Adaptive Command : Real-Time Policy Adjustment via Language Models in StarCraft II

Weiyu Ma
Institute of Automation Chinese Academy of Sciences
Beijing, China
maweiyu2022@ia.ac.cn

Dongyu Xu
Institute of Automation Chinese Academy of Sciences
Beijing, China
xudongyu2023@ia.ac.cn

Shu Lin
Institute of Automation Chinese Academy of Sciences
Beijing, China
shu.lin@ia.ac.cn

Haifeng Zhang
Institute of Automation Chinese Academy of Sciences
Beijing, China
haifeng.zhang@ia.ac.cn

Jun Wang
University College London
London, United Kingdom
jun.l.wang@ucl.ac.uk

Abstract

We present Adaptive Command, a novel framework integrating large language models (LLMs) with behavior trees for real-time strategic decision-making in StarCraft II. Our system focuses on enhancing human-AI collaboration in complex, dynamic environments through natural language interactions. The framework comprises: (1) an LLM-based strategic advisor, (2) a behavior tree for action execution, and (3) a natural language interface with speech capabilities. User studies demonstrate significant improvements in player decision-making and strategic adaptability, particularly benefiting novice players and those with disabilities. This work contributes to the field of real-time human-AI collaborative decision-making, offering insights applicable beyond RTS games to various complex decision-making scenarios.

Keywords

LLM Agents, StarCraft II, Human-AI Collaboration, Adaptive Decision-Making, Game AI

ACM Reference Format:
Weiyu Ma, Dongyu Xu, Shu Lin, Haifeng Zhang, and Jun Wang. 2024. Adaptive Command : Real-Time Policy Adjustment via Language Models in StarCraft II. In *Sixth International Conference on Distributed Artificial Intelligences (DAI '24), December 18–22, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3719545.3721029

1 Introduction

Real-time strategy (RTS) games, exemplified by StarCraft II (SC2), present formidable challenges in AI research, demanding rapid tactical decisions and adaptive long-term planning. SC2's complex gameplay, detailed in Section 9, encompasses resource management, base construction, and military command, serving as an ideal testbed for advanced AI systems. While previous efforts, such as DeepMind's AlphaStar [22], have shown AI's potential in mastering such games, recent advancements have begun to explore the integration of Large Language Models (LLMs) in this domain.

The rapid advancement of LLMs in reasoning, planning, and decision-making has opened new possibilities for their application in complex gaming environments. A notable contribution in this area is the TextStarCraft II project [16], which developed a specialized environment for assessing LLMs in real-time strategic scenarios within SC2. Their work demonstrated that fine-tuned LLMs could perform on par with Gold-level players in real-time matches, showcasing the potential of LLMs in strategic gameplay.

Building upon these developments, we identify a gap in frameworks that leverage LLMs for enhancing human-AI collaboration in real-time strategy decision-making and long-term planning in environments like StarCraft II. While TextStarCraft II focused on evaluating LLMs' autonomous performance, our work aims to explore how LLMs can augment human gameplay through real-time interaction and collaboration.

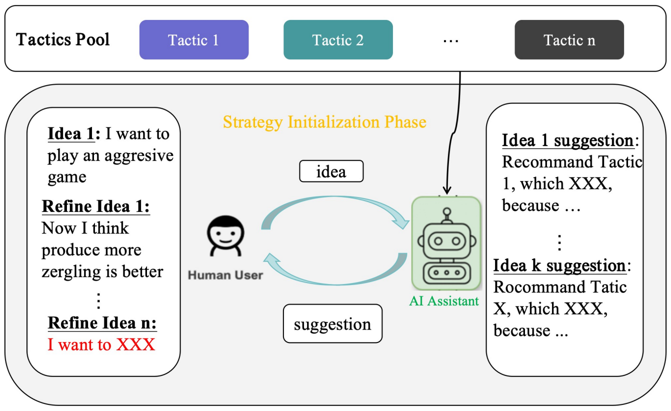
To address this gap, we present Adaptive Command, a system based on TextStarCraft II that extends the capabilities of LLMs in StarCraft II beyond autonomous play to facilitate human-AI collaboration. Our approach introduces a language-conditioned and adjustable StarCraft II policy, leveraging the strengths of LLMs in reasoning and strategic planning, while integrating them with behavior trees for tactical execution.

