

Harnessing Language for Coordination: A Framework and Benchmark for LLM-Driven Multi-Agent Control

Timothée Anne¹, Noah Syrkis¹, Meriem Elhosni², Florian Turati², Franck Legendre², Alain Jaquier², and Sebastian Risi¹

Abstract—Large Language Models (LLMs) have demonstrated remarkable performance across various tasks. Their potential to facilitate human coordination with many agents is a promising but largely under-explored area. Such capabilities would be helpful in disaster response, urban planning, and real-time strategy scenarios. In this work, we introduce (1) a real-time strategy game benchmark designed to evaluate these abilities and (2) a novel framework we term **HIVE**. **HIVE** empowers a single human to coordinate swarms of up to 2,000 agents through a natural language dialog with an LLM. We present promising results on this multi-agent benchmark, with our hybrid approach solving tasks such as coordinating agent movements, exploiting unit weaknesses, leveraging human annotations, and understanding terrain and strategic points. Our findings also highlight critical limitations of current models, including difficulties in processing spatial visual information and challenges in formulating long-term strategic plans. This work sheds light on the potential and limitations of LLMs in human-swarm coordination, paving the way for future research in this area. The **HIVE** project page, hive.syrkis.com, includes videos of the system in action.

Index Terms—Multi-agent, Strategy Games, Large Language Models, Behavior Tree

I. INTRODUCTION

The growing capabilities of Large Language Models (LLMs) have opened up new frontiers in artificial intelligence, including enabling human-AI collaboration across diverse and complex domains [1]–[3]. While much of the existing research focuses on LLMs’ proficiency in tasks like natural language understanding and generation, their potential for coordinating is a new area of exploration [4], [5]. This ability can be particularly critical in disaster response, urban planning, and strategy games, where efficient coordination of multiple agents can significantly impact the operation’s outcome.

This paper introduces a novel framework called **HIVE** (*Hybrid Intelligence for Vast Engagements*) designed to facilitate such coordination by enabling natural language-based control of thousands of agents in real-time. Leveraging the strengths of LLMs, **HIVE** translates high-level human instructions into operational plans for agent swarms. Specifically, after receiving a high-level strategy prompt from the player, **HIVE** generates a plan following a structured output. This plan assigns each unit a target position and a behavior tree that takes an action at each step based on the unit’s local observation. Fig. 1 illustrates an example where **HIVE** responds to the player’s prompt and devises a successful plan in a defense scenario.

The **HIVE** framework enables a new LLM-based benchmark that evaluates five core capabilities of multi-agent systems:

coordination, exploitation of weaknesses, adherence to spatial markers, terrain utilization, and strategic planning (Sec. IV). To our knowledge, it is the first real-time strategy game benchmark for LLMs in large-scale multi-agent systems. In this paper, we compared nine state-of-the-art LLMs. Additionally, we conduct detailed evaluations to address the following questions: How does **HIVE** scale to a larger number of units? (Sec. V-A) Does **HIVE** benefit from machine-human collaboration? (Sec. V-B) And, how does **HIVE**’s performance scale with different input modalities, e.g., a visual vs. a text description of the map (Sec. V-C)?

Our findings demonstrate that **HIVE** can execute complex multi-agent tasks successfully by combining human strategic input with LLM-powered operational planning. However, the study also reveals the current limitations of LLMs, including their sensitivity to input variations, challenges in visual-spatial reasoning, and difficulties with long-term strategic planning. These insights underline the dual nature of LLMs as both powerful tools for enhancing human decision-making and as systems requiring significant refinement for seamless integration into dynamic, multi-agent environments.

This work sets the stage for advancing human-AI collaboration in multi-agent coordination tasks, highlighting both the potential and the hurdles of deploying LLMs in such contexts. By addressing these challenges, future research should be able to refine these systems to unlock their full potential.

II. RELATED WORK

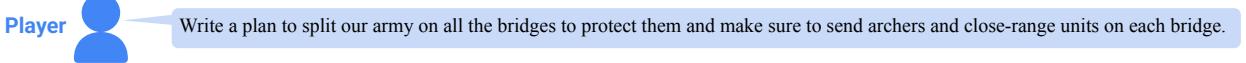
Recent works have demonstrated the potential of LLMs and vision language models (VLMs) in complex strategic games. Cicero [3] showed that LLMs can achieve human-level performance in Diplomacy through natural language negotiation and tactical coordination. SwarmBrain [6], an agent for StarCraft II, combines high-level LLM-based strategic planning with low-level non-LLM-based tactical execution. These works demonstrate that LLMs can reason about and execute strategic decisions in game environments. However, controlling multiple units in RTS games presents unique challenges. One approach developed by Zhou et al. [7] is a hierarchical command and control architecture, tested on StarCraft, which combines high-level and low-level reinforcement learning agents to balance micro and macro management.

Using behavior trees for low-level actions with LLMs has been explored in robotics, translating the behavior tree from one domain to another [8]. However, they only focused on generating behavior trees and did not evaluate their execution.

Furthering the ability to conduct AI research in the RTS domain, the StarCraft Multi-Agent Challenge (SMAC) en-

¹IT University of Copenhagen, Copenhagen, Denmark

²armasuisse Science+Technology, Thun, Switzerland



I'll help you create a defensive plan to protect the bridges using a combination of spearmen and archers. Looking at the map, we have 10 bridges in total (5 on the Northern River and 4 on the Southern River) that need to be defended. Let's analyze our forces: Spearmen (IDs 0-349): 350 units and Archers (IDs 350-699): 350 units. We can split them into 9 groups (one for each bridge) with approximately 39 spearmen and 39 archers each.

[Gives the actual plan following a structured output] This plan:

1. First positions our forces at all bridges, with:
 - Spearmen in front (at the bridge level)
 - Archers slightly behind them (2 units back) to provide ranged support
2. Each bridge gets approximately 39 spearmen and 39 archers
3. The archers are set to attack_static to maintain their position and use their range advantage
4. The spearmen are set to attack_in_close_range to engage any enemies that get close
5. Once all units are in position (Step 1), they will defend their positions while focusing on eliminating enemy units, with the center point (150, 134) as a fallback position if needed



Fig. 1. **Example of interactions between the player and HIVE.** To win this scenario, the player and HIVE have to devise a plan to prevent the enemy units (in red) from reaching the center of the camp. To do so, the player proposes a high-level description, which HIVE takes as input to write down an actual plan using a structured output. In this example, HIVE also briefly describes the situation and the plan. Notice that even though the situational description is not always correct (ten bridges instead of nine), the actual plan is correct and wins this scenario. In this example, HIVE uses claude-3-5-sonnet-20241022 as the underlying LLM. Snapshots from the corresponding game are shown at the bottom.

vironment [9] established key benchmarks for multi-agent control [10], [11]. SMACv2 [12] extends this work, introducing procedurally generated scenarios requiring generalizations. Recently, JaxMARL’s SMAX [13] has enabled easy parallelization in an SMAC-like environment, lowering the barrier of entry for research into AI applied in an RTS setting. However, while these frameworks provide foundations for developing multi-agent control systems, they focus on tactical execution rather than strategic planning, leaving the integration with higher-level decision-making under-explored.

Significant recent research has focused on LLM-based multi-agent systems. A comprehensive survey discussing key aspects of agent profiling, communication, and environment interaction has also been written [14]. For example, Agent-Coord [15] is a visual interface designed for coordination strategies in multi-agent collaboration to solve a joint goal. Challenges in multi-agent LLM systems include effective task allocation, robust reasoning, and efficient memory management [16]. A key limitation across these systems is the computational overhead of LLM inference, which makes real-time applications challenging without significant architectural compromises (partly motivating hybrid systems like Swarm-Brain, in which low-level execution is done without the LLM).

Zhang et al. [5] surveyed the strategic reasoning capabilities of LLMs, highlighting their ability to understand game structures and adapt to different contexts. Kramár et al. [17] investigated how artificial agents can use communication for better cooperation in strategic games (e.g., Diplomacy). Loré and Heydari [4] analyzed how different LLMs are affected by contextual framing and game structure.

However, in sufficiently complex and dynamic environments, LLMs and VLMs struggle to perform well [18], [19], sometimes even performing worse when visual information like an image of the game map is included. LLMs also struggle with physical common-sense reasoning in 3D environments, performing worse than human children on basic spatial reasoning tasks [20]. Furthermore, long-range reasoning problems are exacerbated because these models are prone to hallucinations [21]. Although algorithms have been proposed to fortify against hallucination in multi-step reasoning by dividing up reasoning steps, the problem of long-range causal reasoning remains unsolved [21]. This is particularly problematic in strategy games, where decisions must build on each other, and errors compound over time. AndroidArena [22] is a benchmark and environment that evaluates an LLM’s ability to navigate an operating system (e.g., using a smartphone with apps), showing this to be a current limitation of LLMs.

Models also struggle to generalize beyond their training data when moving toward geospatial modalities, prompting the need for dedicated Vision-Language Geo-Foundation models [23]. This limitation is relevant for strategy games involving spatial reasoning and map awareness, where current models often fail to develop coherent long-term strategies based on terrain and unit positioning (and sometimes struggle to identify basic landmarks). This echoes recent research that shows VLMs often struggle with trivial tasks (e.g., determining if two circles intersect) and a necessity for spatial reasoning [24].

Integrating LLMs into real-time strategy games faces several challenges not fully addressed by current research. This suggests that while LLMs can engage in high-level strategic

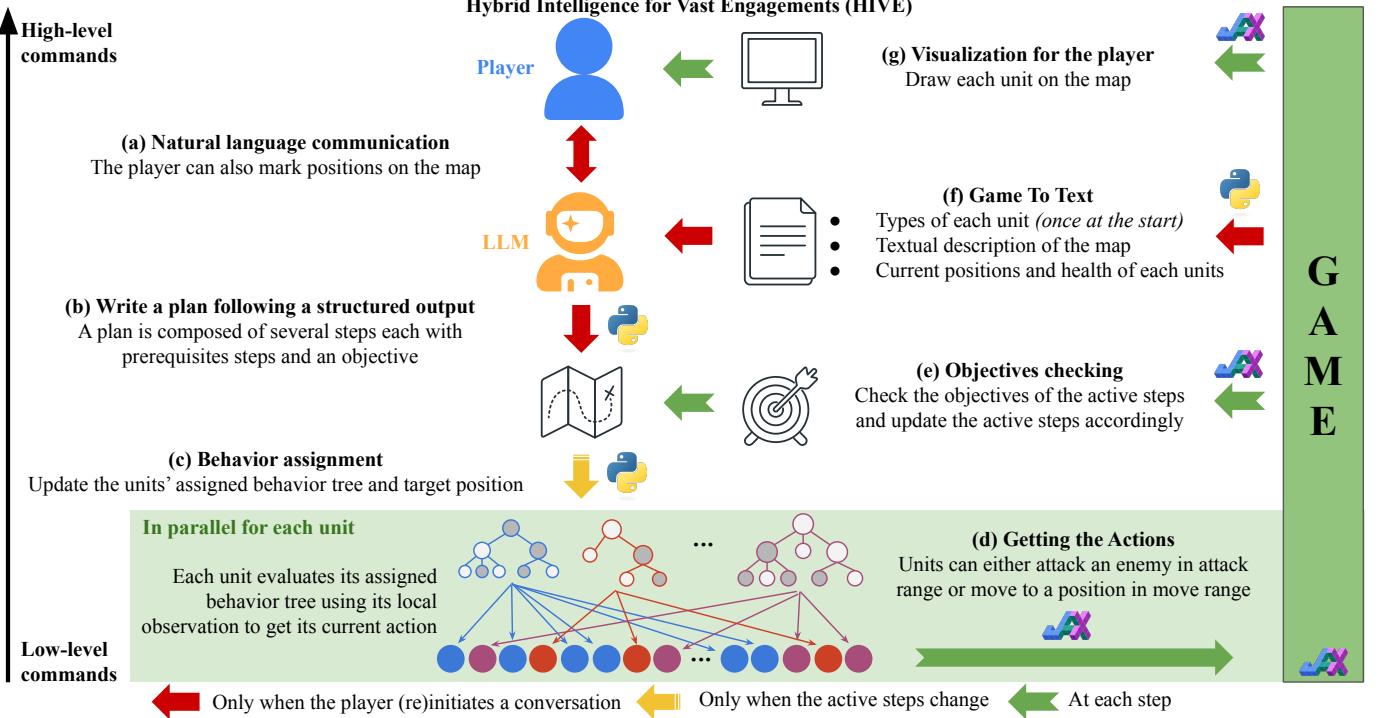


Fig. 2. Overview of the HIVE approach. HIVE enables players to command up to several thousand units by delegating the tedious task of assigning behaviors and objectives to each unit to a general-purpose LLM. HIVE operates in two main phases: (1) a discussion phase with an LLM to develop a plan before the game starts and (2) the execution phase, where the plan assigns a behavior tree to each unit. We implemented the modules that handle the units in JAX [25] and those that manage high-level information in Python. See the main text for a more detailed description.

planning, they need significant support systems to translate these plans into tactical actions.

