	7	7	9	8	-
11 Qwen 2.4T	0	0	1	-	-	-	-	3	3	-		-	-	7	8	7	9	10
12 GLM 5.3 Flash	-	0	0	-	-	-	0	5	4.5	-	-		5.5	-	6	7	9	10
13 Terra	0	-	-	-	2	2	-	3	-	6	-	4.5		6	6	7.5	6	-
14 Luna	-	-	-	0	-	2	4	-	-	3	3	-	4		7	1	7	10
15 DeepSeek Pro	-	-	0	0	-	-	-	-	-	3	2	4	4	3		6	6	10
16 Qwen 27B	-	-	-	-	-	2	-	-	1	1	3	3	2.5	9	4		6	7
17 DeepSeek Flash	-	-	-	-	-	-	0	-	1	2	1	1	4	3	4	4		9
18 Haiku 4.5	-	-	-	-	0	-	0	0	0	-	0	0	-	0	0	3	1	

Figure 8: Observed pairwise game scores after ten Swiss rounds (900 games). Each cell is the row model's score against the column model out of ten games: one point per win and half per draw. A – means unplayed; gray diagonal cells denote self-comparisons. Rows and columns follow the BT rank order in Table 10.

Rating estimation. Bradley–Terry (BT) ratings use draws as half a win and a symmetric prior of 0.01 games per player pair. Abilities are geometrically normalized and reported as $400 \log_{10}(a_i)$, giving zero-centered ratings.

Draws and starting positions. Only five of 900 games end in a draw (0.56%). The two starting positions yield 444 and 451 wins overall, but individual models show larger differences. Qwen3.8-27B wins 28% from one position and 48% from the other; Kimi K3 wins 54% and 38%, respectively. Each estimate uses 50 games. Aggregate balance therefore does not establish equal performance across starting positions for every model; per-position results are reported in Table 10.

Table 10: Final Swiss Leaderboard, ordered by zero-centered BT rating; colors identify model families. Each model played 100 games. Draw and seat columns are percentages; P1/P2 win rates use 50 games each and count draws as non-wins. Selected configurations are in Table 9.

Rank	Model	BT	W–L–D	Draw %	P1 %	P2 %
1	GPT-6 Astra	796.7	96–4–0	0	94	98
2	Claude Fable 5.1	517.6	82–18–0	0	78	86
3	Claude Opus 5	424.8	76–24–0	0	72	80
4	Gemini 3.7 Flash	310.6	66–33–1	1	64	68
5	GPT-5.6 Sol	115.7	49–50–1	1	48	50
6	Grok 4.6	84.7	48–52–0	0	44	52
7	GLM 5.3	53.8	50–49–1	1	50	50
8	Kimi K3	14.1	46–53–1	1	54	38
9	Claude Sonnet 5	-51.2	51–48–1	1	50	52
10	Gemini 3.1 Pro	-58.6	45–55–0	0	48	42
11	Qwen3.8-2.4T-A95B	-95.0	48–52–0	0	46	50
12	GLM 5.3 Flash	-126.0	46–52–2	2	50	42
13	GPT-5.6 Terra	-141.2	42–56–2	2	46	38
14	GPT-5.6 Luna	-223.8	41–59–0	0	46	36
15	DeepSeek V4 Pro	-257.2	38–62–0	0	40	36
16	Qwen3.8-27B	-277.7	38–61–1	1	28	48
17	DeepSeek V4 Flash	-362.6	29–71–0	0	26	32
18	Claude Haiku 4.5	-724.8	4–96–0	0	4	4

Cost per tick in Figure 2. For each model, we divide its summed recorded charges plus the existing null/zero-cost estimates by its player ticks (one model decision per tick) in the same 900 retained Swiss Leaderboard games. The 201,622 player ticks include final failures; retries and cap recoveries contribute their logged charges within a tick, not additional denominator ticks. Failures without usable cost or token data remain unpriced. The retained-game total is $4,435.27.

Table 11: Swiss Leaderboard behavioral metrics averaged over each model's 100 retained games, ordered by BT rank as in Table 10; colors identify model families. Economy is the replay analytics proxy: summed positive changes in resource balance per tick, not gross harvested income. Valid orders count actors, not JSON objects. Incurred USD/game allocates all Swiss Leaderboard costs for that model, including discarded attempts and null/zero-cost estimates, across its 100 retained games; it is not a retained-only replay charge.

Rank	Model	Economy/tick	Valid orders/tick	Incurred USD/game
1	GPT-6 Astra	10.93	2.64	6.89
2	Claude Fable 5.1	10.47	2.67	6.93
3	Claude Opus 5	10.02	2.99	3.76
4	Gemini 3.7 Flash	9.99	2.30	1.23
5	GPT-5.6 Sol	9.39	2.08	3.63
6	Grok 4.6	8.94	4.24	3.60
7	GLM 5.3	8.86	2.34	1.08
8	Kimi K3	9.25	1.89	4.89
9	Claude Sonnet 5	8.85	2.11	3.12
10	Gemini 3.1 Pro	8.78	2.05	1.04
11	Qwen3.8-2.4T-A95B	8.48	1.74	2.13
12	GLM 5.3 Flash	8.68	1.98	0.23
13	GPT-5.6 Terra	8.85	1.79	2.44
14	GPT-5.6 Luna	8.06	1.49	0.25
15	DeepSeek V4 Pro	8.98	2.47	1.65
16	Qwen3.8-27B	8.95	1.75	0.81
17	DeepSeek V4 Flash	8.27	3.06	0.28
18	Claude Haiku 4.5	6.79	2.16	0.46

Table 12: API costs for the four main experiments only (USD). Costs include unique discarded attempts and estimates for null/zero cost entries; failures without usable cost or token data are excluded. Game counts refer to retained completed games. Older and skill-learning experiments are not included.

Experiment	Retained games	Cost (USD)
Thinking-Variant Selection	310	1,746.14
Scratchpad Selection	180	1,124.73
Swiss Leaderboard	900	4,442.31
Team-Chat Evaluation	72	1,378.16
Total (four main experiments)	**1,462**	**8,691.33**

Appendix E analyzes full-game Swiss trajectories using behavioral features, a two-dimensional t-SNE embedding, and descriptive strategic clusters, with model-level cluster distributions.

C SKILL LEARNING: PROTOCOL AND RESULTS

A teacher model (GPT-5.6 Sol, high reasoning) reads a student's game replays and writes a *handbook*: a short set of natural-language skills that is added to the student's prompt for later games. Each generation adds new training games and produces a new frozen handbook. Generation labels (Gn) identify these handbooks, not independent training runs. Unless stated otherwise, training and evaluation use the `simple_2p` map with player-relative coordinates. Records are wins–losses–draws (W–L–D), and win rates count draws as non-wins.

C.1 STATEFUL VERSUS STATELESS TEACHERS

Haiku 4.5 trains against four bots (Random, Economy, Rush, and Defensive) for twelve generations. Each generation adds four games, one per bot, with alternating seats. A handbook holds at most eight skills and 8,000 characters. Both teachers can inspect every accumulated training game. The *stateful* teacher also sees the current handbook and the generation labels. The *stateless* teacher instead receives anonymized games with handbook messages and chronology removed, so it rebuilds the handbook from the evidence alone (Appendix D). We evaluate each checkpoint over twenty games against Sonnet 4.6, which is never used in training. G0 is the shared no-skill baseline.

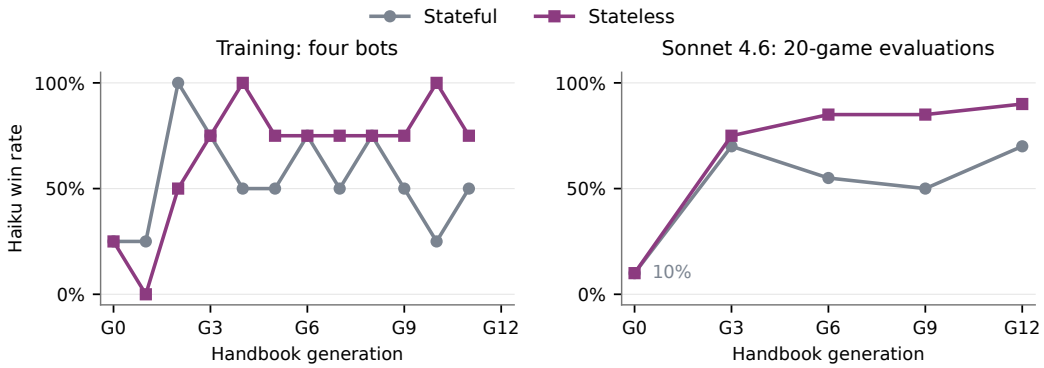


Figure 9: **Stateless reconstruction yields a steadier trajectory.** Training uses four games per handbook; each Sonnet evaluation uses twenty games. G0 is the shared unskilled baseline (2/20 wins, 10%). Stateful learning improves early and then regresses. Stateless win rates rise from 75% at G3 to 90% at G12. Each curve is a single training trajectory.

Table 13: Teacher comparison against Sonnet 4.6, twenty games per checkpoint. G0 is the shared no-skill baseline.

Checkpoint	Stateful	Stateless
G0	2–18–0 (shared)	
G3	14–1–5	15–3–2
G6	11–3–6	17–3–0
G9	10–9–1	17–2–1
G12	14–5–1	18–0–2

We introduced the stateless design after observing that the stateful teacher anchors on recent evidence. After its first session, 265 of its 299 file reads (88.6%) target the latest generation, and nine of its twelve rationales open by naming the preceding generation. The comparison does not isolate chronology alone, because the stateless teacher also loses access to earlier handbooks and receives a larger tool-call budget. Over the 48 training games, the stateful and stateless runs record 26–10–12 and 32–11–5. The Economy bot remains difficult for both teachers, with two wins each.

C.2 BOT-TRAINED TRANSFER

Haiku and Luna each train for twelve generations against the four scripted bots, with one game per bot per generation. We freeze each evaluated handbook and play twenty games per opponent and checkpoint on `simple_2p`, using ten seeds in both seats. Evaluation games never enter teacher evidence. Haiku uses thinking off and Luna uses xhigh reasoning, following the tournament configuration; Sonnet, Sol, and Opus use high reasoning, with an additional Sonnet-without-thinking condition for Haiku. Figure 4 reports each available checkpoint and sample standard deviations across seed-pair win rates. Draws remain in the denominator.

Behavioral assignments apply the frozen Swiss preprocessing and nearest centroids in the 270-dimensional feature space. The t-SNE reference remains fixed, and checkpoint centers are geometric medians of projected player-games. Paper labels use the positional numbering of Appendix E.1; A1 denotes order errors and delayed army, A4 striker-heavy army decline, and A10 the marauder-led opening. These profiles summarize realized games rather than independently chosen policies.

C.3 HAIKU'S THINKING CONFIGURATION

We retain Haiku's tournament-selected configuration: thinking off and no scratchpad (Table 9). In Thinking-Variant Selection, this configuration won six of ten games against Haiku with a 1,024-token thinking budget (Appendix B.1). Keeping it fixed in the main skill-learning evaluation makes the comparison consistent with the tournament.

Table 14 reports a reference evaluation with Haiku thinking off and on against Sonnet 5 both without thinking and with adaptive high thinking. We reuse the same frozen G3/G6/G9/G12 handbooks, without retraining. Every cell uses twenty games on `simple_2p`: seeds 21000–21009 in both seats, a 200-tick limit, and no scratchpad. Haiku's enabled-thinking budget is 1,024 tokens; Sonnet receives no handbook.

Table 14: **Haiku's final-checkpoint results are similar with and without thinking.** Haiku wins–losses–draws and win rates over twenty games per cell. Column groups fix Sonnet's thinking setting; columns compare Haiku's setting. Draws count as non-wins. All four conditions use the same handbook at each checkpoint.