Adaptive Command introduces several key innovations:

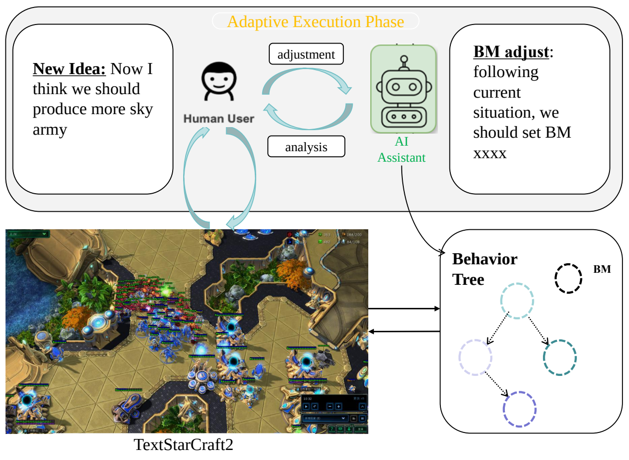
- **Language-Conditioned Policy**: Integration of LLMs with behavior trees provides a flexible policy that adapts to player input and game state.
- **Real-time LLM-assisted Decision Making**: The system offers adaptive, context-aware strategic advice during gameplay, enhancing player decision-making capabilities.
- **Natural Language Interface**: Players interact with the AI assistant using natural language, including voice commands, facilitating intuitive strategic discussions.
- **Dynamic Strategy Adaptation**: Real-time policy adjustments allow players to modify strategies based on evolving game conditions and AI recommendations.

This work is licensed under a Creative Commons Attribution International 4.0 License.

DAI '24, Singapore, Singapore
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1007-0/24/12
https://doi.org/10.1145/3719545.3721029


Tactics Pool
Tactic 1
Tactic 2
…
Tactic n
Strategy Initialization Phase
Idea 1: I want to play an aggresive game
Refine Idea 1: Now I think produce more zergling is better
Refine Idea n: I want to XXX
idea
suggestion
Human User
AI Assistant
Idea 1 suggestion: Recommand Tactic 1, which XXX, because …
Idea k suggestion: Rocommand Tatic X, which XXX, because ...

(a) Strategy Initialization Phase


Adaptive Execution Phase
New Idea: Now I think we should produce more sky army
Human User
adjustment
analysis
AI Assistant
BM adjust: following current situation, we should set BM xxxx
Behavior Tree
BM
TextStarCraft2

(b) Adaptive Execution Phase

Figure 1: Adaptive Command Framework: Two-stage process integrating LLMs with behavior trees for strategic decision-making in StarCraft II.

- **Enhanced Accessibility**: Incorporation of speech-to-text and text-to-speech capabilities aims to broaden the accessibility of complex RTS gameplay.

Our research investigates how this language-conditioned, human-AI collaborative approach can enhance player decision-making, strategic flexibility, and overall game performance. Through comprehensive user studies, we evaluate the system's effectiveness across various player skill levels and assess its potential in improving gaming experiences.

This work contributes to the broader field of human-AI collaboration in complex, real-time decision-making environments. While centered on StarCraft II, the insights and methodologies developed have potential applications beyond gaming, offering new perspectives on integrating AI assistance in dynamic, strategic scenarios across various domains. By focusing on the collaborative aspect between humans and AI, our work extends the capabilities of LLM-based systems in RTS games, potentially opening new avenues for AI-assisted gameplay and decision-making in complex environments.

2 Related Work

StarCraft II Full Game AI: StarCraft AI research, initially focused on StarCraft I with developments like BiCNet [19] for multi-agent coordination, has significantly advanced in the StarCraft II era. The release of PySC2 [23] by DeepMind, coupled with Blizzard's game replays, propelled this research field. A key breakthrough was AlphaStar [22], which achieved Grandmaster level and defeated top players, demonstrating the potential of RL in complex environments.

Subsequent research [2, 8, 10, 25] expanded upon these foundations. Mini-AlphaStar[14] simplified input variables without compromising learning effectiveness. TG[12] and HierNet-SC2[13] explored efficient RL strategies, with the latter bypassing supervised pre-training. AlphaStar Unplugged[17] represented a leap in offline RL using human replays. TStarBotsX[7] and SCC[26] furthered federated learning approaches, achieving notable success against master and grandmaster level players.