III. THE HIVE APPROACH

Our approach, HIVE, allows a player to control thousands of units in a strategy game through a natural language LLM-based dialog. We hypothesize that this type of human-machine collaboration should alleviate some of the aforementioned shortcomings of purely LLM-based approaches.

Fig. 2 presents an overview of its main components. **(a)** The player communicates in natural language and optionally adds markers to the map to give absolute positions (Sec. III-C). **(b)** The LLM writes a plan following a structured output that controls the behavior of all units (Sec. III-D1). **(c)** The plan assigns a target position and behavior tree to each unit when initialized or triggered by the objective checking (i.e., when a new step of the plan becomes active) (Sec. III-D2). **(d)** Each unit evaluates its behavior tree using local observation to determine an action (Sec. III-B). **(e)** After each step, the game checks the current plan’s objectives (Sec. III-D3). If some are achieved, the plan is rolled forward, and the active steps are updated. This module also checks for global winning and losing conditions specific to each scenario. **(f)** At the start of the discussion, the LLM receives information about the types of each unit present (Sec. III-E). When triggered by the player, the LLM is given a precomputed textual description of the map (see Sec. V-C for a discussion of the geo-visual abilities of LLMs) and the current health and position of each

unit. (g) After each game step, the units' positions are plotted on the map and displayed on the screen (Sec. III-A6).

A. The multi-agent game

In our top-down strategy game, the player commands an army of several hundred to thousands of units against an opposing army of comparable size. The player controls their army exclusively by interacting with HIVE, which generates a plan that determines each unit's behavior at every step. The game is implemented in JaX [25], a Python library for vectorization and just-in-time compilation. It allows fast parallelization of the units' behaviors, a key component for having an RTS game with many units.

1) *Global objectives*: There are two kinds of objectives: *elimination objective* (i.e., eliminating a set of enemy units) and *position objective* (i.e., reaching a target position).

2) *The units*: Units are parameterized by their move speed, max health, attack damage, and attack range. We used three types: spearman, a slow, close-range unit with high health; archer, a long-range unit with low health; and cavalry, a fast, close-range unit with medium health. Those three types of units allow for rock-paper-scissors dynamics, as archers easily eliminate spearmen units using their longer range, cavalry units easily eliminate archer units by coming in contact faster than they get killed, and spearmen units easily win against cavalry units because of their higher health. Tab. II in the appendix details their characteristics.

3) The observation: Each unit knows the position and health of every unit inside its sight range (set to 15 m; sizes

of the different environments are detailed in Section IV). It also has access to a distance map (i.e., a matrix containing the distance to the unit's target position from anywhere on the map) to follow the shortest path to its target position. The map size in this paper ranges from 100 m to 300 m.

4) *The actions:* At each step, each unit takes one action: do nothing, move to any position in sight reachable given the unit's speed (2D continuous action), or attack an enemy unit in attack range with a line of sight (discrete action).

5) *The terrain:* We use four types of terrain:

- **Normal:** the units can move and see through it;
- **Forest:** the units can move but not see through it;
- **Water:** the units can see over it but not move through it;
- **Building:** the units cannot move or see through it.

Those allow the creation of diverse maps and scenarios with strategic features such as choke points over bridges (normal terrain over water) or stealth opportunities by hiding in forests. We generated the maps with the INKARNATE platform [26].

6) *The visualization:* At each time step, the game plots the units' positions over the map using different shades of colors (blues for the player's units and reds for the enemy units) and shapes (square for the spearmen, circle for the archers, and triangle for the cavalry). The player does not receive information about each unit's health, as it would be difficult for them to assess the health of hundreds of units [27].

In summary, the main game loop proceeds as follows:

- (a) Apply the attack action of each attacking unit;
- (b) Apply the move action of each moving unit while checking for collision with buildings or water terrain;
- (c) Push units in collision with each other;
- (d) Check for collision with a building or water terrain;
- (e) Compute the distance between each unit, considering the broken line of sight over buildings and forest terrain.

B. Behavior Trees

We use behavior trees (BTs) [28] as a middle layer between the high-level instruction of the player, written down as a plan, and the low-level actions taken by the units. They offer a convenient way to control agents while being easily designed by hand. Alternative methods are discussed in Section VI.

BTs are composed of four different types of nodes:

- **Sequence node S:** stops at the first failed node;
- **Fallback node F:** stops at the first successful node;
- **Condition node C:** evaluates a condition given the unit observation and returns failure or success;
- **Action node A:** tries to perform an action given the unit observation and returns failure or success.

For this paper, we designed a list of condition and action nodes parametrized by qualifiers (see Sec. G in the appendix for the full Lark grammar). The available conditions are:

- Is there an ally or enemy of a given type in sight?
- Is there an ally or enemy of a given type in sight that is or could be in attack range in one, two, or three steps?
- Am I or could I be in attack range in one, two, or three steps of an ally or enemy of a given type in sight?

- Am I of a given type?
- Do I have health below a 75%, 50%, or 25% threshold?
- Am I in a forest?

The available actions are:

- Do nothing;
- Move toward the target position (following the shortest path with some added noise) up to a given threshold relative to the unit sight range and move speed;
- Move toward the closest or farthest or weakest or strongest or random ally or enemy unit in sight of a given type (without optimal path guarantee);
- Attack the closest or farthest or weakest or strongest or random enemy in attack range.

We handcrafted five behavior trees available to HIVE:

- **Long-range attack:** move away from enemies, or attack an enemy if possible, or move toward the target position;
- **Close-range attack:** attack a random unit in attack range if possible or move toward the closest enemy if there is any; otherwise, move toward the target position;
- **Attack and move:** (1) attack a unit in attack range if possible or move toward the target position, and (2) only if the target is reached, move toward the closest enemy;
- **Move toward target position:** follow the shortest path to the target position with some noise;
- **Stand:** do nothing.

In all behavior trees concerning enemy units, the LLM can target any type of unit or subset of units. The exact syntax of all the behavior trees can be found in the appendix Sec. H1.

C. The language interface between the player and HIVE

The player can only control their army through an interface with HIVE. Its front end is composed of an LLM that receives natural language prompts from the player and can reply to them. HIVE gives the LLM task-specific instructions via a *System* prompt, making it easy to use different LLMs by calling a different API.

Moving many units to different positions on a map can be conveniently and quickly realized through natural language commands. For example, asking the system to send them to cover all bridges can be faster than using a traditional mouse interface. However, there are situations where it can still be helpful to communicate a specific absolute map position to the model. Therefore, HIVE also includes the ability for the player to click on the map to add labeled markers that can then be referenced in the dialog with the model.

The outer loop unfolds as follows, HIVE: (1) initializes the game state; (2) waits for the user's prompt and sends it alongside the game state to the LLM; (3) waits for the LLM to provide a plan and parses it; and (4) executes the plan if it is valid. HIVE never asks the LLM to update the plan or try again. All the plans are saved, allowing us to re-execute them anytime for debugging or visualization using the same random seed, ensuring deterministic behavior.

D. The plan

- 1) *Writing a plan with our structured outputs:* HIVE translates the player's high-level commands into a plan that

follows a structured output. We instruct the LLM (using a *System* prompt) about the expected output format to write a plan with a simple example and a short list of mistakes to avoid. The player is not required to know the plan’s format. One main advantage of HIVE is that it delegates the tedious part of writing the plan from the player to the LLM.

A plan is composed of several steps, and a step is composed of several components:

- A numerical label to refer to it;
- A list of prerequisite steps that need to be achieved before this step (e.g., waiting for two groups to reach their assigned locations before ordering them to attack);
- An objective (similar to the global objectives) that is used to infer if the step is achieved, active, or inactive;
- A list of unit groups that dictate the behaviors of each unit concerned by the step.

A unit group comprises a list of unit IDs, a target position, and a behavior tree. The step is valid only if each unit is included in no more than one group.

The actual instruction message used for Fig. 1 is detailed in the appendix Sec. E1. The plan written by the LLM is parsed, and an error is raised if it is not grammatically correct or if the units’ or behavior trees’ IDs are incorrect.

2) *Applying the plan*: When the plan is created or the active steps change, HIVE passes through each active step and assigns a behavior tree and target position to each unit. A unit keeps its previous assignment if it does not have a new one. If it never receives an assignment, it does nothing by default.

This part constitutes HIVE’s computational bottleneck since (1) each target position requires computing a breadth-first search on the map, and (2) reassigning the behavior trees and the steps’ objective checking requires a new compilation of the JAX functions. However, most plans only require a handful of steps with no change in behavior trees between each transition, thus not affecting most game steps.

3) *Checking the plan*: After each game step, HIVE checks the global and active step objectives to determine if one side is victorious, if the plan is fully unrolled, or if new steps become active. If the plan is fully unrolled, but no side is victorious, we stop the game and evaluate the plan as an early completion.

E. LLM game information

1) *Unit information*: The LLM gets a complete description of the unit types and the current composition of both sides. After each player’s prompt, it receives the units’ health and positions. An example is shown in the appendix Sec. E2.

2) *Textual description of the map*: During preliminary experiments, we explored visual map representations but found that current all-purpose LLMs are inefficient in understanding our top-down map and often cannot accurately position the units, terrain, or landmarks. Thus, we rely on a precomputed map description that gives the absolute position of the different terrain types (forest and rivers) and bridge positions. Sec. V-C compares HIVE’s abilities with the textual prompts and with different variations of inputting an image of the map. Fig. 12 and 13 in the appendix show two examples of such descriptions for two maps used in the benchmark.

IV. MULTI-AGENT CONTROL BENCHMARK

A. The five ability tests

We designed five ability tests on four scenarios (Fig. 3). All the scenarios use the same enemy units’ behaviors: the spearmen and cavalry perform close-range attacks; the archers make long-range attacks while moving away from close combat. HIVE sends a pre-written prompt (along with state-specific information) to the LLM, parses the returned plan, and executes it (if it is valid) for a maximum number of game time steps, depending on the scenario. If, after this time, no side has won, the episode concludes in a tie. As all scenarios are designed to be winnable within the allocated time steps, a tie is better than a loss but worse than actually winning the scenario. If the plan is fulfilled without achieving the global objective, we call it an early completion. In all cases, we do not ask the LLM to propose a new plan.

1) *Coordinate*: The player controls 1,000 units (500 spearmen and 500 archers) and must eliminate the opposite army of 1,000 spearmen coming from the north in less than 300 time steps. This test thus asses the ability to coordinate many units with two types in a simple terrain. With the units only able to see 15 m ahead on the 150-meter-wide map, one challenge is to spread the units on the battlefield so as not to miss any enemies. Another challenge is to place the archers behind the spearmen to minimize casualties.

2) *Exploit weakness*: The player controls 750 units (250 spearmen, 250 archers, and 250 cavalry) and must eliminate the opposing army of equal composition in less than 500 time steps. The 100-meter-wide map is divided into four quadrants by vertical and horizontal rivers crossed by bridges. The opposite army is divided by types into three quadrants. This tests the ability to exploit the rock-paper-scissors dynamics to win while minimizing casualties. This demands that HIVE estimate the opposing army battalions’ positions and unit types, dispatching the fitting counter type (HIVE is given the strengths and weaknesses of each unit type as in-context information).

3) *Follow markers & Exploit terrain*: The player controls 300 spearmen starting on the northeast corner of the 200-meter-wide map and must bring at least one to an objective position in the southwest corner in less than 500 time steps. The units must cross a bridge guarded by 1,200 opposite units (600 spearmen and 600 archers). The challenge is efficiently using the trees’ cover (the units cannot fight in the forest) to minimize casualties. We use this scenario for two ability tests: **Follow markers**, which tests the ability to follow marker positions given by the player, and **Exploit terrain**, which tests the ability to follow the player’s textual description of the path to follow through the different terrains.