	Sonnet: thinking off		Sonnet: high thinking	
Checkpoint	Haiku off	Haiku on	Haiku off	Haiku on
G3	14–6–0 (70%)	16–4–0 (80%)	5–15–0 (25%)	8–12–0 (40%)
G6	18–2–0 (90%)	20–0–0 (100%)	4–16–0 (20%)	7–12–1 (35%)
G9	20–0–0 (100%)	20–0–0 (100%)	7–12–1 (35%)	13–7–0 (65%)
G12	19–1–0 (95%)	20–0–0 (100%)	11–9–0 (55%)	10–10–0 (50%)

At G12, enabling Haiku's thinking changes the result by only one win in either direction: 19/20 to 20/20 against Sonnet without thinking, and 11/20 to 10/20 against Sonnet with high thinking. The final-checkpoint conclusion is therefore similar under either Haiku setting: near-perfect performance against Sonnet without thinking and roughly half the games won against Sonnet with high thinking. Earlier differences can be larger, reaching 30 percentage points at G9 against high-thinking Sonnet. These single-handbook evaluations do not establish equivalence across settings, but they show that the final result is not dependent on disabling Haiku's thinking.

C.4 TRAINING-OPPONENT REGIMES

We compare five bot-training conditions for Haiku 4.5 and GPT-5.6 Luna: Random, Defensive, Economy, and Rush individually, and a mixed-bot condition with one game against each bot per generation. There is one lineage per learner and condition, giving ten lineages in total. Each handbook is derived only from its own learner's training games.

Each generation gives every learner four games on `simple_2p`, two in each seat. The stateless teacher writes at most eight skills and 8,000 characters. It works only from the assigned learner's accumulated games and never sees previous handbooks.

We evaluate the frozen G3 and G6 handbooks against eight fixed, unskilled opponents: Opus 5, GPT-5.6 Sol, Gemini 3.7 Flash, Sonnet 5, Gemini 3.1 Pro, GPT-5.6 Terra, original Luna, and original Haiku. Each opponent is played on five maps in both seats, which gives ten games per opponent and 80 per handbook. Both checkpoints use identical schedules that are disjoint from training. We chose to continue to G6 after inspecting partial G3 results, so G6 is an exploratory extension. Every lineage is a single training run.

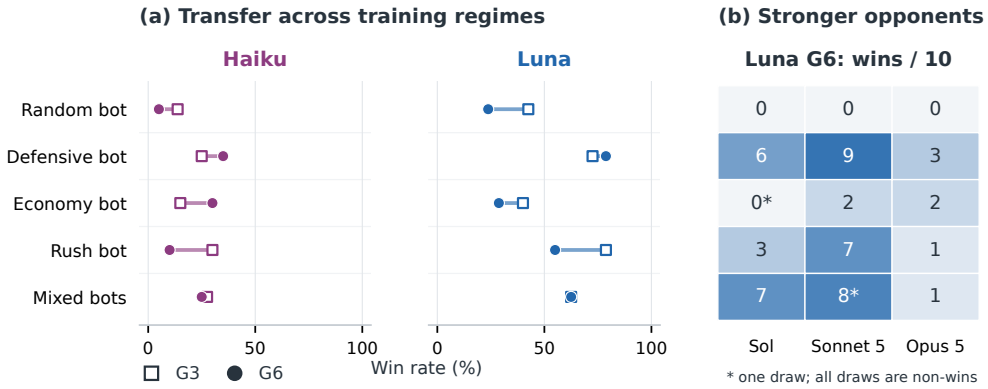


Figure 10: **The training opponent decides which counter is distilled, and only some counters transfer.** Left: G3 and G6 win rates against the full eight-opponent panel (80 games each). Lines join evaluations of successive handbooks for the five bot-training conditions. Right: Luna's G6 wins out of ten against three stronger opponents. Tables 15 and 16 report every matchup. These are individual training runs, not averages over repeated training.

The training bot changes the learned handbook and its evaluation performance. At G6, Defensive-bot Luna wins 63/80 panel games, compared with 50/80 for mixed bots, 44/80 for Rush, 23/80 for Economy, and 19/80 for Random. Haiku's G6 records range from 4/80 with Random to 28/80 with Defensive. Each matchup contains five map/seed blocks, so these differences describe the learned handbooks.

Table 15: **Haiku across training regimes.** Entries are wins out of ten, and Panel is wins out of 80. Superscripts give draw counts (e.g., 1^{1d} is one win and one draw); all other games are losses. Flash and Pro denote Gemini 3.7 Flash and Gemini 3.1 Pro.

Training opponent	G	Panel	Opus	Sol	Flash	Sonnet	Pro	Terra	Luna	Haiku
Random bot	G3	11^{1d}	0	0	0	2	1	0	1^{1d}	7
	G6	4	0	0	0	0	0	0	0	4
Defensive bot	G3	20	0	1	0	0	2	2	6	9
	G6	28^{1d}	0	0	1^{1d}	7	2	4	5	9
Economy bot	G3	12	0	1	0	0	0	1	2	8
	G6	24	1	0	1	2	2	3	6	9
Rush bot	G3	24	0	2	0	3	5	3	2	9
	G6	8^{1d}	0	0	0	1	0	1	0^{1d}	6
Mixed bots	G3	22	0	0	0	1	3	4	5	9
	G6	20	0	2	0	2	2	3	3	8

What each regime teaches. Each handbook encodes a counter to what its training opponent fielded. Table 17 relates Luna's G6 handbooks to the unit composition it actually produced and to its evaluation record.

- **Defensive bot.** This bot fields only soldiers. The handbook prescribes five workers, one barracks, one soldier, and then continuous marauders. The result is 63/80.
- **Economy bot.** This bot fields only marauders. The handbook defaults to pure strikers, two barracks, and a 12–14-striker attack. Luna loses its first four training games against this bot

Table 16: **Luna across training regimes.** Same schedule and notation as Table 15.

Training opponent	G	Panel	Opus	Sol	Flash	Sonnet	Pro	Terra	Luna	Haiku
Random bot	G3	34	0	5	0	3	5	2	9	10
	G6	19^{1d}	0	0	0	0	4^{1d}	4	2	9
Defensive bot	G3	58	3	6	3	8	8	10	10	10
	G6	63	3	6	6	9	9	10	10	10
Economy bot	G3	32^{1d}	0	3	1	0^{1d}	4	7	7	10
	G6	23^{1d}	2	0^{1d}	1	2	1	2	6	9
Rush bot	G3	63	2	8	4	10	9	10	10	10
	G6	44	1	3	1	7	7	6	9	10
Mixed bots	G3	50	2	6	3	7	7	8	7	10
	G6	50^{3d}	1	7	5	8^{1d}	8	5^{2d}	7	9

and wins all twenty after that, so its weak transfer (23/80) is not a failure to learn the training opponent. The counter simply does not generalize: in evaluation, 83% of its basic-unit hits land in unfavorable matchups.

- **Rush bot.** From G2 onward, Luna wins every training game, so the G6 teacher sees only wins. Between G3 and G6 the handbook's worker cap drifts from 8–9 to 20–24. Mean living workers rise from 8.5 to 14.7, and the marauder share falls from 65.7% to 37.0%.
- **Mixed bots.** The handbook has no default composition. It prescribes 7–8 workers before the first barracks, a second barracks around ticks 40–50, and production chosen after scouting. The result is 50/80 at both checkpoints.

These bot-trained runs associate different realized compositions with different evaluation records. Defensive-bot Luna produces mostly marauders and wins 63/80 at G6, whereas Economy-bot Luna produces almost entirely strikers and wins 23/80.

Table 17: **Luna's G6 realized composition and transfer by training opponent.** Production shares are mean per-game percentages of newly produced non-worker units, including scouts. A10 is the share of the 80 evaluation games nearest profile A10. The Random-bot and mixed-bot lineages are omitted.

Training opponent	Soldier %	Marauder %	Striker %	A10 %	Wins / 80
Rush bot	33.6	37.0	15.0	26.2	44
Defensive bot	10.3	81.0	0.3	68.8	63
Economy bot	0.0	0.0	99.0	1.2	23

Behavioral profiles. We assign each learner's G3/G6 evaluation games to the nearest of the ten strategic profiles from Section 4.2. The 270-dimensional features and centroids are frozen, and nothing is refitted. Figures 11 and 12 show the full profile shares. Different regimes drive the same model to different profiles:

- **Defensive-bot Luna** lands in A10 in 71% and 69% of games, winning 72.5% and 78.8%.
- **Mixed-bot Luna** most often lands in A9 (26% and 31%) and wins 62.5% at both checkpoints.
- **Mixed-bot Haiku** moves away from its historical A1 profile (92%), falling to 0% at G3 and 45% at G6.

Profile IDs do not rank strength. Some learned behaviors also lie far from every reference profile: 78–80% of economy-bot Luna's games exceed the Swiss 95th-percentile nearest-centroid distance. Labels therefore describe resemblance to known profiles, not adoption of a known strategy.

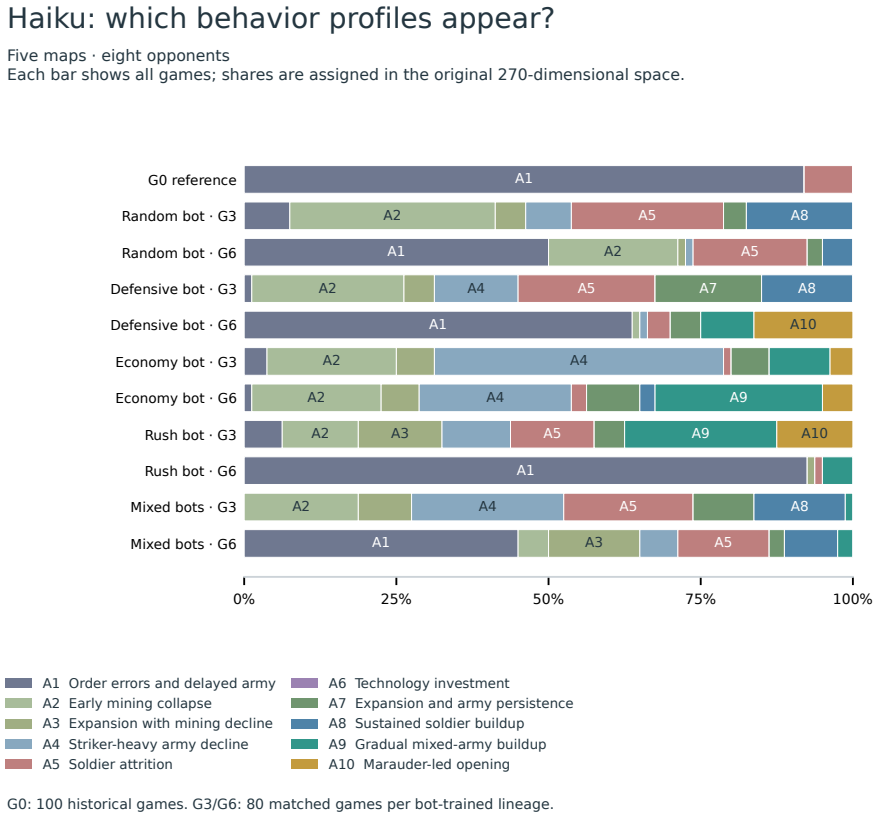


Figure 11: **Haiku's profile shares across training regimes.** Nearest-profile assignments for 100 evaluation games per bot-trained lineage and checkpoint.