Recent advancements include DI-star[1], which is accessible for home computer deployment, and ROA-Star[9], enhancing AlphaStar's training framework with goal-conditioned exploiters and refined opponent modeling techniques. ROA-Star's practical tests against professional players have shown impressive results, marking significant progress in real-time strategy AI.

Grounding Natural Language Commands to StarCraft II[28] made a pioneering contribution in this direction with their work on Grounding Natural Language Commands to StarCraft II Game States.

Building upon this integration of language and game mechanics, TextStarCraft II [16] further advanced the field by introducing a specialized environment for evaluating Large Language Models (LLMs) in real-time strategic scenarios within SC2. This work highlights the potential of language models in strategic gameplay and decision-making.

These advancements highlight the evolution of StarCraft II AI research, integrating natural language understanding with traditional approaches. This progression aims to create AI systems that interact with complex strategic environments in more human-like ways.

LLM Agents and Benchmarks: The surge of large language models (LLMs) in late 2022, exemplified by models such as , GPT-3.5,GPT4 [18], and subsequent developments like LLaMA [4] and Qwen [1, 30], has significantly propelled research into LLM agents. Projects like React , Chain of Thought [27, 29, 31], and AutoGPT laid the groundwork for more sophisticated implementations. In gaming environments, initiatives such as GITM [32] and Voyager [24] have demonstrated LLM agents' adaptability to various tasks and open-world scenarios. Environments like TextWorld [3] and ALFWorld [20] have integrated text-based strategy and action execution. Recent advancements include Cradle [21], which introduces a General Computer Control setting for broader software interaction, and strategic discussion agents for games like One Night Ultimate Werewolf [11]. Benchmarking platforms like AGENTBENCH [15] play

[1]https://github.com/opendilab/DI-star

a crucial role in evaluating these developments in comprehensive, open-ended contexts.

Human-AI Collaboration in Gaming: Recent advancements in Large Language Models (LLMs) have opened new avenues for human-AI collaboration in complex gaming environments[5]. MindAgent [6] introduces a framework for evaluating AI planning and coordination in multi-agent gaming scenarios, while practical applications like the AI teammate in "Naraka: Bladepoint" mobile game [2] demonstrate the potential of AI assistants in commercial games.

These developments showcase progress in AI-assisted gaming and human-AI interaction, yet there remains a gap in leveraging such technologies for real-time strategy assistance in complex RTS games like StarCraft II. Our work aims to address this gap by introducing a language-conditioned, adaptive system that enhances human decision-making in StarCraft II, while also exploring its potential to improve accessibility and overall gaming experience.

3 Method

This section presents the architecture and key components of Adaptive Command, our novel system based on TextStarCraft II for enhancing human decision-making in StarCraft II through real-time AI assistance.

3.1 System Architecture

Adaptive Command consists of five main components:

1. **Game State Processor**: Employs the Chain of Summarization (CoS) method to efficiently process and compress game information.
2. **LLM-based Strategic Advisor**: Utilizes pre-trained models like GPT-4 and DeepSeek to analyze game states and generate strategic advice.
3. **Behavior Tree Framework**: Translates high-level strategies into executable game actions, controlled by Behavior Modulators (BMs).
4. **Natural Language Interface**: Facilitates communication between the player and the AI system through voice commands and responses.
5. **Real-time Policy Adjustment Mechanism**: Dynamically updates game strategies by modifying BMs based on player input and game state.

Figure 1 illustrates the overall architecture of these components.

3.2 Mathematical Formulation

We formulate Adaptive Command as a two-phase process within a reinforcement learning framework:

Where:

- $\mathcal{E}$ is the StarCraft II game environment
- $s_t \in \mathcal{S}$ is the game state at time step t
- $a_t \in \mathcal{A}$ is the action taken at time step t
- π_t is the policy at time step t
- c_t is the player's instruction at time step t
- r_t is the reward at time step t
- θ represents the parameters of the LLM
- f_{LLM} is the LLM's decision function