4) *Strategize points*: The player controls 700 units (350 spearmen and 350 archers) and must prevent the opposite army of 900 spearmen from reaching the center of the camp (symbolized by the campfire) for up to 500 time steps. The opposite units are divided in the northeast and southwest corners of the 300-meter-wide map and are designed to split and cross over each bridge. This tests the ability to divide the army to cover strategic points.

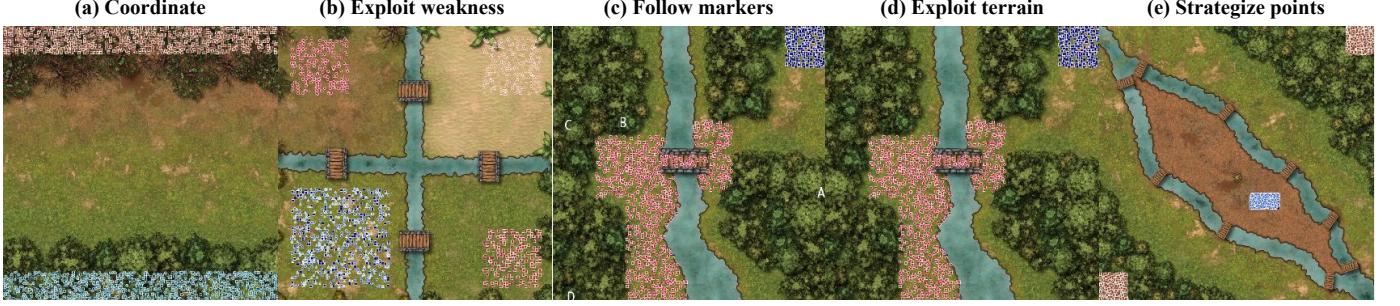


Fig. 3. **The HIVE Benchmark Ability Tests.** (a) **Coordinate** where the player has to eliminate all the enemies using 1,000 units, (b) **Exploit weakness** where the player has to efficiently use the three types of units to eliminate the enemies, (c) **Follow markers** where the player has to bring at least one unit to the south, (d) **Exploit terrain** where the player has to bring at least one unit to the opposite corner of the map, and (e) **Strategize points** where the player has to prevent the enemies from reaching the center of their camp.

B. The tested LLMs

We compare nine LLMs: eight state-of-the-art closed-source models from the OpenAI, Claude, and Gemini families, and one small open model as a baseline:

- **4o**: gpt-4o-2024-11-20 [29];
- **4o-mini**: gpt-4o-mini-2024-07-18 [29];
- **oi-mini**: o1-mini-2024-09-12 [29];
- **Sonnet**: claude-3-5-sonnet-20241022 [30];
- **Haiku**: claude-3-5-haiku-20241022 [30];
- **Gemini 2**: models/gemini-2.0-flash-exp [31];
- **Gemini-pro**: gemini-1.5-pro [31];
- **Gemini-flash**: gemini-1.5-flash [31];
- **Llama3 (8B)**: Llama3 (8B) [32].

We performed an ablation to study the effect of the LLM’s temperature on the plan’s validity and quality (see Fig. 11 in the appendix). We concluded that high temperatures increase the chance of returning an invalid plan and settled on a minimal temperature of 0 (except for **oi-mini**, which was a preview with a fixed temperature of 1).

C. Results

While experimenting, we found that the LLMs’ proposed plans were susceptible to small variations in the player prompt, i.e. just one word being different. To consider the variability, we generated ten slightly different prompts for each of the five abilities. For example, to test the **Coordinate** ability, we query each LLM with ten variations of the prompt: “*Make a plan that forms as many squads as you think necessary to cover the middle row of the map to eliminate all the enemies as quickly as possible. Ensure to place our archers so that they are protected by our close-range units.*”, resulting in ten plans (**Coordinate** row in Tab. I). The slight variations are the orders of saying “*Design a plan,*” “*Write down a plan,*” or “*Make a plan,*” using “*North/South/East/West*” or “*Top/Down/Left/Right,*” or saying “*archers units*” or “*long-distance units.*” All 50 prompts are detailed in appendix Sec. I.

Fig. 4 shows snapshots of a successful plan made by HIVE using different LLMs with the corresponding prompt from the player. The actual plans are detailed in the appendix Sec F. Tab. I presents the strict successes of HIVE for the 10 prompt variations for each ability test. Fig. 5 presents a continuous

evaluation of the performance of the plans made by HIVE. Using the percentage of enemies eliminated for the abilities (a), (b), and (e), and the distance to the objective position for (c) and (d). Fig. 6 shows for each 50 prompts the result of the plan execution in terms of wins, losses, ties, achieving the plan objectives without achieving the winning objective, and HIVE returning no plan or an invalid plan. A video showing the executed plans is available at hive.syrkis.com.

HIVE solves all ability tests with **Sonnet** for at least one prompt variation, highlighting that it effectively possesses all the proposed abilities. It can effectively split the units on a battlefield, putting the long-range units in the rear to increase efficiency. Additionally, it can recognize clusters of enemy units with specific weaknesses and exploit them. Furthermore, HIVE can follow a strategic plan to exploit the cover of the trees as much as possible. Finally, it can also split the army to cover all the strategic points that the player asks.

Concerning the LLMs’ inference time (Fig. 10 in the appendix), excluding **o1** (which uses thinking) and **Llama3 (8B)** (which was run locally on an M3 chip), the models need several seconds to respond, e.g., **4o** with 6.4 s [3.3 s, 12.2 s] (median [first quartile, third quartile]), **Sonnet** with 12.6 s [9.9 s, 14.2 s], and **Gemini 2** with 3.9 s [3.0 s, 6.2 s].

The main takeaways from this evaluation are the following:

- **Llama3 (8B)** cannot follow the plan structured output;
- As can be expected, **Follow markers** is the easiest ability test (**4o** and **Sonnet** solved it nine times out of ten and **Gemini 2** solved it every time);
- The other ability tests propose interesting challenges for LLMs as none of them got more than 50% success;
- **Sonnet** is the only LLM to succeed in all five ability tests and only presents one invalid plan out of 50 answers;
- For the continuous evaluations, **Sonnet** is also either first (in **Coordinate** and **Strategize points**) or tied-first (in **Exploit weakness** and **Follow markers**);
- **Haiku**, **4o-mini**, and **Gemini-flash** show, on average, worse performance than their full-version counterparts;
- The inference time is in the order of 1 to 10 s (see appendix Section A);
- Even state-of-the-art LLMs are sensitive to the wording of the prompts, as slight changes in the prompt result in drastic changes in the plan and the execution.

Coordinate with Sonnet: “Make a plan that forms as many squads as you think necessary to cover the middle row of the map to eliminate all the enemies as quickly as possible. Ensure to place our archers so that they are protected by our close range units.”



Exploit weakness with Sonnet: “First, analyse the situation to compute the positions and types of the enemy squads. Then, write a plan that plays on their weaknesses to split our army into three groups with specific strengths and send each one against one of the enemy squads to maximize efficiency.”



Follow markers with 4o-mini: “Move our troops to the markers A, B, C, D while ignoring enemies and then to the objective position.”



Exploit terrain with 4o: “Design a plan to move our spearmen inside the trees on the east close to the bridge, then make them run to the forest on the left, on top of the bridge to quickly regain cover. Then make them hide deep inside the forest on the left of the map. Finally, make them rush south to the south-east corner of the map inside the forest and then to the objective. Everything while ignoring enemies.”



Strategize points with o1-mini: “Split our army on all the bridges to guard them and make sure to send long-range and close-range units on each one.”



Fig. 4. Successful plans by HIVE using different LLMs for each ability test. HIVE can translate high-level commands into successful plans, from coordinating thousands of units to exploiting the terrain, enemy weaknesses, or strategic points. Blue units are the player’s units. Red units are the enemies.

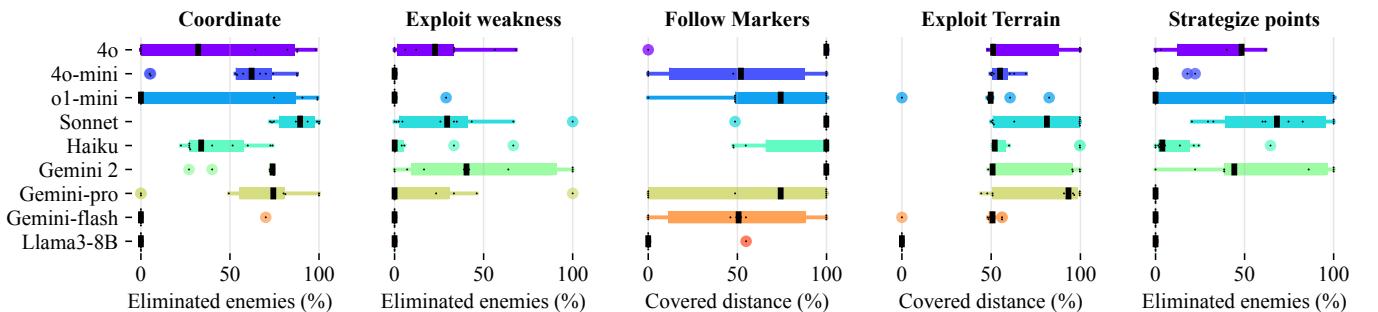


Fig. 5. Ability evaluations of HIVE using nine LLMs with the same ten prompts for each ability test. Apart from Follow Markers, the different ability tests are solvable but challenging for all LLMs. **Sonnet** performs best, while a small LLM like Llama3-8B completely fails.

TABLE I
ABILITY TESTS OF HIVE WITH DIFFERENT LLMs SHOWING THE SUCCESS OF ACHIEVING THE GLOBAL OBJECTIVE OF EACH SCENARIO.

	4o	4o mini	o1 mini	Sonnet	Haiku	Gemini 2	Gemini pro	Gemini flash	Llama 3 (8B)
Coordinate	0/10	0/10	0/10	2/10	0/10	0/10	2/10	0/10	0/10
Exploit weakness	0/10	0/10	0/10	1/10	0/10	3/10	1/10	0/10	0/10
Follow markers	9/10	3/10	5/10	9/10	7/10	10/10	5/10	3/10	0/10
Exploit terrain	3/10	0/10	0/10	4/10	2/10	2/10	3/10	0/10	0/10
Strategize points	0/10	0/10	4/10	3/10	0/10	3/10	0/10	0/10	0/10
Total	12/50	3/50	9/50	19/50	9/50	18/50	11/50	3/50	0/50

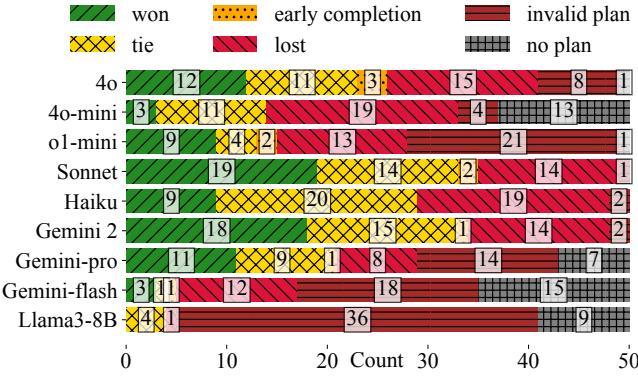


Fig. 6. The summarized result of the ability tests of HIVE using each LLM.

V. HIVE ABLATION STUDIES

A. How does HIVE scale up with the number of units?

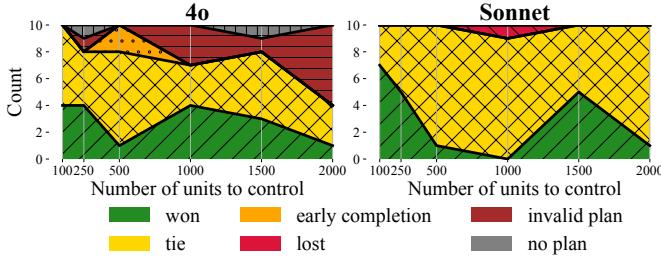


Fig. 7. Unit Scaling. Comparison of the results of the HIVE plan for the Coordinate test by varying the number of units in the game.