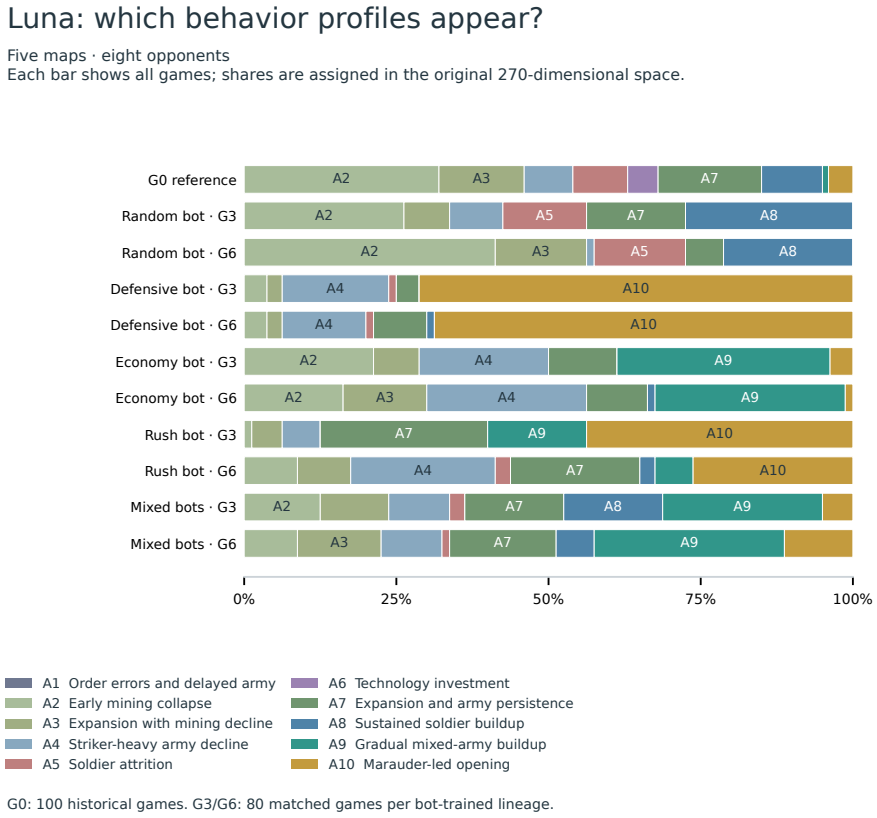


Figure 12: **Luna reaches different profiles under different training regimes.** Same protocol as Figure 11.

C.5 EXAMPLE HANDBOOKS

Below are the complete G12 handbooks of Haiku and Luna from the mixed-bot lineages. They contain the teacher's prescriptions. Haiku's book fixes a marauder recipe and spends a skill on order validity; Luna's conditions production on the visible enemy army.

Skill Handbook Prompt: Haiku G12

These are lessons distilled from previous TextCraft games. They are advice, not rules. Apply them when relevant, but obey the game rules and current state above all.

skill_opening_economy: Immediately build an extractor on the visible starting resource beside your command center and queue four workers from the command center. Wait for completion, then harvest with every unloaded idle worker, including the builder and new workers. Harvest handles all travel/pickup/return; never move workers onto the extractor or command-center tile. Reach 7–8 total workers, then stop. Keep 6–7 harvesting while one builds. If that gives much less than about 24–32 income, find unloaded workers on idle/hold/stale move and reissue harvest. Do not redirect loaded workers returning to deposit.

skill_timely_barracks_and_second_production: After the opening queue, save 150 and start a barracks around ticks 17–21; starts around 29–51 repeatedly failed. Use a currently visible empty passable tile near home, not a resource, structure, ramp, or shuttle lane. Make exactly one scout, then continuously queue marauders. Prefer army to workers beyond the 7–8 cap. If safe and 150 remains after an initial unit queue, add a second barracks in early-midgame; this won the hard marauder mirror. Resume a stalled barracks at the identical site/type. Delay tech, upgrades, and expansion until a strong tier-1 army exists.

skill_composition_rules: Default to a marauder core. Correct matchup facts: marauders get +40% versus soldiers, while soldiers deal −40% to marauders. Soldiers are only for enemy striker masses (+30% versus strikers); never make soldiers into marauders. Strikers are modifier-favored against marauders but must close range, so at most use a small screen, not a replacement for ranged marauders. In a marauder mirror, rely on two-barracks production, cohesion, and numbers rather than panic-pivoting. One scout supplies vision and should not duel full-health combat units.

skill_scout_and_mass: Send the scout through passable terrain toward the unexplored frontier away from home. After finding the enemy, pull it behind the marauders to preserve vision. Gather reinforcements at one safe forward staging area instead of trickling them. Use 4–6 marauders against an undefended, worker-heavy base; against a marauder mass, consolidate about 8–10 or a clear numerical edge. If intel shows many workers or an extra command center but little army, attack with the first coherent force rather than matching the greed.

skill_focused_attacks: Units fire only with attack orders. When an enemy is visible, use exact LIVE INTEL IDs, not move/hold. Assign about 3–4 marauders per enemy instead of one-on-one splitting; use everyone on the sole target. Kill combat units before workers and remove soldiers quickly. Attack pursuit already closes and may fire that tick, so do not alternate attack with move. Valid attacks persist through cooldown. On ENEMY_KILLED, immediately give those units a new visible target because they will not auto-select. Never invent an ID or give one unit multiple orders in a tick.

skill_high_ground_defense: If a large force approaches or a base structure is attacked, stop the outbound march and reunite the army between the threat and command center. Prefer the visible high-ground side of a ramp: firing down gives +25%, charging uphill is penalized. Keep the scout behind the line, then focus the closest visible attackers. Do not chase one retreating unit onto low ground while the main army advances. Keep workers harvesting and producing reinforcements; do not use them as combat scouts.

skill_finish_decisively: After the enemy army breaks, advance the whole core with scout vision. Destroy a producing barracks first if it can replenish defenders; otherwise attack an exposed command center directly. Then keep every combat unit on the exact visible command-center ID until it dies. Ignore extractors and scattered workers. Retarget after each destruction alert. If the base is in fog, move toward its last observed area, reveal it, then use exact IDs. Convert the advantage before resources deplete instead of waiting for a larger army or timing out.

skill_order_validity_check: Before acting, read ALERTS, resources, queues, and exact IDs. Command center produces workers; barracks produces tier-1 combat units. Ensure all new orders together fit the bank, and do not blindly add to an existing queue. Build only on a currently visible empty legal tile; harvest only from your completed owned extractor with real worker IDs. If ORDER_FAILED appears, fix the cause instead of repeating it. Orders persist, so do not reissue them every tick; when nothing needs changing, return [].

Skill Handbook Prompt: Luna G12

These are lessons distilled from previous TextCraft games. They are advice, not rules. Apply them when relevant, but obey the game rules and current state above all.

Skill Handbook Prompt: Luna G12 (continued)

skill_opening_script: Default opening: on tick 0, have one worker build an extractor on the visible resource nearest the command center and have the command center produce a worker in the same response. Keep the other worker home; never scout across the map with an opening worker. When the extractor completes, order all workers to harvest it. Keep producing workers and assign each new one to harvest. At about 8 workers, build one barracks on a visible empty passable tile near home without blocking the ramp or extractor. Use one builder; continue to 12–14 workers.

skill_worker_cap_and_two_barracks: Treat 12–14 workers as a hard standard cap; then stop workers and spend on fighters. From the first barracks make one scout and 1–2 fighters, then build a second barracks when 150 is available and the first is producing—usually in the late 30s to 50, not after losing the army. If resources bank above 150 with only one barracks, build the second. Keep both barracks busy. Skip tech, upgrades, and a second command center during the standard fight; successful games were decided by barracks armies before these paid back.

skill_scout_without_sacrificing_economy: Use one scout, not a worker, for early intel. Move in short legs along passable terrain toward unexplored routes and enemy approaches; inspect the minimap so resources, walls, and edges do not cause no-path orders. Reveal the enemy army, base, and exact LIVE INTEL IDs, but do not fight. Withdraw or circle for vision when threatened. Do not mass scouts. Rally later fighters safely near the barracks and move them forward in groups.

skill_counter_the_visible_army: Counter the most numerous dangerous type in LIVE INTEL. Against marauders, mass strikers and avoid soldiers. Against soldiers, mass marauders and avoid strikers. Against strikers, mass soldiers. For a mixed army, split production by visible counts while keeping a majority counter to its majority. While composition is unknown, one scout plus 1–2 soldiers is enough; switch later cycles when intel arrives. Do not cancel a fighter already in production just to switch; finish it, then change the next cycle and avoid the 25% cancellation loss.

skill_assemble_waves_and_use_terrain: Do not send each new fighter alone. Gather reinforcements at a safe staging tile on the route, then move a wave of roughly 5–8 fighters together; attack earlier only to defend or when the visible enemy is clearly weaker. If a small group meets a larger force, pull it toward reinforcements. When available, stage ranged units on ^ high ground or defend the top of a ramp. Keep marauders behind tougher units and let strikers close together.

skill_explicit_focus_fire: Units never auto-attack. When a target is in LIVE INTEL, issue `attack` with its exact ID to every suitable fighter. Attack pursuit can move and fire that tick; normal movement prevents firing. Focus one enemy fighter at a time, preferring low HP or a counter to your army, then retarget immediately when it dies. Do not split attacks onto workers or structures while enemy weapons still fire. Give each unit at most one order per tick; do not move a ready ranged unit that can attack. An accepted attack can continue through fog, but a dead target leaves units needing a new order.

skill_defend_then_finish_cleanly: A base-under-attack alert overrides a distant raid: recall the army, produce counters from every idle barracks, and attack visible invaders together. Workers cannot fight; repair the command center with only 1–2 workers after the army engages and only while resources remain. Ignore enemy workers until armed units are handled. At the enemy base, kill active fighters first. Once its army breaks, focus the whole army on the visible command center instead of chasing workers or splitting across structures.

skill_every_tick_checklist: Each tick check: (1) alerts and LIVE INTEL; (2) command center if below the worker cap; (3) every idle barracks; (4) new workers needing `harvest`; (5) fighters with dead targets or no useful order. Use only current owned IDs or exact LIVE INTEL target IDs. Build only on currently visible valid cells and issue at most one order per unit. After RESOURCE_DEPLETED, stop harvest retries and worker production; spend the stockpile on fighters and make the grouped finishing push. Idle barracks and large banks are emergencies, not reasons to save for tech.

D ANONYMISATION OF THE TEACHER'S EVIDENCE

Game directories are renamed `game_<sha256(salt:path)[:8]>` with a per-run secret salt; scripted opponents are relabelled `opponent_A..D` in salted order; `result.json` is rebuilt from a whitelist (seat, outcome, final tick, winner, map, tick cap, and behavioural metrics), dropping all token and cost fields; digests are regenerated with neutral labels, no seed, and student-frame coordinates; only the student's call logs are included, with the handbook message and rules message removed and `skill_*` names redacted; replays are excluded. A leak assertion checks the view for real bot names, generation labels, and cost fields before every teacher session.

E BEHAVIORAL PATTERNS IN THE 1V1 SWISS LEADERBOARD GAMES

E.1 DATA AND ANALYSIS PROTOCOL

We analyze all 900 retained games from the ten-round Swiss Leaderboard. Each point represents one player in a game: 1,800 observations, with 100 per model. There are 895 winning, 895 losing, and ten drawing player-games. The analysis covers 100,811 engine ticks and 201,622 turn-log files. We convert each game from the perspective of each player to full-game summary and temporal features. Let T be the final tick. "Mean" averages ticks $1, \ldots, T$ unless an opportunity denominator is stated. A unit-tick counts one unit present in one tick. Military units are soldiers, marauders, strikers, tanks, and enforcers. Composition includes those five types *plus scouts*, but not workers. CC denotes command centre. All stated spatial radii use Chebyshev distance.

Full-game summary features. The full-game block contains 57 numerical features covering economy (7), production (9), composition (9), combat (7), position (4), commands (4), terrain (11), and unit matchups (6) detailed in Table 18. Replay states and executed events supply these measures. Reasoning text is supporting evidence, not an input or proof that an intended action occurred. Model identity, opponent, map name, seat, round, outcome, token usage, cost, and call-failure labels are excluded from fitting. These labels are attached only after fitting. They are not inputs to either feature-space clustering or its visualization.