Algorithm 1 Adaptive Command in StarCraft II

```
Input: StarCraft II game environment $\mathcal{E}$, language model $f_{LLM}$,
       behavior tree $g_{BT}$
1:  Initialize game state $s_0 = \mathcal{E}$.reset()
2:  $c_0 \leftarrow$ initial player instruction
3:  $\pi_0 \leftarrow f_{LLM}(s_0, c_0, \theta)$                ▷ Initial policy selection
4:  $t \leftarrow 0$
5:  while $\mathcal{E}$ is not terminated do
6:      $a_t \leftarrow g_{BT}(\pi_t, s_t)$                    ▷ Action selection
7:      $s_{t+1}, r_t \leftarrow \mathcal{E}$.step($a_t$)           ▷ Environment interaction
8:      if player provides new instruction $c_{t+1}$ then
9:          $\hat{s}_{t+1} \leftarrow$ CoS($s_{t+1}$)              ▷ Chain of Summarization
10:         $\pi_{t+1} \leftarrow f_{LLM}(\hat{s}_{t+1}, a_t, c_{t+1}, \theta)$   ▷ Policy adjustment
11:     else
12:         $\pi_{t+1} \leftarrow \pi_t$
13:     end if
14:     $t \leftarrow t + 1$
15: end while
```

- g_{BT} is the behavior tree execution function
- *CoS* is the Chain of Summarization function

This formulation captures the two-phase nature of Adaptive Command: 1. Initial Policy Selection (line 3): The LLM selects an initial policy based on the initial game state and player instruction. 2. Real-time Policy Adjustment (lines 8-13): The system adjusts the policy based on new player instructions and the current game state, processed through the Chain of Summarization.

3.3 Game State Processing

We adopt the Chain of Summarization (CoS) method, originally introduced in the TextStarCraft II framework [16], for efficient game state processing. This method involves:

1. **Single-Frame Summarization**: Condenses immediate game state information.
2. **Multi-Frame Summarization**: Aggregates information over multiple frames to capture temporal dynamics.
3. **Strategic Context Integration**: Combines summarized game state with player instructions.

In our algorithm, the CoS function (line 9) processes the raw game state s_{t+1} into a summarized state $\hat{s}_{t+1}$, which serves as input for the LLM. This approach significantly reduces information overload while maintaining strategic clarity, enabling more efficient decision-making processes.

3.4 LLM Integration

We leverage pre-trained LLMs without additional fine-tuning, allowing them to apply their general language understanding and reasoning capabilities to StarCraft II strategy. The LLM's decision function f_{LLM} is used for both initial policy selection (line 3) and real-time policy adjustment (line 10). It interprets player queries c_t, analyzes game states s_t or $\hat{s}_{t+1}$, and generates tactical suggestions in the form of a policy π_t. The parameters θ represent the LLM's knowledge and are kept constant during gameplay.

[2]https://fuxi.163.com/database/1179

3.5 Behavior Tree and Modulators

The Behavior Tree Framework, represented by function g_{BT} (line 6), translates the high-level policy π_t into concrete game actions a_t. The Behavior Modulators (BMs) serve as the interface between the abstract policy and the specific game commands. These BMs are dynamically adjusted based on the policy π_t generated by the LLM and the current game state s_t, resulting in actions a_t that are executed in the game environment.

3.6 Voice Interface

The system incorporates Speech-to-Text (whisper-small) and Text-to-Speech (edge-tts) technologies to enable hands-free interaction. This allows players to input instructions c_t verbally and receive policy suggestions π_t as voice output, corresponding to the player input in lines 3 and 8 of our algorithm.

3.7 StarCraft II Integration

Adaptive Command integrates with StarCraft II, represented as the environment $\mathcal{E}$ in our algorithm. It receives real-time game state information s_t (line 7) and executes commands a_t through the Behavior Tree Framework, providing a seamless experience without disrupting normal gameplay.

4 Interaction Pipeline

The Adaptive Command system facilitates a dynamic and interactive gameplay experience, combining AI assistance with traditional StarCraft II gameplay.

4.1 Initial Policy Selection

At the start of the game (corresponding to lines 1-3 in our algorithm):

(1) The game environment $\mathcal{E}$ is initialized, providing the initial state s_0.
(2) The player interacts with the LLM through natural language to discuss initial strategy, providing c_0.
(3) Based on the player's input c_0 and initial game state s_0, the LLM recommends the most suitable rule-based policy $\pi_0 = f_{LLM}(s_0, c_0, \theta)$.
(4) The player can accept the recommendation or request alternatives, refining the strategy through continued dialogue.