To see how HIVE scales with the number of units, we performed another evaluation on the **Coordinate** test, varying the number of units from 200 to 4,000. We stopped at 4,000 due to the current hardware limitation of the machine running the experiment. As the game’s main loop needs to be compiled, the size of the distance matrix between each unit grows quadratically with the number of units. For this study, we used **Sonnet**, which showed the best performance, and **4o** to see if the results generalize to a different model.

Fig. 7 shows the result of the plan made by HIVE. There is no significant conclusion regarding the success rate, as there is too much noise due to the inherent variance of the LLMs. **Sonnet**’s large number of ties is due to its plans not successfully covering the frontline, missing some enemy units, with units of both sides waiting at different locations.

Still, **4o**’s number of invalid and empty plans increases as the number of agents increases. This limitation is not observed for **Sonnet**, again highlighting its superior performance.

B. Can HIVE win alone?

An interesting question is how well HIVE performs without a human’s help. To answer this question, we use ten prompt variations: *“First, analyze the situation to find a good strategy to win this mission, then write down the corresponding plan.”* (the variations are detailed in the appendix Sec. J) on each of the four scenarios for **Sonnet** and **4o**. HIVE always describes the scenario so the LLM knows the global objective.

Fig. 8 compares HIVE with and without the player’s help. For **Coordinate** and **Exploit terrain**, both models no longer win without human help. For **Strategic Points**, **Sonnet** conserves its three wins, but both models have lower median performances. These results show that HIVE is better with human help, confirming the benefits of a hybrid intelligence approach. Additionally, when comparing the median performance, both **Coordinate** and **Exploit Weakness** seem to be easier ability tests for HIVE alone, as they only require sending units toward the enemies. In comparison, **Exploit terrain** and **Strategic Points** require exploiting the map features to win and depend on more long-term planning.

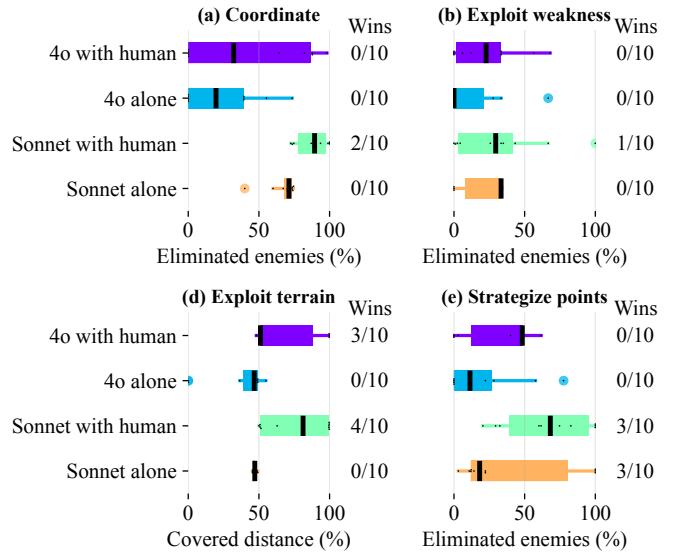


Fig. 8. Human-Machine Collaboration. Without the player’s help, HIVE’s performances and win rates decrease for both models in all ability tests.

C. Does HIVE need the textual description of the map?

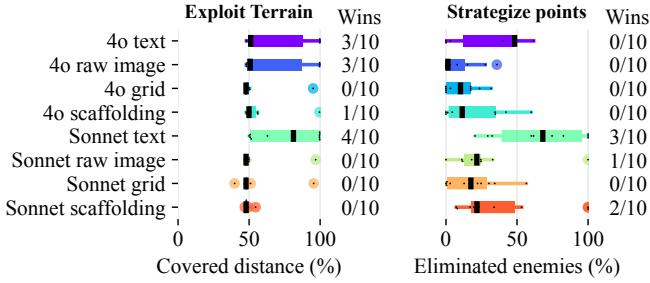


Fig. 9. **Maps description.** LLMs’ performance in using maps drops when switching from textual descriptions to images (raw, with a grid, or scaffolding).

We evaluate HIVE’s vision abilities by replacing the textual description of the map with the image. As it has to use precise target positions, we also compared two ways to improve the image-to-coordinate: a grid with an x and y axis and a scaffolding [33]. We only tested the vision on two ability tests that required it: **Exploit terrain** and **Strategize points**. The four kinds of inputs are shown in the appendix Section C.

Without the textual description, HIVE’s performance decreases (Fig. 9). For **Exploit terrain**, **4o** can exploit the raw image, while **Sonnet** fails to do so. For **Strategize points**, **Sonnet** won once with the raw image and twice with the scaffolding, with lower median performances than the three victories with the textual description.

VI. DISCUSSION

None of the closed-source LLMs were fine-tuned to the plan’s structured output or our game. Still, they returned valid and successful plans, demonstrating that HIVE can be used with an all-purpose LLM. As **Llama3-(8B)** mostly returns invalid plans, closed-source models currently achieve the best performance on the proposed benchmark.

An exciting future prospect is to take a smaller model and fine-tune it, allowing HIVE to run locally. Together with dedicated hardware, this could enable the system to be fast enough to run continuously alongside the game to (1) give real-time feedback to the player, (2) adjust the plan as the game unrolls, or (3) even run beyond real-time for fast self-play. While becoming increasingly efficient at coding in widely used programming languages, LLMs still lag behind for less common languages and specific structured outputs, mainly due to the restricted available training data [34]. Future research is still needed to propose efficient fine-tuning solutions.

The poor results of HIVE on the **Exploit weakness** task can be explained by the LLMs’ inability to extract the actual positions to target from the units’ raw positions given as context. This corroborates that LLMs struggle to process mathematical data [35], [36]. One direction is to improve LLMs’ skills for numerical analysis. Another is to design (or learn) a feature selector that summarizes the unit’s information.

The inference time of the LLMs is between four and 12 seconds. This delay is problematic for our game, where the average plan execution lasts between 10 and 20 seconds. There, we decided to only perform one plan inference per

game. However, this inference time would be less of an issue in games where the average game lasts more than a few minutes. Additionally, LLM performances are improving rapidly. For example, **Gemini 2**, which was designed for low latency, is twice as fast as **Sonnet** and only has one less victory. Another direction to decrease the inference time is to use summarization tools [37] to shrink the context size.

Following the conclusion of our temperature ablation, we decided to set the temperature to 0. A recent study of the effect of the temperature for code generation with ChatGPT [38] highlights that a temperature of 0 reduces the non-determinism but does not guarantee strict determinism. Non-determinism must be considered for tasks requiring long-term planning, such as ours. One direction for our hybrid framework is allowing the player to review the plan before execution to correct any possible mistakes.

We used a structured output format to define the plan that assigns predefined behavior trees to each unit. One alternative is to generate code directly, as it has been done for games like Minecraft [39]. We hypothesized that focusing on a simple, structured format for a plan using behavior trees would allow for easy interpretation of both the high-level plan and the low-level behavior of each unit. In contrast, understanding the code generated by the LLM could be difficult for non-programmers. In addition, having a structured format allows us to constrain the available strategies and behaviors more easily.

Incorporating findings of recent research focusing on improving LLMs’ interaction in a GUI environment [40] and geospatial reasoning [41] could further improve HIVE’s ability to handle multi-modal inputs from the player and the game interface. A recent benchmark for using LLM based on vision for simple adversarial games (e.g., Tic-Tac-Toe, Reversi, Gomoku, and Chess) shows that current models struggle to perform well [42], highlighting the need for further research.

Our benchmark results show limitations of current LLMs; nonetheless, it also discriminates between the models, as Claude Sonnet 3.5 and Gemini 2.0 are less impacted than their GPT-4o. Highlighting the rapid progress in the field, just before the publication of this article, we ran a preliminary comparison on the just-released Sonnet 3.7. This model passes the 50% success mark with 26 victories out of 50, emphasizing the need for the community to build benchmarks such as ours to keep highlighting their progress and limitations.

VII. CONCLUSION

This paper presents a new challenge for LLMs as human assistants to control many units in a strategy game. We proposed a new framework, HIVE, to allow a player to give high-level commands that an LLM translates into a long-term plan that controls the behavior of each unit. We showed that generalist LLMs such as Claude Sonnet 3.5, Gemini 2, and GPT-4o can handle such tasks but are still sensitive to slight changes in the player’s prompts. Complementary experiments showed that HIVE requires human help to get the best performance and that generalist LLMs’ visual capacity to use an out-of-distribution map for terrain and landmark locations still needs improvement. This work opens many interesting avenues for

enhancing LLMs' capacities to collaborate with humans, such as better map-reading abilities and long-term planning, as well as reduced sensitivity to prompt variations.

ACKNOWLEDGMENTS

Funded by the armasuisse S+T project F00-007.