Table 18: All 57 numerical inputs in the full-game feature block. Parenthesized counts identify rows representing multiple separate features. Group totals sum to 57. Ratios with no eligible denominator are unavailable rather than observed zeros.

Feature	Definition
Economy: 7	
Mining rate	This is the amount of resources depleted from nodes attributable to own completed extractors, divided by T. It does not measure bank growth.
Workers, extractors, and bank (3)	Mean living-worker count, completed-extractor count, and unspent resource balance.
Worker idle share	Worker-ticks without a current order / all worker-ticks.
Resource coverage	Mean of twice the remaining resources at own completed extractors divided by remaining resources on the entire map. The range is 0–2, where 1 means half the remaining resources. A fully exhausted map contributes 0.
Post-depletion mining	Own mining strictly after depletion of the starting node / subsequent ticks. The starting node is nearest the initial CC.
Production: 9	
Barracks and busy share (2)	The first measure is mean completed-barracks count. The second is completed-barracks-ticks with a nonempty production queue / all completed-barracks-ticks.
Military production value	Resource cost of completed military units / T.
Military spending share	Completed military-unit cost / all completed-unit cost, including workers and scouts in the denominator. Buildings and upgrades are excluded.
Scout production	Completed scouts / T.
Affordable-idle share	Idle completed-barracks-ticks with bank ≥ 40 / all completed-barracks-ticks. This resource threshold does not establish that a particular order would succeed.
Technology access	Mean number of completed structure types among tech lab and machine shop (0–2).
Upgrades completed and active (2)	The first measure counts distinct upgrades first observed during the game, excluding initial upgrades. The second is the mean number of active upgrades, including the terminal state.
Composition: 9	
Unit-type shares (6)	Completed count of each of soldier, marauder, striker, tank, enforcer, and scout / all completed units of those six types.
Production entropy	$-\sum_{i=1}^{6} p_i \ln p_i / \ln 6$, using full-game completed-unit shares. Measures diversity, not army size.
Fielded entropy	Mean of the same normalized entropy over ticks with at least one living unit of those six types, using living-unit shares in each tick.
Counter-production requests	Fraction of basic-unit production requests choosing the direct counter when the preceding observation contains a strict majority of one visible basic enemy type: soldier $\rightarrow$ marauder, marauder $\rightarrow$ striker, striker $\rightarrow$ soldier. Failed requests remain in the denominator.
Combat: 7	
HP-weighted army value	Mean of $\sum_u \mathrm{cost}(u)\,\mathrm{HP}(u)/\mathrm{maxHP}(u)$ over living military units.
Kills, losses, and worker kills (3)	Enemy unit value killed, own unit value lost to enemy combat, and enemy workers killed, each divided by T.
Focus-fire proxy	Ticks with two or more attributed military shots co-targeting one position / ticks with at least two attributed military shots. This is positional co-targeting, not verified tactical intent.
Local CC repair response	Eligible ticks with a submitted or continuing repair order / eligible ticks. Eligibility: previously damaged completed own CC, bank ≥ 1, and own worker within radius 1. Measures orders, not restored HP.

Table 18: Full-game feature definitions (continued).

Feature	Definition
Siege targeting	Military unit-ticks targeting enemy structures while enemy military is within radius 6 of that unit / all military unit-ticks with enemy military within radius 6.
Position: 4	
Exploration	Mean cumulative fraction of map tiles observed, including initial visibility in the cumulative union.
Forward presence	Military unit-ticks beyond the midpoint along the line between initial CCs / all military unit-ticks.
Isolation	Mean fraction of military units without another friendly military unit within radius 4, over military-present ticks.
Holding / no order	Military unit-ticks holding or without a current order / all military unit-ticks.
Commands: 4	
Gameplay and attack orders (2)	Submitted gameplay-order objects and submitted attack-order objects, each divided by T. Group commands count once, and invalid orders are included. No-ops, scratchpad updates, and messages are excluded.
No new order	This is the number of ticks with no submitted gameplay-order objects divided by T. Persistent orders can still act.
Order-failure share	Ticks with at least one engine `order_failed` alert / T. This includes command conflicts/overrides and is distinct from provider/API failures.
Terrain: 11	
Downhill and uphill attacks (2)	Outgoing high-to-low and low-to-high unit-target impacts, respectively / all outgoing unit-target impacts.
Downhill exposure	Incoming high-to-low unit-target impacts / all incoming unit-target impacts.
Defensive downhill fire	Downhill impacts against enemy military threatening own completed CCs/extractors / all impacts against those threats. Threat radius is 6.
High-ground defense	Surviving high-ground military unit-ticks within radius 6 of threatened own assets / all defensive military unit-ticks. Count each unit-tick once.
Downhill combat frequency	Ticks with downhill unit-target fire / ticks with any outgoing unit-target fire.
Uphill focus fire	Uphill ticks with at least two distinct attackers hitting the same unit / uphill ticks with at least two distinct attackers.
Ascent conversion	Eligible military attacker–plateau instances reaching the target plateau alive within ten ticks / eligible instances.
Post-ascent pressure	Eligible ascents followed by the attacker firing at an enemy unit or structure on that plateau within the next five ticks / eligible ascents.
Uphill assault losses	Eligible instances ending in combat-attributed attacker death before ascent within ten ticks / eligible instances.
Foothold retention	Eligible ascents with the attacker alive on the target plateau five ticks later / eligible ascents. Uninterrupted occupancy is not required.
Unit matchups: 6	
Favorable, unfavorable, and neutral attacks (3)	Executed basic-unit impacts where own attacker counters the enemy target, is countered by it, or has the same type, respectively / all eligible outgoing basic-unit impacts.
Favorable, unfavorable, and neutral exposure (3)	Executed basic-unit impacts where own receiving unit counters the enemy attacker, is countered by it, or has the same type, respectively / all eligible incoming basic-unit impacts. The perspective is the own recipient, not the shooter.

Temporal features. The temporal block measures the 14 variables in Table 19 in seven ordered windows, ticks 1–20 through 121–140. These measures use the same executed-state/event definitions as Table 18. Each 20-tick value averages its available 10-tick summaries weighted by observed tick count; ratio values therefore average subwindow ratios rather than pooling event denominators. A ratio without an eligible denominator remains unavailable. Tick 140 ends the last window whose start retains at least 20% of reference games; a complete-timeline sensitivity extends to tick 250.

Table 19: Shared temporal inputs for the 1v1 and 2v2 analyses. Each of seven windows contains 14 values, giving 98 values and 98 corresponding missingness flags per point. Appendix F describes the 2v2 aggregation adjustments.

Family	Measures
Economy (4)	Attributed mining per tick, living workers, completed extractors, and bank.
Production (3)	Completed barracks, busy-barracks-tick share, and military completion value per tick.
Composition (4)	Soldier, marauder, striker, and advanced-unit (tank-plus-enforcer) shares of military completions. Unlike the six-type full-game composition features, these shares exclude scouts and workers.
Army (2)	Living military count and HP-weighted resource value.
Position (1)	Forward military unit-tick share.

Applicability flags. When a ratio lacks an eligible denominator, it is unavailable rather than zero. For example, an unavailable ascent-success value means that the player has no uphill assault in that particular game with an eligible observation window, rather than a zero success rate. Binary

flags mark unavailable (1) versus available (0) values. The 17 retained full-game flags cover absent military unit-ticks, completed-barracks-ticks, composition completions, qualifying counter requests, local repair opportunities, nearby enemy military, post-depletion ticks, and multi-shot ticks. They also cover absent defensive-target impacts, incoming and outgoing unit impacts, defensive military unit-ticks, eligible ascent instances, eligible five-tick ascent follow-up, and multi-attacker uphill ticks. The final two mark absent incoming and outgoing basic-unit matchup impacts.

Table 20 lists the 17 retained full-game binary flag columns, not all 115 flags in the combined input. A flag is 1 when its listed opportunity is unavailable and 0 otherwise. Features sharing a row retain separate numerical inputs but use one availability flag. A dagger marks distinct conditions whose flag values coincide in these data, not conditions that are equivalent in general. Constant full-game flags are excluded from this table and the full-game block. Separately, each of the 98 temporal variable–window values has a missingness flag marking an undefined ratio or an unobserved future window. All 98 temporal flags remain in the fit, including constant and duplicated patterns. All retained replays contain map terrain.

Table 20: Retained full-game 1v1 applicability flags, in saved-column order. This table covers 17 of the 115 flags in the combined input. The six composition types include scouts but exclude workers. Basic-unit impacts involve soldiers, marauders, and strikers.

Flag	Condition for value 1	Features covered
F1	No own military unit-ticks.	Forward presence, Holding / no order, Isolation
F2	No eligible ten-tick attacker–plateau instance.	Ascent conversion, Uphill assault losses
F3	No completed own barracks observed.	Barracks busy share, Affordable-idle share
F4†	No completed units among the six composition types. No ticks with living units among the six composition types.	Six unit-type shares, Production entropy, Fielded entropy
F5	No basic-unit production requests with a visible strict-majority enemy basic type.	Counter-production requests
F6	No impacts on enemy military threatening own completed CCs or extractors within radius 6.	Defensive downhill fire
F7	No attributed incoming unit-target impacts.	Downhill exposure
F8	No attributed outgoing unit-target impacts.	Downhill attack share, Downhill combat frequency, Uphill attack share
F9	No eligible ascent with complete five-tick follow-up.	Foothold retention, Post-ascent pressure
F10	No surviving own military unit-ticks within radius 6 of threatened own CCs or extractors.	High-ground defense
F11	No local repair opportunity at a damaged own CC with an adjacent own worker and bank at least 1.	Local CC repair response
F12	No eligible incoming basic-unit impacts.	Favorable, unfavorable, and neutral exposure
F13	No eligible outgoing basic-unit impacts.	Favorable, unfavorable, and neutral attacks
F14	No ticks with at least two attributed military shots.	Focus-fire proxy
F15	No own military unit-ticks within radius 6 of enemy military.	Siege targeting
F16	No ticks strictly after starting-node depletion.	Post-depletion mining
F17	No uphill-fire ticks with at least two distinct attackers.	Uphill focus fire

Scaling and applying applicability. For each numerical feature, we standardize observed values using their mean and population standard deviation across all 1,800 points. Unavailable values are set to zero in standardized space (the observed mean, not a raw zero), while flags preserve their unavailability. Genuine zeros, such as producing no advanced units, remain observed values. Within the full-game block only, we remove constant numerical columns and constant or duplicate flags, counting shared flags once. Worker and overall production flags are constant. Fielded-composition and composition-production availability coincide in these data, as do post-ascent pressure and retention availability, so each pair shares one flag. This leaves 57 numerical features and 17 distinct varying flags: 74 columns in the full-game block. No logarithm, clipping, winsorization, or preliminary dimensionality reduction is applied within this full-game block. The separate temporal block uses the transformation described below.

Numerical features share 95% of the nominal squared-distance weight, equally across the eight groups and then within each group. The 17 unstandardized flags share the other 5%. A numerical feature in group g has weight $w_j = 0.95/(8n_g)$, where n_g denotes the number of numerical features in that group. The unit-matchup group contains six ratios: favorable, unfavorable, and neutral shares

for outgoing attacks and incoming exposure. Thus $n_g = 6$ for this group, and each ratio receives weight $0.95/(8 \times 6)$. Each flag receives $0.05/17$. We scale input columns by $\sqrt{w_j}$ before computing Euclidean distances. These weights balance groups, not fixed fractions of empirical variance after imputation. Each directional matchup triplet is dependent, so the six ratios are not independent measurements.