REFERENCES

- [1] A. Sharma, S. Rao, C. Brockett, A. Malhotra, N. Jojic, and B. Dolan, “Investigating Agency of LLMs in Human-AI Collaboration Tasks,” in *EACL 2024*, 2024.
- [2] A. Bakhtin, D. J. Wu, A. Lerer, J. Gray, A. P. Jacob, G. Farina, A. H. Miller, and N. Brown, “Mastering the Game of No-Press Diplomacy via Human-Regularized Reinforcement Learning and Planning,” Oct. 2022.
- [3] E. Dinan, G. Farina, C. Flaherty, D. Fried, A. Goff, J. Gray, H. Hu, A. P. Jacob, M. Komeili, K. Konath, M. Kwon, A. Lerer, M. Lewis, A. H. Miller, S. Mitts, A. Renduchintala, S. Roller, D. Rowe, W. Shi, J. Spisak, A. Wei, D. Wu, H. Zhang, and M. Zijlstra, “Human-level play in the game of *Diplomacy* by combining language models with strategic reasoning,” *Science*, vol. 378, no. 6624, pp. 1067–1074, Dec. 2022.
- [4] N. Lorè and B. Heydari, “Strategic behavior of large language models and the role of game structure versus contextual framing,” *Scientific Reports*, vol. 14, no. 1, p. 18490, Aug. 2024.
- [5] Y. Zhang, S. Mao, T. Ge, X. Wang, A. de Wynter, Y. Xia, W. Wu, T. Song, M. Lan, and F. Wei, “LLM as a Mastermind: A Survey of Strategic Reasoning with Large Language Models,” 2024.
- [6] X. Shao, W. Jiang, F. Zuo, and M. Liu, “SwarmBrain: Embodied agent for real-time strategy game StarCraft II via large language models,” 2024.
- [7] W. J. Zhou, B. Subagdja, A.-H. Tan, and D. W.-S. Ong, “Hierarchical control of multi-agent reinforcement learning team in real-time strategy (RTS) games,” *Expert Systems with Applications*, vol. 186, 2021.
- [8] Y. Cao and C. Lee, “Robot behavior-tree-based task generation with large language models,” in *CEUR workshop proceedings*, vol. 3433. RWTH Aachen University, 2023.
- [9] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson, “The StarCraft Multi-Agent Challenge,” Dec. 2019.
- [10] T. Rashid, M. Samvelyan, C. S. D. Witt, G. Farquhar, J. Foerster, and S. Whiteson, “Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning,” *ArXiv*, Mar. 2020.
- [11] C. Yu, A. Vedula, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, “The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games,” in *Neural Information Processing Systems*, Mar. 2021.
- [12] B. Ellis, J. Cook, S. Moalla, M. Samvelyan, M. Sun, A. Mahajan, J. N. Foerster, and S. Whiteson, “SMACv2: An Improved Benchmark for Cooperative Multi-Agent Reinforcement Learning,” Oct. 2023.
- [13] A. Rutherford, B. Ellis, M. Gallici, J. Cook, A. Lupu, G. Ingvarsson, T. Willi, A. Khan, C. S. de Witt, A. Souly, S. Bandyopadhyay, M. Samvelyan, M. Jiang, R. T. Lange, S. Whiteson, B. Lacerda, N. Hawes, T. Rocktaschel, C. Lu, and J. N. Foerster, “JaxMARL: Multi-Agent RL Environments in JAX,” Dec. 2023.
- [14] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, and X. Zhang, “Large Language Model based Multi-Agents: A Survey of Progress and Challenges,” Apr. 2024.
- [15] B. Pan, J. Lu, K. Wang, L. Zheng, Z. Wen, Y. Feng, M. Zhu, and W. Chen, “AgentCoord: Visually Exploring Coordination Strategy for LLM-based Multi-Agent Collaboration,” Apr. 2024.
- [16] S. Han, Q. Zhang, Y. Yao, W. Jin, Z. Xu, and C. He, “LLM Multi-Agent Systems: Challenges and Open Problems,” 2024.
- [17] J. Kramár, T. Eccles, I. Gemp, A. Tacchetti, K. R. McKee, M. Malinowski, T. Graepel, and Y. Bachrach, “Negotiation and honesty in artificial intelligence methods for the board game of *Diplomacy*,” *Nature Communications*, vol. 13, no. 1, p. 7214, Dec. 2022.
- [18] D. Paglieri, B. Cupial, S. Coward, U. Piterbarg, M. Wolczyk, A. Khan, E. Pignatelli, L. Kuciński, L. Pinto, R. Fergus, J. N. Foerster, J. Parker-Holder, and T. Rocktäschel, “BALROG: Benchmarking Agentic LLM and VLM Reasoning On Games,” Nov. 2024.
- [19] A. Ruoss, F. Pardo, H. Chan, B. Li, V. Mnih, and T. Genewein, “Lmact: A benchmark for in-context imitation learning with long multimodal demonstrations,” *arXiv preprint arXiv:2412.01441*, 2024.
- [20] M. G. Mecattaf, B. Slater, M. Tešić, J. Prunty, K. Voudouris, and L. G. Cheke, “A little less conversation, a little more action, please: Investigating the physical common-sense of LLMs in a 3D embodied environment,” 2024.
- [21] K. Wang, X. Zhang, H. Liu, S. Han, H. Ma, and T. Hu, “CreDes: Causal Reasoning Enhancement and Dual-End Searching for Solving Long-Range Reasoning Problems using LLMs,” 2024.
- [22] M. Xing, R. Zhang, H. Xue, Q. Chen, F. Yang, and Z. Xiao, “Understanding the Weakness of Large Language Model Agents within a Complex Android Environment,” in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Barcelona Spain: ACM, Aug. 2024, pp. 6061–6072.
- [23] Y. Zhou, L. Feng, Y. Ke, X. Jiang, J. Yan, X. Yang, and W. Zhang, “Towards Vision-Language Geo-Foundation Model: A Survey,” 2024.
- [24] P. Rahmanzadehgervi, L. Bolton, M. R. Taesiri, and A. T. Nguyen, “Vision language models are blind,” in *Computer Vision – ACCV 2024*, M. Cho, I. Laptev, D. Tran, A. Yao, and H. Zha, Eds. Singapore: Springer Nature Singapore, 2025, pp. 293–309.
- [25] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, “JAX: composable transformations of Python+NumPy programs,” 2018.
- [26] Inkarnate. (2025) Inkarnate – An online map-making platform. [Online]. Available: <https://inkarnate.com>
- [27] N. Sevcenko, T. Appel, M. Ninaus, K. Moeller, and P. Gerjets, “Theory-based approach for assessing cognitive load during time-critical resource-managing human-computer interactions: An eye-tracking study,” *Journal on Multimodal User Interfaces*, vol. 17, 2023.
- [28] M. Colledanchise and P. Ögren, *Behavior trees in robotics and AI: An introduction*. CRC Press, 2018.
- [29] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Alten Schmidt, S. Altman, S. Anadkat *et al.*, “Gpt-4 technical report,” *arXiv preprint arXiv:2303.08774*, 2023.
- [30] Anthropic, “The claude 3 model family: Opus, sonnet, haiku,” 2024.
- [31] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican *et al.*, “Gemini: a family of highly capable multimodal models,” *arXiv:2312.11805*, 2023.
- [32] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, “The llama 3 herd of models,” *arXiv preprint arXiv:2407.21783*, 2024.
- [33] X. Lei, Z. Yang, X. Chen, P. Li, and Y. Liu, “Scaffolding coordinates to promote vision-language coordination in large multi-modal models,” *arXiv preprint arXiv:2402.12058*, 2024.
- [34] S. Joel, J. J. Wu, and F. H. Fard, “A survey on llm-based code generation for low-resource and domain-specific programming languages,” *arXiv preprint arXiv:2410.03981*, 2024.
- [35] L. . Wang, Y. . S. o. S. E. Shen, and L. Guangzhou Intelligence Communications Technology Co., “Evaluating Causal Reasoning Capabilities of Large Language Models: A Systematic Analysis Across Three Scenarios,” p. 4584.
- [36] S. Oh, “Evaluating Mathematical Problem-Solving Abilities of Generative AI Models: Performance Analysis of o1-preview and gpt-4o Using the Korean College Scholastic Ability Test,” vol. 13, pp. 1227–1235.
- [37] A. P. Widjyassari, S. Rustad, G. F. Shidik, E. Noersasongko, A. Syukur, A. Affandy *et al.*, “Review of automatic text summarization techniques & methods,” *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 4, pp. 1029–1046, 2022.
- [38] S. Ouyang, J. M. Zhang, M. Harman, and M. Wang, “An empirical study of the non-determinism of chatgpt in code generation,” vol. 34, 2025.
- [39] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, “Voyager: An open-ended embodied agent with large language models,” *arXiv preprint arXiv:2305.16291*, 2023.
- [40] H. Liu, X. Zhang, H. Xu, Y. Wanyan, J. Wang, M. Yan, J. Zhang, C. Yuan, C. Xu, W. Hu *et al.*, “Pc-agent: A hierarchical multi-agent collaboration framework for complex task automation on pc,” *arXiv preprint arXiv:2502.14282*, 2025.
- [41] Y. Zhang, C. Wei, Z. He, and W. Yu, “Geogpt: An assistant for understanding and processing geospatial tasks,” *International Journal of Applied Earth Observation and Geoinformation*, 2024.
- [42] X. Wang, B. Zhuang, and Q. Wu, “Are large vision language models good game players?” in *The Thirteenth International Conference on Learning Representations*, 2025.

APPENDIX

A. Inference time analyses

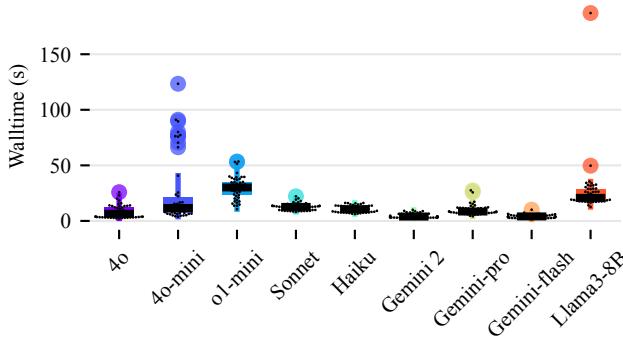


Fig. 10. Wall-times of the 50 LLM queries of the main evaluations. Llama3-8B is run on an M3 chip. HIVE queries the others via their API.

B. Temperature ablation

We performed an ablation study of the effect of the LLM temperature (Fig. 11). Increasing the temperature doesn't significantly affect the plans' quality for the simpler ability test **Follow markers**. For the more complex ability test, **Exploit terrain**, **4o** starts returning no plan 2 out of 10 queries with a temperature above 0.3 and has a slight decrease of victorious plans with temperatures above 0.6, and **Sonnet** starts returning invalid plans 1 out of 10 queries for temperatures above 0.6.

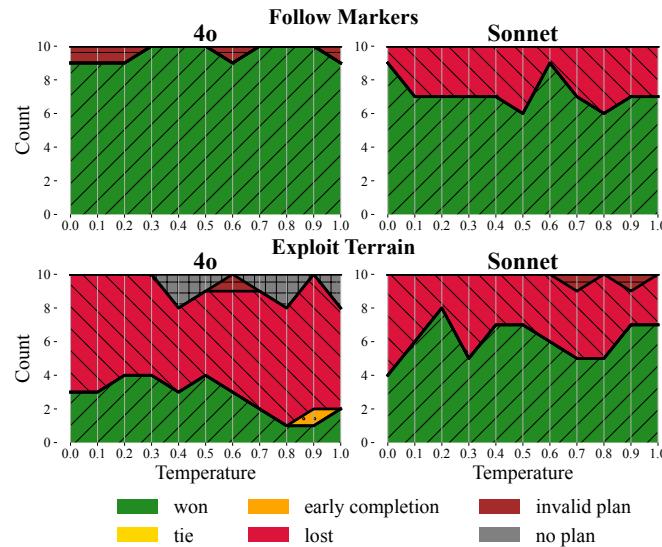


Fig. 11. LLM temperature effect on the plan execution for **4o** and **Sonnet** on the **Follow markers** and **Exploit terrain** ability tests.

C. Map description inputs

The section presents the four inputs used for the "Does HIVE need the textual description of the map?" ablation. Fig. 12 for the **Exploit terrain** ability test and Fig. 13 for the **Strategize points** ability test.

(a) Textual description

River: water at (46, 110) - (102, 200), (85, 57) - (107, 91), (82, 36) - (111, 58), (98, 0) - (134, 34)
 Bridge: normal at (78, 97) - (112, 106)
 Bridge northern wall: building at (77, 105) - (113, 109)
 Bridge southern wall: building at (77, 92) - (113, 96)
 West Forest: trees at (11, 101) with radius 10
 North-West Forest: trees at (33, 159) with radius 30
 South-West Forest: trees at (23, 44) with radius 20
 North-East Forest: trees at (135, 156) with radius 20
 East Forest: trees at (164, 71) with radius 30

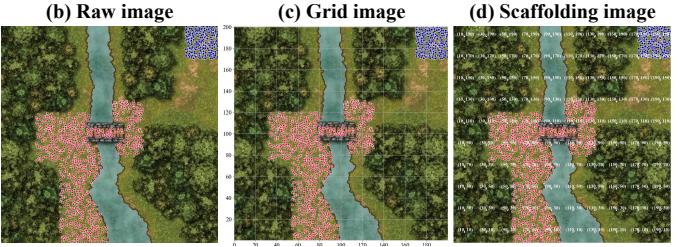


Fig. 12. Inputs describing the map for the **Exploit terrain** ability test.

(a) Textual description

Northern River: water [(0, 300), (34, 253), (73, 238), (118, 217), (153, 185), (192, 147), (217, 131), (245, 2), (260, 70), (268, 40), (300, 0)]
 Southern River: water [(0, 300), (34, 253), (34, 227), (57, 143), (75, 128), (126, 85), (151, 68), (238, 28), (300, 0)]
 Northern Bridge 1: normal at (45, 252)
 Northern Bridge 2: normal at (138, 205)
 Northern Bridge 3: normal at (205, 142)
 Northern Bridge 4: normal at (252, 87)
 Northern Bridge 5: normal at (272, 37)
 Southern Bridge 1: normal at (29, 238)
 Southern Bridge 2: normal at (65, 135)
 Southern Bridge 3: normal at (135, 75)
 Southern Bridge 4: normal at (249, 21)
 Northern Forest: trees at (216, 220) - (300, 300)
 Southern Forest: trees at (0, 0) - (57, 48)

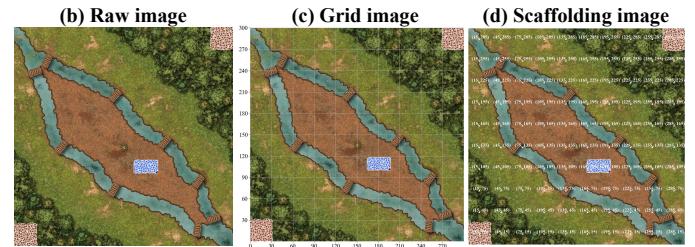


Fig. 13. Inputs describing the map for the **Strategize points** ability test.

D. Units characteristics

E. LLM instruction and game for the example of the teaser figure

1) Instruction:

```
# Map Instruction
```

You are a game assistant that helps the ↳ player in a strategy video game.

TABLE II
CHARACTERISTICS OF THE THREE TYPES OF UNITS USED IN THIS PAPER.

	Spearman	Archer	Cavalry
Visual shape	square	circle	triangle
Speed (unit distance/step)	1	2	6
Max health	24	2	12
Attack damage	1	3	1
Attack range	1	15	1

Your task is to discuss with the player to come up with a plan to win the game's scenario (which will be provided later). Work with the player by giving feedback about their propositions, asking questions to clarify, obtain more details, or decide between different propositions. The player can also ask you questions.