For the temporal block, we apply $\log(1 + x)$ to the eight magnitude variables (mining, workers, extractors, bank, barracks, military production, army value, and army count), and leave ratios untransformed. Each variable uses one observed mean and population standard deviation across all observed windows, shared across time. Missing values take the observed mean in standardized space. Economy, production, composition, army size, and position receive equal nominal weights, with equal weights for variables and windows within each group. The 98 temporal values share 95% of this block's nominal weight. Their 98 missingness indicators share 5%, with weight $0.05/98$ each. Indicators mark both absent future windows and undefined ratios. They do not turn absence into a literal zero measurement. All temporal values and flags are retained, including constants and duplicate flag patterns. Of the 98 temporal flags, 29 are constant in these data. Unlike the full-game block, the temporal block does not remove these columns.

Combining the blocks. We center the weighted 74-column full-game block and divide it by the square root of its total variance across the 1,800 observations. The temporal block is centered and normalized in the same way. For normalized blocks $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{t}}_i$, the combined vector is

$$\mathbf{z}_i = \left(\sqrt{1-\alpha}\,\tilde{\mathbf{x}}_i,\ \sqrt{\alpha}\,\tilde{\mathbf{t}}_i\right), \qquad \alpha = 0.5.$$

The resulting 270 columns comprise 155 numerical values and 115 binary flags (Table 21). The two normalized blocks contribute equally to the mean squared input spread. This prevents the larger number of temporal columns from determining their contribution by count alone. Within-block correlations remain. Equal block weight does not make the measurements independent.

Table 21: Complete 1v1 input inventory. Constant/duplicate removal applies only to the full-game block. The temporal block retains all 98 numerical columns and all 98 flags, including constant and duplicated flags.

Block	Numerical values	Flags	Total columns
Full game	57	17	74
Temporal windows	98	98	196
Combined input	155	115	270

Clustering, visualization, and names. We fit k-means directly to $\mathbf{z}_i$, using $k = 10$, k-means++ initialization, 30 starts, seed 42, Lloyd updates, at most 500 iterations, and tolerance 10^{-4}. The best initialization minimizes within-cluster squared Euclidean distance. All 1,800 observations, including draws, enter the fit. Ten groups is a descriptive resolution, not an estimated optimal number of strategies. A1–A10 are numbered from left to right by each cluster's median t-SNE horizontal coordinate, with median vertical coordinate breaking ties. The numbering is fixed to the displayed embedding and does not rank performance; membership is unchanged. We name groups after inspecting their full-game feature summaries and economic/military timelines: production composition, army buildup, forward presence, mining continuity, technology, and order failures. Names are post-hoc descriptions of overlapping behavior, not labels supplied to the algorithm or inferred intentions.

For display only, scikit-learn t-SNE (van der Maaten & Hinton, 2008) embeds the same 270-dimensional inputs with Euclidean distance, perplexity 30, PCA initialization, seed 42, automatic learning rate (50), early exaggeration 12, and at most 1,500 iterations. The default Barnes–Hut approximation is used. PCA initializes the two output coordinates; it does not reduce the feature vector before fitting. The two-dimensional coordinates do not determine cluster membership. Shading uses a Gaussian-smoothed point histogram in each cluster's color. Unfilled ellipses summarize the central 85% of projected points by robust Mahalanobis distance (minimum covariance determinant, support fraction 0.8, seed 42). Both are visual guides, not assignment boundaries, confidence regions,

or calibrated global distances. Figure 13(a) contains only Swiss player-games. Bots are compared in feature space in Figure 14; neither bots nor skill games determine the Swiss clusters or win rates.

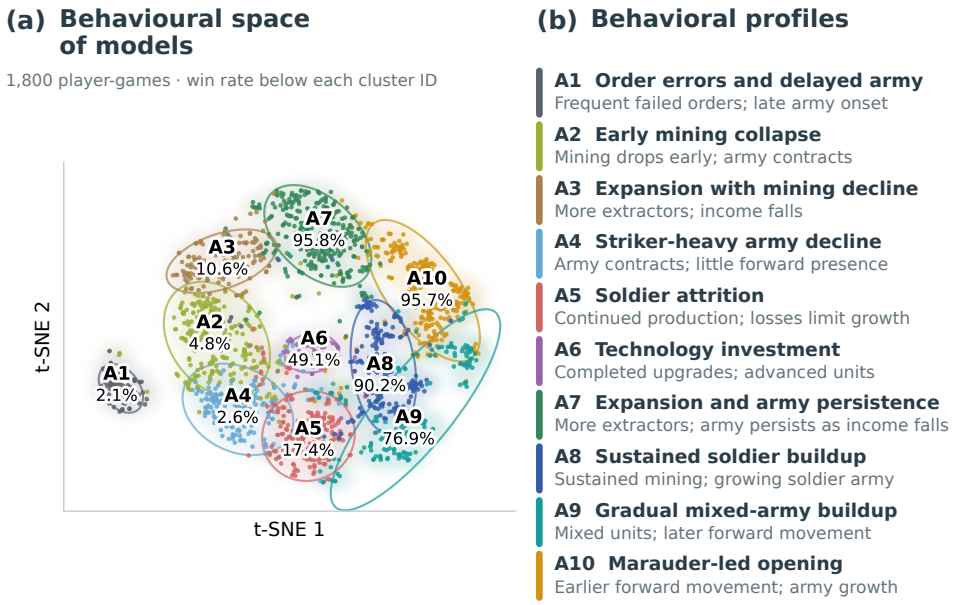


Figure 13: **Behavioral clusters and profiles.** Panels (a) and (b) of Figure 3: the t-SNE view of 1,800 player-games and the corresponding profile legend. Win rates below cluster IDs are $100W/(W + L + D)$, treating draws as non-wins. Cluster IDs do not indicate performance rank.

Bot references. Figure 14 compares the Swiss reference with 600 games between the Economy, Random, Rush, and Defensive bots on Simple: 100 games per unordered pairing, giving 300 player-game observations per bot. We apply the frozen Swiss full-game and temporal preprocessing, then measure each observation's Euclidean distance to its nearest Swiss centroid in the 270-dimensional feature space. Distances are divided by the Swiss 95th percentile (1.0714), so a value above one lies beyond that reference range. The Swiss row uses the 1,800 observations used to fit the centroids, whereas bot observations are external to the fit. This is a descriptive comparison, not a held-out calibration or confidence level. The violins use Gaussian kernel density estimates with a common bandwidth of 0.06 normalized distance units, trimmed to each sample's observed range and scaled to equal peak height. Dots and bars show empirical medians and interquartile ranges; percentages are calculated from unsmoothed observations.

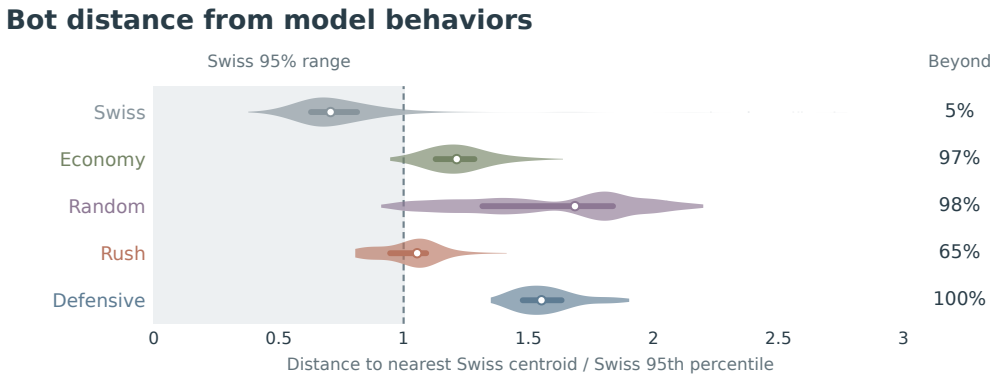


Figure 14: **Scripted bots differ from observed LLM behavior.** Distances to the nearest Swiss centroid in the fitted feature space, normalized by the Swiss 95th percentile. Shading covers distances up to that threshold; violins show distributions, dots mark medians, and bars span the middle 50%. The right column gives the percentage beyond the threshold. The reference contains 1,800 LLM player-games; each bot contributes 300 observations from 600 bot games. Maps and opponents differ between these cohorts.

The threshold is exceeded by 96.7% of Economy, 97.7% of Random, 65.3% of Rush, and 100% of Defensive observations (Table 22). Nearest-centroid labels therefore do not establish that bots share a Swiss strategy. Economy has median first military crossing at tick 77 versus tick 38 in A10 (both conditional on crossing), despite A10 being its nearest centroid in 299 of 300 games. First crossing is not an executed attack. We omit bots from the t-SNE map because proximity after projection can obscure their feature-space mismatch. Differences also reflect evaluation conditions: bots play on one map against scripted opponents (other bots), whereas the reference uses five maps and model opponents. Bots do not enter cluster win rates or the model-distribution panel.

Table 22: Nearest Swiss profiles for scripted bots. Each row summarizes 300 observations. The final column gives the percentage beyond the Swiss 95th-percentile nearest-centroid distance; these are not confident strategy classifications.

Bot	Two largest nearest groups		Beyond threshold (%)
Economy	A10: 99.7%	A9: 0.3%	96.7
Random	A2: 54.3%	A4: 16.7%	97.7
Rush	A8: 63.7%	A4: 25.7%	65.3
Defensive	A5: 64.0%	A8: 35.0%	100.0

E.2 Behavioral patterns observed in the clusters

Table 23 combines cluster sizes and win rates with economic, army, and positional development. The descriptions below also use the saved chronological replay and call audit to identify recurring behavior within each cluster. Percentages refer to player-games in that cluster. These are observed patterns, not controlled estimates of a strategy's effect on winning.

Table 23: Cluster sizes, development, and win rates. Player-games and win rates cover all cluster members, with win rate $100W/(W + L + D)$ and draws counted as non-wins. Temporal entries are means across games still observed in each indicated window: mining per tick, HP-adjusted army resource value, and forward military presence. Arrows compare fixed clock-time windows, not individual matched changes.

Cluster	Strategy	Player-games	Mining 21–40 → 61–80	Army value 41–60 → 61–80	Forward (%) 41–60 → 61–80	Win (%)
A1	Order errors and delayed army	97	10.9 → 7.6	29 → 44	3 → 2	2.1
A2	Early mining collapse	231	16.0 → 4.4	94 → 45	11 → 6	4.8
A3	Expansion with mining decline	161	16.7 → 4.5	133 → 85	17 → 11	10.6
A4	Striker-heavy army decline	192	17.3 → 15.2	125 → 88	12 → 5	2.6
A5	Soldier attrition	184	17.0 → 17.9	154 → 165	21 → 15	17.4
A6	Technology investment	53	17.2 → 18.4	166 → 197	22 → 22	49.1
A7	Expansion and army persistence	284	17.0 → 5.4	185 → 222	32 → 62	95.8
A8	Sustained soldier buildup	194	16.9 → 18.5	221 → 369	27 → 52	90.2
A9	Gradual mixed-army buildup	169	18.3 → 18.4	186 → 263	21 → 33	76.9
A10	Marauder-led opening	235	18.0 → 15.8	246 → 400	55 → 79	95.7

A1: Order errors and delayed army [2.1% W]. Frequent engine-level order errors accompany delayed military development and very little forward presence. In 94.8% of games, at least 20% of ticks contain an `order_failed` alert. The median first military unit appears at tick 51 among the 87 of 97 games with a recorded military unit. Only 4.1% develop a sustained forward army. These are gameplay execution problems, distinct from provider/API call failures.

A2: Early mining collapse [4.8% W]. The common trajectory combines sharply declining income, army contraction, and little sustained forward movement. Mean mining falls from 16.0 to 4.4 resources per tick and army value from 94 to 45. In 58.0% of games, mining stops for at least ten consecutive nonterminal ticks after the starting deposit is depleted. Only 5.6% develop a sustained forward army. These patterns can reflect earlier military losses as well as unsuccessful economic continuation.

A3: Expansion with mining decline [10.6% W]. Additional extractors are common, but military and economic continuity are weaker than in A7. At least two completed extractors coexist in 88.2% of games, while 89.4% have a ten-tick post-depletion mining gap. Mean army value falls from 133 to 85 and forward presence from 17% to 11%. Only 25.5% develop a sustained forward army. Expansion thus does not by itself imply sustained income or battlefield presence.

A4: Striker-heavy army decline [2.6% W]. Striker-heavy production accompanies a contracting army rather than lasting forward presence. Mean army value falls from 125 to 88 and forward presence from 12% to 5%. Command-centre repair orders recur in 79.2% of games, meaning orders on at least five ticks, including travel rather than verified HP restoration. Only 12.5% develop a sustained forward army. Reinforcement and repair activity therefore coexist with poor army retention in this group.

A5: Soldier attrition [17.4% W]. Continued production does not consistently translate into a larger surviving army. In 67.4% of games, at least four military units are completed and soldiers make up at least 75% of those completions. Mining remains near 17–18 resources per tick, but army value grows only from 154 to 165 while forward presence falls from 21% to 15%. Repeated command-centre repair orders occur in 72.3%, consistent with recurrent pressure near the base.

A6: Technology investment [49.1% W]. Upgrades and advanced units distinguish this smaller, mixed-outcome group. An upgrade is observed in 83.0% of games, and tanks or enforcers are completed in 81.1%. Mean army value increases from 166 to 197, but forward presence remains near 22% across the displayed windows. The 49.1% win rate shows that technology investment occurs in both winning and losing trajectories, rather than identifying a uniformly successful profile.

A7: Expansion and army persistence [95.8% W]. Expansion frequently coexists with an army that remains forward despite declining income. At least two completed extractors coexist in 90.1% of games, and 95.4% develop a sustained forward army. Mean mining falls from 17.0 to 5.4, yet army value increases from 185 to 222 and forward presence from 32% to 62%. Army persistence, rather than extractor construction alone, distinguishes this high-win-rate group from A3.

A8: Sustained soldier buildup [90.2% W]. Soldier production combines with sustained mining and increasing army size. Soldier-dominated production, using the A5 definition, occurs in 90.7% of games, and 98.5% develop a sustained forward army. Mining increases from 16.9 to 18.5 resources per tick, army value from 221 to 369, and forward presence from 27% to 52%. Compared with A5, a similar unit emphasis accompanies much stronger army accumulation and forward presence.

A9: Gradual mixed-army buildup [76.9% W]. Mixed basic units accumulate while income remains relatively steady. In 62.1% of games, at least two of soldiers, marauders, and strikers are completed by tick 50. Mean mining stays near 18 resources per tick, army value grows from 186 to 263, and forward presence rises from 21% to 33%. A sustained forward army eventually occurs in 88.2%. The recurring pattern is gradual growth with mixed units, rather than the contraction seen in A4.

A10: Marauder-led opening [95.7% W]. Early marauder production commonly accompanies army growth and sustained forward movement. In 75.7% of games, at least three military units are completed by tick 50 and at least half are marauders. A sustained forward army occurs in 99.1%: at least three military units, at least half beyond the starting-base midpoint, for ten consecutive ticks. Mean army value increases from 246 to 400 and forward presence from 55% to 79% across the table's windows.

E.3 Model-level strategy distributions

Figure 15 shows how many of each model's 100 retained player-games belong to each cluster, with the win rate for those games underneath. Model rows are ordered by final Bradley–Terry rank. Columns follow cluster IDs, which are not performance ranks. Blank cells indicate no games, and each model's game counts sum to 100. Win rates are calculated separately for each model–cluster combination, counting draws as non-wins. Membership describes an observed trajectory, not a deliberate choice of strategy.

Strategy counts and win rates by model

Games above · Win rate below · Draws count as non-wins

		A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
01	GPT-6 Astra		1 0%	1 0%				1 100%		4 50%	93 100%
02	Claude Fable 5.1		3 0%	12 16.7%	5 0%			23 100%	19 100%	26 100%	12 100%
03	Claude Opus 5		5 0%	8 25%	10 0%			25 100%	4 100%	18 88.9%	30 96.7%
04	Gemini 3.7 Flash		5 20%	6 0%	17 0%	1 0%		22 100%		16 75%	33 93.9%
05	GPT-5.6 Sol		14 0%	8 0%	20 0%	3 0%		20 95%		19 73.7%	16 100%
06	Grok 4.6		11 0%	17 17.6%	16 0%	4 50%		9 100%	5 100%	31 71%	7 100%
07	GLM 5.3		18 11.1%	4 0%	18 5.6%	13 7.7%		18 100%	22 100%	5 80%	2 100%
08	Kimi K3		11 0%	13 23.1%	21 9.5%	8 25%	13 46.2%	15 100%	16 93.8%	2 100%	1 100%
09	Claude Sonnet 5		15 0%	12 0%	3 0%	23 21.7%	1 100%	17 100%	29 96.6%		
10	Gemini 3.1 Pro		15 6.7%	10 10%	13 0%	15 13.3%	5 80%	17 94.1%	7 100%	9 66.7%	9 88.9%
11	Qwen3.8-2.4T	1 0%	16 0%	10 10%	15 6.7%	11 36.4%		15 93.3%	21 85.7%	1 100%	10 90%
12	GLM 5.3 Flash		13 7.7%	16 0%	10 0%	13 30.8%	15 53.3%	20 100%	10 100%	2 100%	1 100%
13	GPT-5.6 Terra		15 6.7%	9 11.1%	14 0%	3 0%	13 30.8%	23 91.3%		6 50%	17 70.6%
14	GPT-5.6 Luna		32 6.2%	14 7.1%	8 12.5%	9 22.2%	5 60%	17 100%	10 100%	1 100%	4 100%
15	DeepSeek V4 Pro	1 0%	26 7.7%	8 0%	13 0%	12 25%	1 0%	8 100%	18 94.4%	13 61.5%	
16	Qwen3.8-27B		12 8.3%	7 14.3%	4 0%	30 10%		22 81.8%	25 60%		
17	DeepSeek V4 Flash	3 0%	19 0%	6 33.3%	5 0%	31 6.5%		12 75%	8 62.5%	16 68.8%	
18	Claude Haiku 4.5	92 2.2%				8 25%					

Figure 15: **Behavioral-profile counts and win rates by model.** Each occupied cell shows the number of player-games above and their win rate below, using $100W/(W + L + D)$. Each row contains 100 games and follows final Bradley–Terry rank. Colors identify profiles in Figure 13, with darker cells indicating larger counts. Blank cells have no games, not a zero win rate.

E.4 ASTRA BEHAVIOR ANALYSIS

Astra's four losses show how an opponent can interrupt the opening that usually develops into A10, rather than four games in which it starts with a different plan. It wins 96 of 100 games, including all 93 assigned to A10 (Marauder-led opening). In each loss, marauders still account for 80–100% of military completions through tick 50. We examine the observations, available reasoning, submitted orders, and executed replay events to follow how this similar opening leads to different full-game profiles.

When the opening develops into A10. In its A10 wins, **Astra** preserves the combination of a marauder-led opening, army reinforcement, and sustained forward pressure because opponents do not successfully shut that combination down. This does not mean that they never attempt a counter. **Fable** produces strikers in all six of its losses to A10 **Astra**. In the King of the Hill example (game 83), it completes seven strikers, but **Astra** retains its marauder core, adds soldiers, and sustains a forward army from ticks 65–95. The distinction is therefore a counter that breaks the opening's continuation, not simply recognition of the unit matchup. A10 accounts for 93 of **Astra**'s 96 wins. Its other three wins belong to A7 (one) and A9 (two), so leaving A10 is not itself sufficient for defeat.

Fable: observation-driven counter-production. In all three wins, **Fable**'s observations reveal **Astra**'s marauders, and its reasoning explicitly identifies strikers as the counter. It then issues striker-production orders, with completions confirmed in the replays. Strikers deal 20% bonus damage to marauders, while marauders deal 20% less damage to strikers. The resulting engagements interrupt **Astra**'s forward pressure, although the subsequent trajectories differ across maps.

In Map Control (game 66), **Fable** observes a marauder in the tick-37 state, orders a striker at tick 38, and pulls its soldiers back rather than taking the unfavorable engagement. It completes four strikers by tick 50 and seven by tick 58. **Astra**'s initial sustained forward presence ends at tick 59, and its remaining marauders are eliminated by tick 75. Despite completing a second extractor at tick 32, it cannot maintain its army or income. Its full-game profile is consequently A3: Expansion with mining decline.

In Simple (game 78), **Fable** orders its first striker at tick 37 after explicitly discussing the observed marauders. Three strikers are completed by tick 45, and four of **Astra**'s first five marauders are dead by tick 55. **Astra** adds soldiers and later rebuilds its marauder force, while **Fable** shifts toward marauders of its own. This becomes a longer contest with changing compositions, not an uninterrupted striker rush. **Astra**'s resulting profile is A9: Gradual mixed-army buildup.

In Mounds (game 89), **Fable** similarly orders strikers after identifying marauders, completing six by tick 50. Its tick-48 orders concentrate pairs of strikers on individual marauders. **Astra** starts adding soldiers at tick 49, but all four opening marauders are gone by tick 58. **Fable** explicitly recognizes the change to soldiers and orders marauders at tick 60 to counter them. **Astra** later turns to strikers, ending in A9 rather than preserving the A10 trajectory. These exchanges show both the initial counter and continued adaptation to the opponent's production.

Gemini Flash: better exchanges with the same units. In the King of the Hill loss (game 448), both players produce only marauders as military units through tick 50. **Gemini Flash** gains its initial advantage through micromanagement rather than a different unit composition. Its reasoning at ticks 43–44 explicitly considers the observed marauders' positions, attack range, downhill damage bonus, and concentrated fire. The corresponding orders direct three marauders onto one target and then retarget the next. Replay events confirm that the two **Astra** marauders die at ticks 43 and 45. In each exchange, two of **Gemini Flash**'s attackers fire from high ground onto low ground, receiving the 25% terrain bonus.

By tick 60, **Gemini Flash** has killed six enemy marauders while losing one, despite completing only four against **Astra**'s seven. It uses the surviving force to contest **Astra**'s expansion and pressure its economy. **Astra** never establishes a sustained forward army and later has no mining for ticks 73–98. Its full-game profile is A2: Early mining collapse. **Gemini Flash** adds strikers only from tick 71, after the marauder engagements have already established its advantage.

Across these losses, the departure from A10 describes what happens when the usual opening is disrupted: expansion without military continuity in A3, reactive mixed-army rebuilding in A9, or army and economic collapse in A2. The cluster label summarizes the resulting game, not a strategy selected before play. The same marauder-led opening can therefore remain A10 when opposing responses fail to stop its development, or yield another profile when an opponent successfully counters its composition or outplays its units.

F STRATEGIC PATTERNS IN THE 2V2 TEAM GAMES

F.1 DATA AND ANALYSIS PROTOCOL

We apply the full-game and temporal analysis of Appendix E.1 to all 72 Team-Chat Evaluation games. Each of three rosters supplies 24 matches: beginner (**Haiku 4.5** with **DeepSeek V4 Flash**), intermediate (**Sonnet 5** with **Gemini 3.1 Pro**), and expert (**Astra** with **Fable 5.1**). Every match pairs chatting and non-chatting copies of the same roster. Rosters do not compete against one another. The audit covers 9,919 simulation ticks, 37,531 player-turn call files, and 17,276 messages. Observations remain private. Unions of explored areas used in analysis do not imply that agents share observations.

The analysis contains 144 *team-game* points (71 wins, 71 losses, two draws). Each roster supplies 48 team points, equally divided between chat settings. These points represent 72 matches, not independent trials.