You will be given a textual description of each pertinent element of the map using their name, terrain type, and bounding boxes (bottom-left corner - top-right corner).

In the game, there are four types of terrain:

- Normal: units can cross and see through (by default, the whole map is normal).
- Buildings: units cannot cross or see through buildings.
- Water: units cannot cross over water but can see over it.
- Trees: units cannot see through trees but can move over them. In particular, once a unit is inside a tree area, it cannot see any other unit.

Also, for simplicity, bridges that allow crossing water terrain will be specified. They correspond to normal terrain.

You can assume that any part of the map that is not included in any of the pertinent elements has a normal type of terrain.

A common convention is that East = Right, North = Top, West = Left, and South = Bottom. So the point (0, 0) is the bottom-left corner of the map.

The x-axis increases from West to East = from Left to Right, and the y-axis increases from South to North = from Bottom to Top.

Format:

[Name]: [terrain type] at [circle or square coordinate], [circle or square coordinate], ..., [circle or square coordinate]
 where [circle] = [(x, y) coordinates of center] with radius [R]
 where [square coordinate] = [(x, y) coordinates of the bottom-left corner] - [(x, y) coordinates of the top-right corner]

For example
 East Forest: trees at (12, 63) with radius 20
 North River: water at (0, 85) - (100, 90)
 North River's Bridge: normal at (45, 85) - (55, 90)
 West Forest: trees at (0, 23) with radius 6, (5, 33) with radius 7

Markers

Through the discussion, the player can define markers on the map, which will be provided to you using the following format:

Markers:

- A at (17, 5)
- B at (6, 32)
- C at (25, 28)

Information About the Units

Description of the unit types in the game:
 spearmen: Health=24; Sight range=15;

- Attack range=1; Moving speed=2; Attack damage=1; Attack cooldown=1

archer: Health=2; Sight range=15; Attack

- range=15; Moving speed=4; Attack damage=3; Attack cooldown=1

cavalry: Health=12; Sight range=15; Attack

- range=1; Moving speed=12; Attack damage=1; Attack cooldown=1

Spearmen are strong against Cavalry

- because they have more health.

Cavalry are strong against Archers

- because they can quickly engage in close combat where Archers are weak.

Archers are strong against Spearmen

- because they can attack from a longer distance.

Here is the list of unit IDs (in the form
 ↳ of a Python slice a:b with a included
 ↳ and b not included) for each team and
 ↳ the type of units they are composed
 ↳ of:

Descriptions of each team's composition:

Allies:

```
spearmen: [0:350]
archer: [350:700]
```

Enemies:

```
spearmen: [0:900]
```

You will be provided with a list of all
 ↳ the units' current health and
 ↳ position.

Both teams are composed of many units. You
 ↳ should start by analyzing the
 ↳ positions of the units still alive to
 ↳ compose groups and compute their
 ↳ average position so that you can form
 ↳ the plan and send units to the
 ↳ appropriate positions.

How You Should Handle the Player's
 ↳ Prompt

The user will ask you to write one plan.

↳ If you already have a shared
 ↳ conversation and the user asks you to
 ↳ modify or add steps to the plan, take
 ↳ care to build upon the already
 ↳ selected plan.

Syntax for a Detailed Plan

A detailed plan is a set of steps to
 ↳ achieve in a given order until all the
 ↳ steps are completed.

You must provide the steps of the plan
 ↳ between the two keywords "BEGIN PLAN"
 ↳ and "END PLAN".

As you only propose one plan, you should
 ↳ use these two keywords once.

One step comprises:

- A numeral ID
- A list of prerequisite steps that need
 ↳ to be completed before the step is
 ↳ rolled out
- An objective

A succession (at least one) of groups of
 ↳ units:

- The unit IDs of the group

- Target position on the map for the units
 ↳ to go to if there are no enemies in
 ↳ sight
- A behavior for the units if there are
 ↳ enemies in sight

Syntax for a Detailed Plan (continued)

The list of prerequisites corresponds to
 ↳ the list of steps' IDs that must be
 ↳ completed before trying to achieve the
 ↳ objective.

There are two kinds of objectives:

- **Elimination objective:** where the
 ↳ objective is completed when all the
 ↳ targets are eliminated. In that case,
 ↳ you must provide a list of enemies'
 ↳ IDs or the keyword "all" if all
 ↳ enemies are targets.
- **Position objective:** where the
 ↳ objective is completed when all the
 ↳ concerned units are close to their
 ↳ target position.

Position objectives are a good way to move
 ↳ your units, but if the end objective
 ↳ is to eliminate enemies, it can be
 ↳ more straightforward to directly set
 ↳ an elimination objective and use the
 ↳ target position to move the units.

The concerned units are either the keyword
 ↳ "all" (if all the allies' units are
 ↳ concerned) or a list of integers
 ↳ corresponding to the unit IDs.

The behavior corresponds to a low-level
 ↳ and local behavior that the units will
 ↳ follow.

Here is the list of available behaviors:

- **attack_in_close_range:** the unit
 ↳ attacks the enemies in close range if
 ↳ there are enemies or moves toward the
 ↳ target position if there are no
 ↳ enemies.
- **attack_and_move:** the unit attacks
 ↳ the enemies without moving if there
 ↳ are enemies or moves toward the target
 ↳ position if there are no enemies.
- **attack_in_long_range:** the unit
 ↳ attacks the enemies in long range if
 ↳ there are enemies or moves toward the
 ↳ target position if there are no
 ↳ enemies.
- **follow_map:** ignore the enemies and
 ↳ simply move straight to the target
 ↳ objective. Only use it when the player
 ↳ asks you to ignore the enemies.

Apart from standing still, all these behaviors will only be active if an enemy is in sight. Otherwise, they will move to a target position if you set one in the plan or stand still if no target position is set.

Remember that units collide and push each other, so ignoring the enemies may not be the fastest way to reach a target position if there are enemies on the way who could block you.

The syntax format of a step is the following:

Step ID: (where you replace ID with the integer ID of the step)

prerequisites: [s_1, s_2, ..., s_n] (where the s_i correspond to the prerequisites' steps IDs. Note that the list can be empty. In that case, simply write [])

objective: position [or] elimination
→ UNIT_LIST

(At least one list of units and their assigned behavior and target position, but there can be as many groups as there are units, as one unit can belong to two groups:)

UNIT_LIST:

- target position: (x, y) (The integer x and y coordinate on the map.)
- behavior: behavior_name target_1
→ target_2 ... target_n (where behavior_name is an available behavior and target_1 to target_n are the targeted unit types or just the keyword "any" if the behavior targets any unit_types)

A UNIT_LIST is the list of unit IDs, and it has the following format:

- Either it's the word "all" (without quotes), which means that all the units of the corresponding team are concerned.
- Or it's a list of IDs in the format "[X1, X2, ..., Xn]" where Xi can either be an integer or a slice "a:b" (with a and b as integers, b>a, a included and b not included like in Python's range function). If a is not specified (i.e., ":b") it is considered to be 0, and if b is not specified (i.e., "a:") it is considered to be the total number of agents in the considered team.

IMPORTANT:

- All positions (x, y) must be integers.
→ If you want to give float positions, convert them first into integers.
- Do not add comments to the plan specification, as it interferes with the parser. If you want to give comments, give them outside the "BEGIN PLAN" and "END PLAN".
- A unit can only belong to one group of units for the same step.

Example of a Valid Detailed Plan:

BEGIN PLAN

Step 0:

prerequisites: []

objective: position

units: all

- target position: (24, 14)
- behavior: attack_and_move any

Step 1:

prerequisites: [0]

objective: position

units: [0,1,2,10:]

- target position: (24, 16)
- behavior: attack_and_move any
- units: [10:15,30:]
- target position: (24, 14)
- behavior: attack_and_move any

Step 2:

prerequisites: [1]

objective: elimination [:15]

units: [:30]

- target position: (24, 24)
- behavior: attack_in_close_range archer
→ spearmen
- units: [30:60]
- target position: (24, 24)
- behavior: attack_in_long_range spearmen

List of Planning Mistakes You Should Avoid:

Avoid creating a series of position objectives with different groups of

- units waiting for each other (through a chain of prerequisites). Instead, regroup those positions into one step so that all units can move simultaneously.

- ↳ Ensure all units have assigned behaviors
 - ↳ at every step. If two or more steps can be active simultaneously (because they share the same prerequisites), each unit must belong to at least one group.

For elimination objectives, ensure the target position is close enough to the targeted enemy units for effective engagement.

Elimination objectives already have a default target position, so avoid unnecessary position objectives before them. Instead, set the target position of the elimination objective to the same location.

Ensure the units' IDs are enclosed in
→ square brackets [and].

Use the correct type name for long-range units as "archer" (not "archers").

Verify the name of the unit behaviors
→ (e.g., "attack_in_close_range") and
→ ensure all units' target positions are
→ integers.

Be sure to adapt this format and guidance

- ↳ as needed when responding to the
- ↳ player's requests.

In this scenario, you control the Allies → and must prevent the Enemies from → invading the center of your camp in → (150, 134).

- You must elaborate a defensive plan
 - that eliminates all the Enemies
 - while preventing any of them from
 - reaching the center of your camp.

If one Enemy unit reaches the center
→ of your camp, all your units are
→ eliminated, or if your plan is
→ completed while there are still
→ standing Enemy units, you lose the
→ game.

Your intel reports inform you that the
→ enemy will converge toward the
→ center of your camp and attack
→ your units in sight at close
→ range.

2) The game info:

Health and positions of all the units of
→ each team (\emptyset means that the unit is
→ dead).

Allies:

Y positions: [107, 104, 105, 105, 113,
 ↵ 103, 109, 113, 105, 114, 110, 114,
 ↵ 112, 110, 110, 112, 110, 103, 107,
 ↵ 103, 103, 111, 106, 101, 110, 107,
 ↵ 105, 112, 112, 115, 106, 106, 115,
 ↵ 107, 102, 103, 111, 110, 107, 105,
 ↵ 109, 107, 109, 114, 101, 101, 112,
 ↵ 115, 108, 111, 102, 110, 106, 110,
 ↵ 105, 107, 103, 103, 112, 112, 110,
 ↵ 116, 103, 101, 114, 111, 103, 110,
 ↵ 106, 103, 108, 101, 116, 116, 101,
 ↵ 109, 102, 101, 104, 103, 110, 115,
 ↵ 103, 116, 111, 116, 103, 115, 102,
 ↵ 115, 113, 102, 114, 108, 104, 105,
 ↵ 114, 103, 106, 111, 107, 113, 113,
 ↵ 101, 107, 108, 104, 116, 104, 110,
 ↵ 105, 116, 103, 114, 102, 101, 115,
 ↵ 106, 115, 113, 112, 105, 109, 115,
 ↵ 113, 111, 105, 104, 114, 103, 105,
 ↵ 115, 103, 108, 107, 114, 102, 116,
 ↵ 106, 115, 106, 112, 110, 106, 106,
 ↵ 102, 111, 115, 111, 106, 110, 112,
 ↵ 105, 105, 105, 114, 113, 101, 114,
 ↵ 109, 110, 101, 110, 113, 110, 107,
 ↵ 116, 115, 112, 101, 104, 114, 107,
 ↵ 115, 111, 108, 109, 106, 101, 115,
 ↵ 104, 106, 108, 114, 101, 101, 105,
 ↵ 104, 106, 114, 105, 112, 108, 116,
 ↵ 111, 105, 104, 115, 106, 106, 103,
 ↵ 101, 109, 108, 109, 106, 101, 102,
 ↵ 109, 102, 104, 114, 103, 110, 107,
 ↵ 110, 103, 116, 115, 114, 107, 108,
 ↵ 109, 109, 109, 115, 107, 115, 104,
 ↵ 111, 110, 112, 101, 110, 116, 103,
 ↵ 110, 103, 102, 110, 110, 113, 105,
 ↵ 112, 113, 110, 115, 107, 108, 105,
 ↵ 113, 110, 101, 112, 104, 116, 103,
 ↵ 107, 112, 106, 115, 111, 103, 113,
 ↵ 110, 104, 108, 106, 114, 113, 105,
 ↵ 101, 116, 112, 106, 102, 101, 109,
 ↵ 101, 116, 102, 102, 108, 103, 105,
 ↵ 104, 116, 103, 112, 111, 110, 109,
 ↵ 116, 103, 112, 110, 110, 116, 114,
 ↵ 112, 111, 106, 101, 110, 116, 107,
 ↵ 111, 110, 113, 102, 109, 112, 101,
 ↵ 114, 103, 116, 105, 116, 104, 113,
 ↵ 105, 101, 105, 106, 112, 116, 103,
 ↵ 116, 101, 103, 101, 106, 104, 105,
 ↵ 110, 105, 105, 112, 112, 108, 114,
 ↵ 106, 112, 114, 116, 116, 111, 114,
 ↵ 111, 109, 103, 108, 114, 112, 110,
 ↵ 111, 108, 103, 103, 106, 107, 102,
 ↵ 112, 104, 110, 116, 105, 112, 101,
 ↵ 104, 101, 108, 112, 106, 106, 102,
 ↵ 101, 115, 112, 106, 114, 106, 116,
 ↵ 103, 109, 103, 110, 105, 109, 113,
 ↵ 104, 113, 115, 116, 111, 102, 111,
 ↵ 101, 106, 107, 104, 105, 108, 101,