Full-game summary features. We use the same 57 full-game summary features defined in Appendix E.1, Table 18, with team aggregation. We add 17 2v2 features: opening milestones (3), coordination (11), and three further terrain measures, defined in Table 24. Coordination remains a single full-game summary block. In addition to the labels excluded in Appendix E.1, roster, chat setting, and messages are used only after fitting.

Team measures pool underlying counts and opportunity denominators rather than averaging individual ratios. Composition uses pooled units, while technology and upgrades are counted per owner and summed. Exploration uses the union of members' explored tiles, isolation includes nearby allied units, and forward presence uses the line between initial team-base centroids. Command measures divide by active member-turns, and post-depletion mining divides by pooled member-ticks after each member's starting node depletes.

Table 24: The 17 additional full-game 2v2 summary features. These augment the 57 shared measures in Table 18. Parenthesized counts denote separate features.

Feature	Definition
Opening milestones: 3	
First extractor, first barracks, and first military (3)	First completed structure or first living military-unit tick. Team value is the mean of both members' first ticks and is unavailable unless both reach the milestone.
Coordination: 11	
Joint army availability	Fraction of all game ticks with living military from both teammates.
Army-readiness gap	This is the absolute difference between teammates' first military-unit ticks. It is unavailable unless both field military.
Military production imbalance	$\lvert A - B\rvert/(A + B)$, where A, B are teammates' completed military resource costs. Unavailable if neither produces military.
Fielded army imbalance	Mean of $\lvert A - B\rvert/(A + B)$ for living HP-weighted military values over team-army-present ticks.
Composition specialization	This is the base-2 Jensen–Shannon divergence of teammates' completed six-type distributions. A value of 0 means identical shares, while 1 means disjoint types. Both must produce eligible units.
Ally-army proximity	Military unit-ticks within radius 4 of the other teammate's military / own military unit-ticks. Team values pool both members.
Joint combat participation	Ticks with military shots from both teammates / ticks with any team military present or firing.
Same-target joint firing	Ticks when both teammates fire at the same target tile / ticks with military fire from both. Positional proxy, not verified tactical intent.
Same-opponent joint firing	Ticks when both teammates fire at the same enemy player / ticks with military fire from both. Ambiguous target ownership is not credited.
Defensive support	Threatened own-base ticks with teammate military within radius 6 / threatened own-base ticks. A threatened base is a surviving completed own command centre with enemy military within radius 6.
Unprotected base pressure	Threatened own-base ticks without *any* friendly military within radius 6 / threatened own-base ticks. Teams pool base-tick opportunities.
Additional terrain measures: 3	
Teammate-covering downhill fire	Downhill impacts on enemy military threatening the other teammate's surviving command centre or completed extractor / all impacts on those threats. Threat radius is 6.
Joint downhill fire	Ticks when both teammates fire downhill at enemy units / ticks when either does.
Coordinated ascent	Eligible, non-overlapping ten-tick team assault windows in which military from both teammates reaches the same contested plateau within five ticks of each other / eligible team assault windows. One-sided attempts remain opportunities. Incomplete/censored windows are excluded.

Temporal features. The same 14 variables and seven windows in Table 19 are used here, with no additional temporal features. Economy and army counters sum both teammates, and each ten-tick

ratio pools member counts and opportunity denominators before the shared window aggregation. Forward presence uses the team-base centroids defined above. After a member is eliminated, its living counters contribute zero while its teammate continues.

Applicability flags. Shared full-game flags follow Table 20, evaluated over pooled team opportunities. Table 25 lists only the seven team-specific retained flags, labeled TF1–TF7. Team milestones require both members to reach the event. The full-game block retains 17 flag columns in total: ten shared and seven team-specific. The temporal block uses the same 98 variable–window flags as in Appendix E.1.

Table 25: Team-specific retained full-game applicability flags. Shared flags are covered in Table 20 and omitted here. Each flag is 1 when its condition holds and 0 otherwise. The dagger marks distinct conditions whose flag vectors coincide in these games.

Flag	Condition for value 1	Features covered
TF1	No impacts on enemy military threatening a teammate's completed CC or extractor within radius 6.	Teammate-covering downhill fire
TF2	No eligible ten-tick team assault windows.	Coordinated ascent
TF3	No own-base ticks threatened by enemy military within radius 6.	Defensive support, Unprotected base pressure
TF4	At least one teammate never fields military.	First military timing, Army-readiness gap
TF5†	At least one teammate never completes a barracks. At least one teammate completes no units among the six composition types.	First barracks timing, Composition specialization
TF6	No ticks when either teammate fires downhill at a unit.	Joint downhill fire
TF7	No ticks with military fire from both teammates.	Same-opponent joint firing, Same-target joint firing

Scaling and applying applicability. We apply the same observed-value standardization, missing-value imputation, and flag handling as Appendix E.1, fitted independently on the 144 team points. All 74 full-game numerical features remain. The only weighting change is that the two additional groups give ten numerical groups, so $w_j = 0.95/(10n_g)$, where n_g is the number of features in the group. The terrain group now has 14 features, while $n_g = 6$ remains unchanged for unit matchups. The 17 retained full-game flags share the other 5%. Temporal transformations and weights are unchanged.

Combining the blocks. We use the same block centering, normalization, and equal combination ($\alpha = 0.5$) as Appendix E.1, fitted to this team population. The combined input has 287 columns: 172 numerical values and 115 flags (Table 26).

Table 26: Complete team-level input inventory. The temporal block is the same size as in the 1v1 analysis.

Block	Numerical values	Flags	Total columns
Full game	74	17	91
Temporal windows	98	98	196
Combined input	172	115	287

Clustering, visualization, and names. We independently fit the 144 team points using the same k-means and display-only t-SNE settings as Appendix E.1: $k = 10$ in the 287-dimensional input space and perplexity 30 for visualization. All points, including draws, enter the fit. We assign B1–B10 along the shortest open path through the mean cluster positions in the t-SNE plot, starting from the leftmost endpoint. This makes consecutive labels spatially close without changing the fitted groups. Profile names follow the same post-fitting inspection as in the 1v1 analysis. These IDs are not performance ranks or counterparts of A1–A10.

Figure 16 shows the embedding and profile legend. The team display uses covariance ellipses with semiaxes of two standard deviations and a small circle for the singleton cluster. The fitted t-SNE has KL divergence 0.299, trustworthiness 0.951, and 71.9% neighbor overlap, with the latter two measured at 15 neighbors.

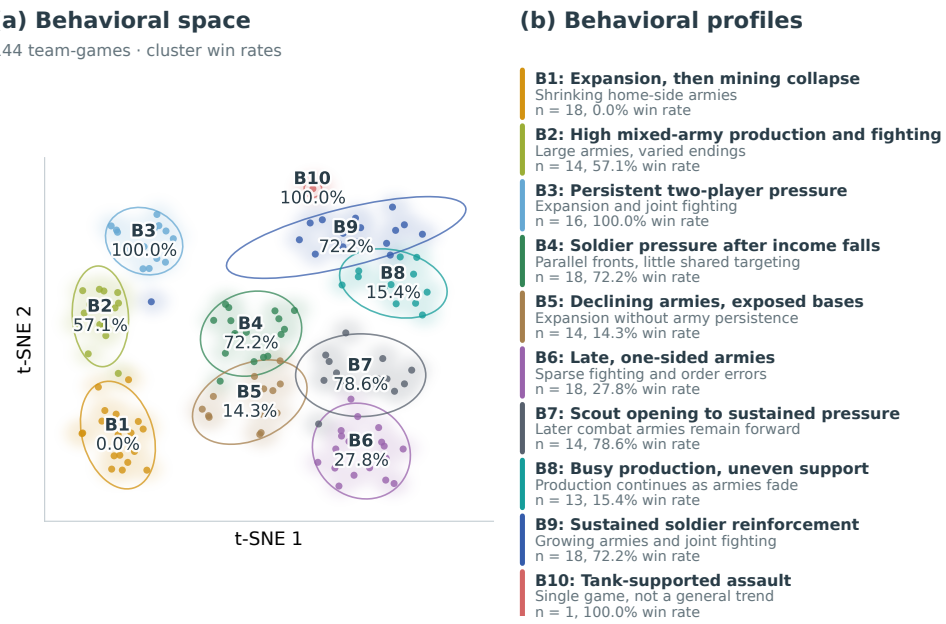


Figure 16: **Team behavioral clusters and profiles.** Panel (a) shows the 144 team-game points colored by their feature-space clusters. Soft density shading and ellipses summarize projected spread, with a circular outline for the singleton cluster. Panel (b) names the profiles and gives their sample sizes. Win rates use the same draws-as-non-wins convention as Figure 13.

F.2 Behavioral patterns observed in the clusters

Table 27 contrasts economic continuity with the ability to retain and deploy military forces. Falling income can accompany either sustained pressure or military collapse.

B1: Expansion with mining collapse and home-side armies [0.0% W]. All 18 games contain unprotected base pressure, and 17/18 contain sustained one-sided army contribution. Weak protection and uneven military support help explain why expansion coexists with declining income and shrinking, mostly home-side armies.

B2: High mixed-army production and fighting [57.1% W]. Military production averages 20.7 resources of unit value per tick, and repeated joint combat occurs in 13/14 games. Continued reinforcement and participation by both teammates support the observed army buildup and increasing forward presence.

B3: Expansion with persistent two-player pressure [100.0% W]. Every team expands (16/16), and both teammates have military units for 74.2% of game ticks on average. Economic continuation is accompanied by a persistent two-player force, consistent with the growing army value and forward presence in the table.

B4: Soldier pressure sustained after income falls [72.2% W]. Soldiers make up 91.2% of produced non-worker units, and all 18 games develop sustained forward pressure. The soldier force keeps advancing as income falls, so a smaller surviving army can still exert greater forward pressure.

B5: Expansion with declining armies and exposed bases [14.3% W]. Expansion occurs in 12/14 games, but all 14 contain unprotected base pressure. Additional extractors do not secure economic or military continuity when bases remain exposed, consistent with falling mining and army value without greater forward presence.

B6: Late, one-sided armies with order errors [27.8% W]. Mean first military timing is tick 52.7 among teams where both members field units, and both teammates have armies for only 16.8% of game ticks. Delayed, limited two-player participation leaves little military strength to sustain pressure as income falls.

Table 27: Team cluster sizes, development, and win rates, following Table 23. Team-game counts and win rates cover all cluster members. Temporal entries average available team points in each window, with mining and army counters including an eliminated member's zeros while its teammate continues.

Cluster	Strategy	Team-games	Mining 21–40 → 101–120	Army value 41–60 → 101–120	Forward (%) 41–60 → 101–120	Win (%)
B1	Expansion with mining collapse	18	34.9 → 0.8	260.6 → 86.1	12.5 → 8.3	0.0
B2	High mixed-army production and fighting	14	36.8 → 17.9	465.6 → 669.8	27.4 → 62.1	57.1
B3	Persistent two-player pressure	16	35.3 → 9.0	394.4 → 477.9	33.3 → 75.5	100.0
B4	Soldier pressure after income falls	18	32.2 → 1.4	475.3 → 324.3	23.0 → 83.1	72.2
B5	Declining armies and exposed bases	14	34.2 → 2.5	357.1 → 124.0	20.3 → 20.8	14.3
B6	Late, one-sided armies and order errors	18	28.8 → 2.3	168.7 → 140.1	21.7 → 16.7	27.8
B7	Scout opening to sustained pressure	14	32.1 → 4.4	258.0 → 366.2	21.2 → 87.5	78.6
B8	Busy production with fading support	13	29.3 → 19.7	262.9 → 306.1	35.1 → 25.5	15.4
B9	Sustained soldier reinforcement	18	32.1 → 25.7	354.0 → 892.4	23.7 → 53.9	72.2
B10	Single-game tank-supported assault	1	27.2 → 35.6	362.0 → 1083.1	42.2 → 45.0	100.0

B7: Scout-heavy openings that develop sustained pressure [78.6% W]. Both teammates have military units for 49.7% of game ticks, and all 14 games develop sustained forward pressure. The scout-heavy opening is followed by a usable combat force that continues advancing despite declining income.

B8: Busy production with fading, uneven army support [15.4% W]. Barracks are busy for 79.5% of completed-barracks-ticks, yet all 13 games contain sustained one-sided army contribution. Production remains active, but uneven support accompanies only modest net army growth and declining forward presence.

B9: Sustained soldier reinforcement and joint fighting [72.2% W]. Barracks are busy for 69.1% of completed-barracks-ticks, and repeated joint combat occurs in all 18 games. Continued production translates into a growing army that both teammates use in combat, matching the table's rise in army value and forward presence.

B10: Single-game tank-supported assault [100.0% W]. Tanks constitute 16.7% of produced non-worker units, and military production averages 16.6 resources of unit value per tick. This single game combines expensive units with continued reinforcement, explaining the strong rise in army value without a comparable increase in forward share.

F.3 Roster-level strategy distributions

Figure 17 separates each roster by chat setting and shows game counts with their within-profile win rates. Both settings use the same fitted team clusters.

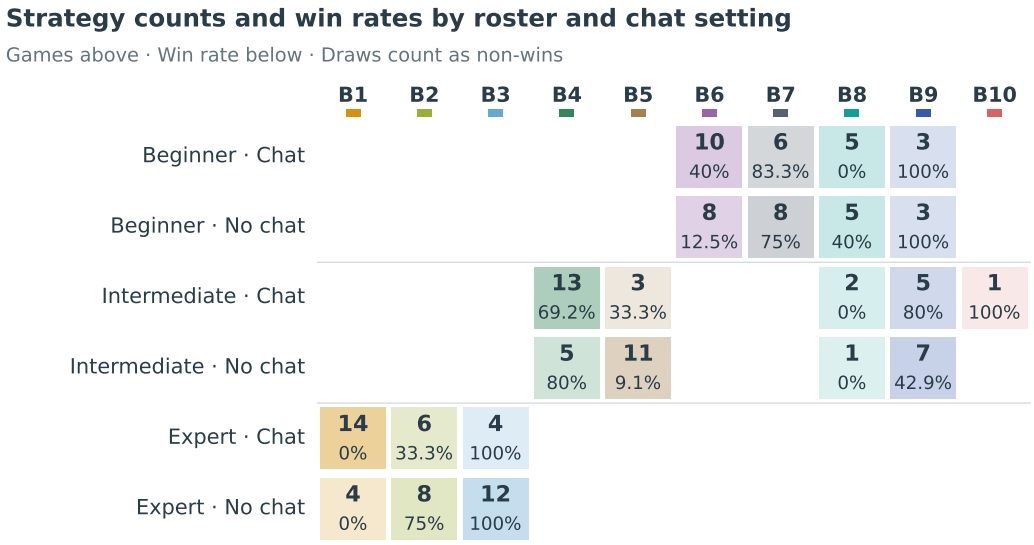


Figure 17: **Team strategy counts and win rates by roster and chat setting.** Each occupied cell shows the number of team-games above and their win rate below, using $100W/(W + L + D)$. Each row contains 24 games. Colors identify profiles in Figure 16, with darker cells indicating larger counts. Blank cells have no games. Each roster plays only its own counterpart, so the rows do not rank rosters against one another.

F.4 Communication, coordination, and outcomes

The overall chat-side records are 12–12–0 for the beginner roster, 15–8–1 for intermediate, and 6–18–0 for expert. Each is based on 24 matches against its non-chatting counterpart. Figure 17 separates strategy frequency from within-strategy team win rates. All Chat/No chat comparisons below refer to the same retained matches.

Paired coordination comparisons. Table 28 subtracts the non-chatting team's feature from the chatting team's feature within each match, then averages the differences. This controls the roster and map of each pair, but does not isolate why a team won. Joint army availability measures simultaneous military presence. Joint combat measures ticks when both teammates actually fire. Same-target firing is conditional on joint-fire ticks and uses a common target tile, not verified tactical intent.

Table 28: Mean paired Chat minus No chat differences in percentage points. Parentheses give the number of pairs where both values are available. Positive imbalance means less evenly distributed surviving military value. Missing same-target opportunities are excluded, not imputed as no coordination.

Roster	Forward	Joint army	Joint combat	Same target	Imbalance
Beginner	-6.7 (24)	+4.3 (24)	-4.0 (24)	+7.5 (13)	-6.1 (24)
Intermediate	+15.7 (24)	+11.9 (24)	+8.4 (24)	+13.5 (24)	-14.9 (24)
Expert	-28.5 (24)	-11.4 (24)	-10.2 (24)	-8.0 (22)	+11.7 (24)

The conversations help explain why the same communication channel has different effects across rosters. The examples below link messages to actual orders and replay states. B labels describe the chatting team's full-game behavior.

Beginner: waiting for an army that is not ready. For **Haiku** and **DeepSeek Flash**, a recurring problem is unreliable readiness information. In the Map Control loss (beginner game 17, B6), **Haiku** says "Hold position, don't overextend" at tick 37. **DeepSeek Flash** replies "Will hold till

you join" and redirects two soldiers to a staging position. However, **Haiku**'s repeatedly promised reinforcements do not exist yet: its first military unit appears at tick 66. This fits B6's late, one-sided armies. In the Simple loss (beginner game 15, B8), the mismatch persists during an attack. **DeepSeek Flash** reports "7 soldiers engaging. Need your soldiers ASAP!" while **Haiku** claims its barracks is complete, six ticks before it actually completes. Later production does not prevent the fading, uneven support characteristic of B8.

Chat works better when both players can act on the plan. In the King of the Hill win (beginner game 20, B7), both field military by tick 40. At tick 51, **DeepSeek Flash** says "Moving 3 soldiers to (20,15) to mass with you. Producing another." The resulting reinforcement and pressure fit B7, where scout-heavy openings develop into sustained attacks. These successes explain why chat is not uniformly harmful. But having two armies, or occasionally firing at the same target, does not mean they advance and fight together consistently. Waiting and defensive regrouping help explain why the chat-enabled beginner team has better army balance but lower forward presence and joint combat than its no-chat counterpart, with no overall win advantage.

Intermediate: requests become useful reinforcements. **Sonnet** and **Gemini Pro** more often communicate about forces that are already available. In the Simple win (intermediate game 12, B9), **Gemini Pro** asks "Push your army to (15, 15)!" at tick 41. Rather than waiting for its preferred five soldiers, **Sonnet** sends four at tick 43, then sends the fifth while producing another. This fits B9's sustained soldier reinforcement and joint fighting.

Useful coordination also means dividing the work. In the Map Control win (intermediate game 14, B4), **Sonnet** reports depleted income but keeps its soldiers active. It then spots undefended workers and redirects to the other opponent. **Gemini Pro** responds "good call on destroying P2's economy" and maintains its own attack. This is B4's soldier pressure sustained after income falls. The team wins without any same-target firing, showing that the benefit is not simply both players attacking one target.

The contrast is the King of the Hill loss (intermediate game 23, B5). **Gemini Pro** reports being "heavily outnumbered." **Sonnet** sends help, but diverts the first reinforcements to enemy workers while its ally's soldiers are dying. Expansion continues, yet **Gemini Pro**'s army and base fall, matching B5's exposed-base decline. Across the roster, chat more often sustains both players' participation than produces this mismatch. This supports the higher forward presence and joint combat in Table 28, without implying that every request is useful or timely.

Expert: a shared plan can weaken both armies. **Astra** and **Fable** often exchange precise tactical information, but agreement can reinforce an expensive plan. In the King of the Hill loss (expert game 24, B1), they endorse saving for additional command centres while their fighting forces fall behind. Neither proposed command centre starts. Later, **Fable** recommends defending separate plateaus, and **Astra** agrees to "stopping all pursuit". The result fits B1's mining collapse and home-side armies, not coordinated forward pressure. This is a recurring tradeoff: by tick 50, expert chat teams have produced less military than their silent opponents in 22 of 24 games.

They can also coordinate an assault successfully but lose the wider game. In the Simple loss (expert game 12, B2), the players regroup marauders and share target assignments. Their combined attack destroys an enemy command centre, but **Fable** warns "my base is lost, so your CC is the anchor." Its own base then falls, followed by **Astra**'s. B2's high mixed-army production and fighting captures this active but unsuccessful attack, rather than B1's retreat into defense.

The Map Control win (expert game 18, B3) shows a more useful adjustment. **Fable** keeps producing while **Astra** saves, and **Astra** eventually announces "switching savings to marauder reinforcements". They continue attacking while exchanging updates on home defense, fitting B3's persistent two-player pressure. The difference is therefore not an inability to communicate. Chat helps when it supports timely military commitments, but hurts when it prolongs an investment or attack that leaves an ally unable to hold. Smaller or isolated surviving forces also explain the expert roster's lower joint-army and joint-combat scores and greater imbalance.

G TEXTCRAFT: STRATEGIC DATA FOR LANGUAGE-MODEL AGENTS

TEXTCRAFT is a text-native strategy environment in which language-model agents compete in 1v1 matches or cooperate within opposing 2v2 teams. Each decision combines a structured text observation with a JSON array of orders. Agents must gather resources, build armies, scout opponents, and coordinate attacks and defense under partial observability. Outcomes depend on adapting to other agents and executing plans over time, rather than answering a fixed question.

A diverse strategic corpus. Thinking-Variant Selection, Scratchpad Selection, Swiss Leaderboard, and Team-Chat Evaluation cover 18 models, different thinking and scratchpad settings, five 1v1 maps, and three 2v2 maps with chat enabled or disabled. Their 1,462 completed games contain 357,219 player turns and 659,143 submitted gameplay actions (Table 29). Logs and replays link observations, orders, game states, and outcomes, with team messages, scratchpads, and reasoning traces where available. They expose successful counterplay, failed plans, cooperation, and breakdowns in execution.

Table 29: Recorded data volume for the four main experiments only. M denotes 10^6 tokens; MB and GB denote 10^6 and 10^9 bytes, respectively. Values are rounded independently after aggregation.

Metric	Thinking variants	Scratchpad	Swiss leaderboard	Team chat	All
Completed games	310.00	180.00	900.00	72.00	1,462.00
Game ticks	36,320.00	22,713.00	100,811.00	9,919.00	169,763.00
Player turns	72,640.00	45,426.00	201,622.00	37,531.00	357,219.00
Successful LLM turns	72,638.00	45,125.00	201,425.00	37,525.00	356,713.00
Input tokens (M)	441.61	289.85	1,255.15	291.14	2,277.76
Output tokens (M)	74.85	83.99	364.85	53.32	577.01
Total tokens (M)	516.47	373.84	1,620.00	344.47	2,854.77
Cached tokens (M)	317.32	188.26	758.15	179.35	1,443.08
Successful logs (GB)	1.96	1.37	5.91	1.21	10.45
Total response text (MB)	13.07	20.84	64.60	14.90	113.41
Total gameplay actions	131,299.00	82,348.00	374,996.00	70,500.00	659,143.00
Total response time (h)	358.96	386.70	1,856.67	240.54	2,842.87

Evaluating competition and cooperation. The data support comparisons of playing strength, cost, adaptation, and execution across opponents and settings. Partial observations allow analysis of how scouting informs counter-strategies. Team conversations can be checked against subsequent actions to distinguish useful information sharing and timely support from inaccurate reports or mutually reinforced bad plans.

Improving agents through experience. Decision traces and their consequences provide feedback for diagnosing recurring mistakes, identifying effective behaviors, and developing reusable natural-language skills. The environment also supports collecting new experience and testing whether an improved agent transfers to different opponents and situations. Thus the corpus is both a resource for studying strategic behavior and a starting point for experience-based agent improvement.

Explore interactive game replays and model traces.