108, 108, 116, 101, 107, 104, 106, 114,
 ↵ 110, 107, 115, 105, 107, 110, 103,
 ↵ 106, 109, 115, 112, 112, 109, 113,
 ↵ 106, 105, 102, 112, 110, 105, 104,
 ↵ 102, 109, 103, 108, 107, 115, 113,
 ↵ 111, 115, 113, 104, 110, 113, 115,
 ↵ 107, 103, 109, 112, 110, 105, 113,
 ↵ 102, 116, 115, 114, 102, 116, 112,
 ↵ 108, 109, 110, 111, 111, 102, 116,
 ↵ 102, 110, 113

F. LLM answer and plan to Fig. 4 examples

1) Coordination with Sonnet:

I'll help you create a plan to effectively
 → eliminate the enemies. Looking at the
 → positions, I notice that all enemy
 → units are currently in the Northern
 → Forest (y-coordinates between
 → 135-149), and they will need to move
 → south through the middle of the map.

Let's analyze our forces:

- We have 500 spearmen (IDs 0-499)
- We have 500 archers (IDs 500-999)

Given the map's width (150) and our need

→ to protect archers, I suggest creating
 → three main battle groups across the
 → middle of the map, each with both
 → spearmen and archers. The spearmen
 → will form a protective front line
 → while archers attack from behind.

Here's the detailed plan:

```
BEGIN PLAN
Step 0:
prerequisites: []
objective: position
units: [0:167]
- target position: (25, 75)
- behavior: attack_in_close_range any
units: [167:334]
- target position: (75, 75)
- behavior: attack_in_close_range any
units: [334:500]
- target position: (125, 75)
- behavior: attack_in_close_range any
units: [500:667]
- target position: (25, 65)
- behavior: attack_in_long_range any
units: [667:834]
- target position: (75, 65)
- behavior: attack_in_long_range any
units: [834:1000]
- target position: (125, 65)
- behavior: attack_in_long_range any
```

Step 1:

```

prerequisites: [0]
objective: elimination all
units: [0:167]
- target position: (25, 75)
- behavior: attack_in_close_range any
units: [167:334]
- target position: (75, 75)
- behavior: attack_in_close_range any
units: [334:500]
- target position: (125, 75)
- behavior: attack_in_close_range any
units: [500:667]
- target position: (25, 65)
- behavior: attack_in_long_range any
units: [667:834]
- target position: (75, 65)
- behavior: attack_in_long_range any
units: [834:1000]
- target position: (125, 65)
- behavior: attack_in_long_range any
END PLAN

```

This plan creates three battle groups
 → positioned across the map:

1. Western group (at x=25)
2. Central group (at x=75)
3. Eastern group (at x=125)

Each group has:

- About 167 spearmen in front (at y=75)
- About 167 archers behind them (at y=65)

The spearmen are positioned 10 units ahead
 → of the archers to provide protection
 → while still allowing the archers to
 → shoot over them at approaching
 → enemies.

The groups are spread out to cover the
 → entire width of the map, ensuring no
 → enemies can slip through. When enemies
 → come down from the Northern Forest,
 → they'll face a solid line of spearmen
 → backed by archer fire.

Would you like me to adjust any aspect of
 → this plan?

2) Exploit weaknesses with Sonnet:

Let me analyze the situation and the enemy
 → positions.

The map is divided into 4 quadrants by two
 → rivers that cross at (46-52, 47-53),
 → with bridges allowing passage between
 → these quadrants.

Enemy forces are organized in three
 → distinct groups:

1. Northwest quadrant (around x:5-25,
 → y:75-95): Mainly archers (health=2)
2. Northeast quadrant (around x:75-95,
 → y:5-25): Mainly spearmen (health=24)
3. Northeast quadrant (around x:75-95,
 → y:75-95): Mainly cavalry (health=12)

Our forces are currently positioned in the
 → Southwest quadrant (around x:5-40,
 → y:5-40).

Strategic considerations:

1. We should use cavalry against archers
 → (NW group)
2. Archers against spearmen (NE group)
3. Spearmen against cavalry (NE group)

Here's my proposed plan:

BEGIN PLAN

Step 0:

```

prerequisites: []
objective: position
units: [0:250]
- target position: (19, 49)
- behavior: attack_and_move any
units: [250:500]
- target position: (21, 49)
- behavior: attack_and_move any
units: [500:750]
- target position: (20, 49)
- behavior: attack_and_move any

```

Step 1:

```

prerequisites: [0]
objective: position
units: [500:750]
- target position: (21, 76)
- behavior: attack_in_close_range archer
units: [250:500]
- target position: (76, 21)
- behavior: attack_in_long_range spearmen
units: [0:250]
- target position: (76, 76)
- behavior: attack_in_close_range cavalry

```

Step 2:

```

prerequisites: [1]
objective: elimination all
units: [500:750]
- target position: (15, 85)
- behavior: attack_in_close_range archer
units: [250:500]
- target position: (85, 15)
- behavior: attack_in_long_range spearmen
units: [0:250]

```

```

- target position: (85, 85)
- behavior: attack_in_close_range cavalry
END PLAN

```

Plan explanation:

1. First, we move all forces to the west
 - bridge (Step 0)
2. Then split our forces (Step 1):
 - Cavalry (500:750) goes north to engage
 - archers
 - Archers (250:500) goes east to engage
 - spearmen
 - Spearmen (0:250) goes northeast to
 - engage cavalry
3. Finally, each group moves to eliminate
 - their targeted enemies (Step 2)

This plan leverages each unit type's

- strengths against their natural
- enemies while using the bridges
- strategically to reach their
- positions.

3) Markers with 4o-mini:

To achieve your goal of moving the troops

- to the markers A, B, C, and D while
- ignoring enemies, we can create a plan
- that sequentially moves the units to
- each marker and then to the objective
- position at (61, 0).

However, since the objective is to reach

- the position at (61, 0) with at least
- one unit, we should ensure that the
- units are grouped effectively and can
- reach the objective after visiting the
- markers.

Let's break down the plan:

1. Move all units to marker A at (193, 85).
2. Move all units to marker B at (49, 136).
3. Move all units to marker C at (9, 134).
4. Move all units to marker D at (11, 9).
5. Finally, move all units to the
 - objective position at (61, 0).

I will now create the detailed plan based

- on this sequence.

BEGIN PLAN

```

Step 0:
prerequisites: []
objective: position
units: all
- target position: (193, 85)
- behavior: follow_map

```

```

Step 1:
prerequisites: [0]
objective: position
units: all
- target position: (49, 136)
- behavior: follow_map

```

```

Step 2:
prerequisites: [1]
objective: position
units: all
- target position: (9, 134)
- behavior: follow_map

```

```

Step 3:
prerequisites: [2]
objective: position
units: all
- target position: (11, 9)
- behavior: follow_map

```

```

Step 4:
prerequisites: [3]
objective: position
units: all
- target position: (61, 0)
- behavior: follow_map
END PLAN

```

This plan allows your units to move to the

- specified markers and then to the
- objective position while ignoring
- enemies. Would you like to make any
- adjustments or proceed with this plan?

4) Terrain with 4o:

Here is the detailed plan based on your

- instructions:

```

BEGIN PLAN
Step 0:
prerequisites: []
objective: position
units: all
- target position: (164, 71)
- behavior: follow_map any

```

```

Step 1:
prerequisites: [0]
objective: position
units: all
- target position: (33, 159)
- behavior: follow_map any

```

```

Step 2:
prerequisites: [1]
objective: position
units: all

```

```
- target position: (11, 101)
- behavior: follow_map any
```

Step 3:

```
prerequisites: [2]
objective: position
units: all
- target position: (23, 44)
- behavior: follow_map any
```

Step 4:

```
prerequisites: [3]
objective: position
units: all
- target position: (61, 0)
- behavior: follow_map any
END PLAN
```

5) Strategic points with o1-mini:

BEGIN PLAN

Step 0:

```
prerequisites: []
objective: position
units: [0:39]
- target position: (45, 252)
- behavior: attack_in_close_range any
units: [350:389]
- target position: (45, 252)
- behavior: attack_in_long_range any
```

```
units: [39:78]
- target position: (138, 205)
- behavior: attack_in_close_range any
units: [389:428]
- target position: (138, 205)
- behavior: attack_in_long_range any
```

```
units: [78:117]
- target position: (205, 142)
- behavior: attack_in_close_range any
units: [428:467]
- target position: (205, 142)
- behavior: attack_in_long_range any
```

```
units: [117:156]
- target position: (252, 87)
- behavior: attack_in_close_range any
units: [467:506]
- target position: (252, 87)
- behavior: attack_in_long_range any
```

```
units: [156:195]
- target position: (272, 37)
- behavior: attack_in_close_range any
units: [506:545]
- target position: (272, 37)
- behavior: attack_in_long_range any
```

```
units: [195:234]
- target position: (29, 238)
- behavior: attack_in_close_range any
units: [545:584]
- target position: (29, 238)
- behavior: attack_in_long_range any

units: [234:273]
- target position: (65, 135)
- behavior: attack_in_close_range any
units: [584:623]
- target position: (65, 135)
- behavior: attack_in_long_range any

units: [273:312]
- target position: (135, 75)
- behavior: attack_in_close_range any
units: [623:662]
- target position: (135, 75)
- behavior: attack_in_long_range any

units: [312:350]
- target position: (249, 21)
- behavior: attack_in_close_range any
units: [662:700]
- target position: (249, 21)
- behavior: attack_in_long_range any
END PLAN
```

G. Grammar for our Behavior trees

?start: node

```
%import common.WS
%ignore WS
```

```
nodes : node ("::" node | "|>" node)*
node  :
| sequence
| fallback
| action
| condition
```

```
sequence  : "S" "(" nodes ")"
fallback  : "F" "(" nodes ")"
action    : "A" "(" atomic ")"
condition : "C" "(" atomic ")"
```

```
atomic  :
| move
| attack
| stand
| follow_map
| in_sight
| in_reach
| is_dying
| is_armed
```

```

| is_flock
| is_type
| is_in_forest
| success_action
| failure_action

move      : "move" (direction | sense
→  qualifier (foe | friend) (unit ("or"
→  unit)* |any)?
attack    : "attack" qualifier (unit ("or"
→  unit)* |any)?
stand     : "stand"
in_sight  : "in_sight" (foe | friend)
→  (unit ("or" unit)* |any)?
in_reach  : "in_reach" (foe | friend)
→  source time (unit ("or" unit)* |any)?
is_dying  : "is_dying" (self | foe |
→  friend) intensity
is_armed   : "is_armed" (self | foe |
→  friend)
is_flock   : "is_flock" (foe | friend)
→  direction
is_type    : "is_type" negation unit
follow_map : "follow_map" sense intensity?
is_in_forest : "is_in_forest"
success_action : "success_action"
failure_action: "failure_action"

sense      : /toward|away_from/
direction  : /north | east | south | west | 
→  center/
foe        : /foe/
friend     : /friend/
qualifier  : /strongest | weakest | closest
→  | farthest | random/
intensity  : /low | middle | high/
self       : /self/
unit       : /spearmen | archer| cavalry |
→  balista | dragon | civilian/
any        : /any/
negation   : /a | not_a/
source     : /them_from_me | me_from_them/
time       : /now | low | middle | high/

```

H. List of behavior trees used in this paper

1) Available for HIVE: Long-range attack:

```
F(S(C( in_reach foe me_from_them high any)
→  :: A (move away_from closest foe any)
→  :: A (attack random any) :: A
→  (follow_map toward))
```

Close-range attack:

```
F( A (attack random any) :: A (move toward
→  closest foe any) :: A (follow_map
→  toward))
```

Attack and move:

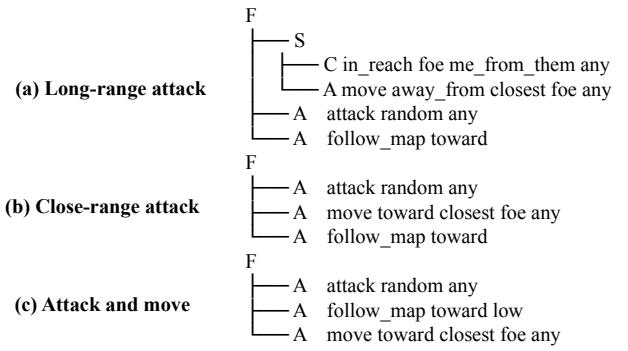


Fig. 14. The three main behavior trees available to HIVE to control each unit.

```
F( A (attack random any) :: A (follow_map
→  toward low) :: A (move toward closest
→  foe any) )
```

Move toward target position:

```
A (follow_map toward)
```

Stand:

```
A (stand)
```

2) Only present in the opposing units: The two following behavior trees are used by the opposing side in the Markers and Terrain ability test to avoid the enemy unit to get lost in the trees where they cannot see the player's units any longer.

Attack in long range and stay out of the forest:

```
F( S (C (is_in_forest) :: A (follow_map
→  toward)) :: S(C( in_reach foe
→  me_from_them high any) :: A (move
→  away_from closest foe any)) :: A
→  (attack closest any))
```

Attack in close range and stay out of the forest:

```
F( S (C (is_in_forest) :: A (follow_map
→  toward)) :: A (attack closest any) :: A
→  (move toward closest foe any))
```

I. Prompts used for the 5 ability tests

1) (a) Coordinate:

0. Make a plan that forms as many squads as you think necessary to cover the central row of the battlefield to eliminate all the enemies as fast as possible. Judiciously place our long range units so that they are not in close combat.
1. Design a plan that forms as many groups of units as you think necessary to cover the central row of the battlefield to eliminate all the enemies as quickly as possible. Judiciously place our long range units so that they are protected by our close range units.

2. Write down a plan that forms as many group of units as you think necessary to cover the central row of the map to eliminate all the enemies as fast as possible. Ensure to place our long range units so that they are not in close combat.
3. Design a plan that forms as many group of units as you think necessary to cover the central row of the map to eliminate all the enemies as quickly as possible. Make sure to place our long range units so that they are in far range of the enemies.
4. Make a plan that forms as many squads as you think necessary to cover the middle row of the map to eliminate all the enemies as quickly as possible. Ensure to place our archers so that they are protected by our close range units.
5. Design a plan that forms as many battalions as you think necessary to cover the middle row of the battlefield to eliminate all the enemies as fast as possible. Ensure to place our archers so that they are in far range of the enemies.
6. Form as many battalions as you think necessary to cover the central row of the battlefield to eliminate all the enemies as fast as possible. Make sure to place our archers so that they are not in close combat.
7. Write down a plan that forms as many battalions as you think necessary to cover the central row of the battlefield to eliminate all the enemies as fast as possible. Make sure to place our long range units so that they are protected by our close range units.
8. Make a plan that forms as many squads as you think necessary to cover the middle row of the battlefield to eliminate all the enemies as quickly as possible. Judiciously place our archers so that they are in far range of the enemies.
9. Make a plan that forms as many group of units as you think necessary to cover the middle row of the battlefield to eliminate all the enemies as quickly as possible. Make sure to place our archers so that they are in far range of the enemies.
0. First, compute the positions and types of the enemy battalions. Then, design a plan that plays on their weaknesses to split our units into three groups with specific strengths and send each one against one of the enemy squads to minimize our casualties.
1. First, determine the positions and types of the enemy battalions. Then, make a plan that plays on their weaknesses to split our army into three battalions with specific strengths and send each one against one of the enemy groups to maximize efficiency.
2. First, analyse the situation to determine the positions and types of the enemy squads. Then, write a plan that plays on their weaknesses to split our army into three battalions with specific strengths and send each one against one of the enemy battalions to minimize our casualties.
3. First, analyse the situation to determine the positions and types of the enemy groups. Then, make a plan that plays on their weaknesses to split our units into three squads with specific strengths and send each one against one of the enemy battalions to minimize our casualties.
4. First, analyse the situation to compute the positions and types of the enemy squads. Then, design a plan that plays on their weaknesses to split our army into three groups with specific strengths and send each one against one of the enemy battalions to minimize our casualties.
5. First, analyse the situation to compute the positions and types of the enemy groups. Then, design a plan that plays on their weaknesses to split our army into three squads with specific strengths and send each one against one of the enemy groups to maximize efficiency.
6. First, analyse the situation to determine the positions and types of the enemy groups. Then, make a plan that plays on their weaknesses to split our army into three squads with specific strengths and send each one against one of the enemy groups to minimize our casualties.

2) (b) Exploit weakness:

7. First, analyse the situation to
 - compute the positions and types of the enemy squads. Then, write a plan that play on their weaknesses to split our army into three groups with specific strengths and send each one against one of the enemy squads to maximize efficiency.
 8. First, compute the positions and types of the enemy squads. Then, make a plan that play on their weaknesses to split our army into three squads with specific strengths and send each one against one of the enemy battalions to maximize efficiency.
 9. First, compute the positions and types of the enemy battalions. Then, design a plan that play on their weaknesses to split our army into three squads with specific strengths and send each one against one of the enemy groups to minimize our casualties.
- 3) (c) Follow markers:** The player mark four positions:
- A: (193, 85);
 - B: (49, 136);
 - C: (9, 134);
 - D: (11, 9).
0. Move the spearmen to each position in alphabetical order while ignoring enemies and then to the final objective position.
 1. Write a plan to move our troops to each marker while ignoring enemies and then to the final objective position.
 2. Design a plan to move our spearmen to each marker in alphabetical order while ignoring enemies and then to the final objective position.
 3. Make a plan to move our spearmen to each position in alphabetical order while ignoring enemies and then to the final objective position.
 4. Write a plan to move our army to each position while ignoring enemies and then to the final objective position.
 5. Write a plan to move my troops to the positions A, B, C, D while ignoring enemies and then to the objective position.
 6. Move our troops to the markers A, B, C, D while ignoring enemies and then to the objective position.
 7. Make a plan to move our army to the markers A, B, C, D while ignoring enemies and then to the objective position.

8. Move our spearmen to the positions ABCD while ignoring enemies and then to the objective position.
 9. Make a plan to move the units to the markers A, B, C, D while ignoring enemies and then to the final objective position.
- 4) (d) Exploit terrain:**
0. Make a plan to move my spearmen inside the forest on the right to the bridge, then make them rush to the forest on the west, slightly on top of the bridge. Then make them hide deep inside the forest on the west of the map. Finally, make them rush south to the left-bottom corner of the map inside the forest and then to the objective, minimizing the time out of cover. Everything while ignoring enemies to not loose time fighting.
 1. Write a plan to move my units inside the trees on the east to the bridge, then make them rush to the forest on the west, slightly on top of the bridge. Then make them hide inside the trees on the west of the map. Finally, make them rush down to the south-east corner of the map inside the forest and then to the objective. Everything while ignoring enemies.
 2. Write a plan to move my spearmen inside the trees on the right to the bridge, then make them rush to the forest on the west, a bit north of the bridge as it is closer. Then make them hide deep inside the forest on the west most part of the map. Finally, make them rush to the south-east corner of the map inside the forest and then to the objective, minimizing the time out of cover. Everything while ignoring enemies to not loose time fighting.
 3. Make a plan to move our units inside the forest on the east not too far to the bridge, then make them rush to the trees on the west, on top of the bridge to quickly regain cover. Then make them hide inside the trees on the west of the map. Finally, make them rush south to the left-bottom corner of the map inside the forest and then to the objective. Everything while ignoring enemies.

4. Design a plan to move our spearmen inside the trees on the east close to the bridge, then make them run to the forest on the left, on top of the bridge to quickly regain cover. Then make them hide deep inside the forest on the left of the map. Finally, make them rush south to the south-east corner of the map inside the forest and then to the objective. Everything while ignoring enemies.
5. Write a plan to move our army inside the forest on the right to the bridge, then make them move to the forest on the west, on top of the bridge as it is closer. Then make them hide deep inside the forest on the west most part of the map. Finally, make them rush to the south-east corner of the map inside the forest and then to the objective, minimizing the time out of cover. Everything while ignoring enemies to not loose time fighting.
6. Write a plan to move my army inside the forest on the east not too far to the bridge, then make them rush to the forest on the west, north of the bridge to quickly regain cover. Then make them hide deep inside the forest on the west most part of the map. Finally, make them rush down to the left-bottom corner of the map inside the forest and then to the objective. Everything while ignoring enemies to not loose time fighting.
7. Move the spearmen inside the trees on the east close to the bridge, then make them rush to the forest on the west, a bit on top of the bridge to quickly regain cover. Then make them hide inside the trees on the left of the map. Finally, make them rush to the south-east corner of the map inside the forest and then to the objective. Everything while ignoring enemies to not loose time fighting.
8. Design a plan to move our spearmen inside the forest on the east close to the bridge, then make them move to the forest on the west, slightly on top of the bridge. Then make them hide deep inside the trees on the west most part of the map. Finally, make them rush to the left-bottom corner of the map inside the forest and then to the objective. Everything while ignoring enemies.
9. Make a plan to move the army inside the forest on the right not too far to the bridge, then make them run to the trees on the west, north of the bridge. Then make them hide inside the trees on the left most part of the map. Finally, make them rush to the south-east corner of the map inside the forest and then to the objective, minimizing the time out of cover. Everything while ignoring enemies.
- 5) (e) Strategize points:**
0. Split our units on all the bridges to defend them and make sure to send archers and spearmen units on each one.
1. Make a plan to split our army on all the bridges to guard them and make sure to send long-range and spearmen units on each one.
2. Design a plan to split our units on all the bridges to protect them and make sure to send long-range and spearmen units on each bridge.
3. Make a plan to split our units on all the bridges to guard them and make sure to send long-range and close-range units on each one.
4. Write a plan to split our army on all the bridges to protect them and make sure to send archers and close-range units on each bridge.
5. Make a plan to split our units on all the bridges to protect them and make sure to send archers and spearmen units on each one.
6. Split our army on all the bridges to guard them and make sure to send long-range and close-range units on each one.
7. Split our army on all the bridges to protect them and make sure to send archers and spearmen units on each one.
8. Design a plan to split our units on all the bridges to defend them and make sure to send archers and close-range units on each bridge.
9. Make a plan to split our army on all the bridges to guard them and make sure to send archers and close-range units on each bridge.

J. Prompts used for HIVE alone

'First, study the situation to find a good
→ approach to achieve this mission then
→ design the corresponding plan.',
'First, analyze the situation to find a
→ good strategy to win this mission
→ then write down the corresponding
→ plan.',
'First, study the situation to find a
→ good approach to achieve this
→ scenario then make the corresponding
→ plan.',
'First, analyze the situation to find a
→ good strategy to win this mission
→ then design the corresponding plan.',
'First, study the situation to find a
→ good approach to achieve this mission
→ then write down the corresponding
→ plan.',
'First, analyze the situation to find a
→ good strategy to achieve this mission
→ then make the corresponding plan.',
'First, analyze the situation to find a
→ good strategy to achieve this
→ scenario then write down the
→ corresponding plan.',
'First, analyze the situation to find a
→ good approach to win this mission
→ then make the corresponding plan.',
'First, study the situation to find a
→ good approach to win this scenario
→ then make the corresponding plan.',
'First, analyze the situation to find a
→ good approach to achieve this
→ scenario then design the
→ corresponding plan.'