4.2 Real-time Policy Adjustment

During gameplay (corresponding to lines 5-14 in our algorithm):

(1) The system selects actions $a_t = g_{BT}(\pi_t, s_t)$ based on the current policy and game state.
(2) The game environment evolves: $s_{t+1}, r_t = \mathcal{E}.\text{step}(a_t)$.
(3) If the player provides new instructions c_{t+1}:
   - The system processes the current game state using the Chain of Summarization: $\hat{s}_{t+1} = \text{CoS}(s_{t+1})$.
   - The LLM analyzes the summarized game state and player's input to suggest tactical adjustments: $\pi_{t+1} = f_{LLM}(\hat{s}_{t+1}, a_t, c_{t+1}, \theta)$.
(4) Upon player approval, the system modifies the behavior tree to implement the new strategy.

4.3 Voice-based Interaction

To facilitate hands-free interaction:

(1) All interactions between the player and the LLM are conducted through a voice interface.
(2) Speech-to-Text (STT) technology converts player's voice commands into text instructions c_t for LLM processing.
(3) Text-to-Speech (TTS) technology converts LLM's policy suggestions π_t into voice output for the player.
(4) This enables players to focus on gameplay while discussing strategy, corresponding to the input of c_t and output of π_t in our algorithm.

4.4 Traditional Gameplay Integration

To maintain player agency:

(1) Players retain full control over traditional gameplay mechanics while the AI assistant is available.
(2) Keyboard and mouse inputs for direct unit and resource management remain functional, as in standard StarCraft II gameplay.
(3) This hybrid approach allows players to leverage both AI strategic assistance π_t and their own micro-management skills to influence the final actions a_t executed in the game.

4.5 Win Conditions

The victory conditions in Adaptive Command remain consistent with standard StarCraft II rules:

- The primary win condition is the destruction of all enemy buildings.
- All other standard StarCraft II victory and defeat scenarios apply.

This interaction pipeline demonstrates how Adaptive Command seamlessly integrates AI assistance with traditional StarCraft II gameplay, offering players a unique, adaptive, and interactive strategic experience while following the mathematical formulation presented in our algorithm.

5 Experiment Design and Evaluation

5.1 Experimental Setup

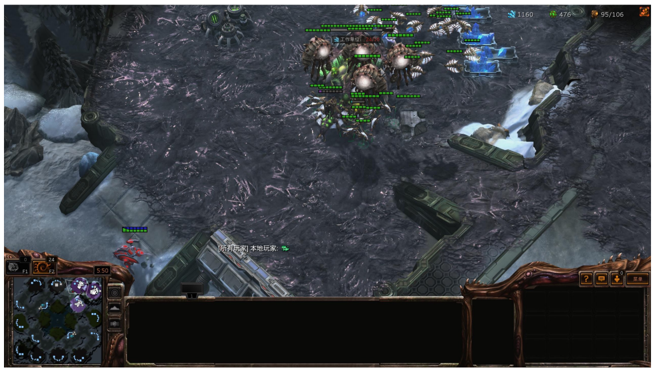
5.1.1 Participants. We recruited 12 participants with varying levels of StarCraft II experience through online gaming communities and university networks. Participants were categorized as follows:

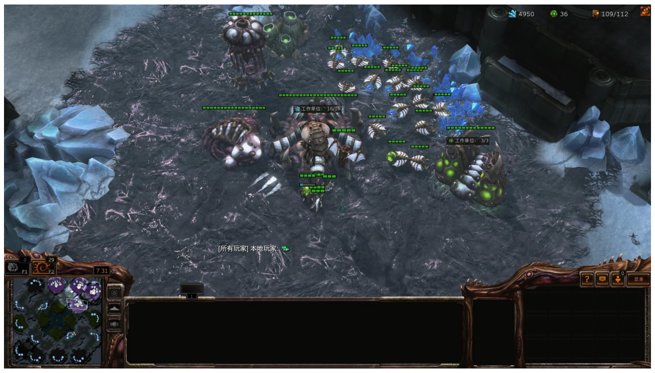
- 4 novice players (less than 50 hours of gameplay)
- 4 intermediate players (50-1000 hours of gameplay)
- 4 expert players (over 1000 hours of gameplay and achieved grandmaster level)

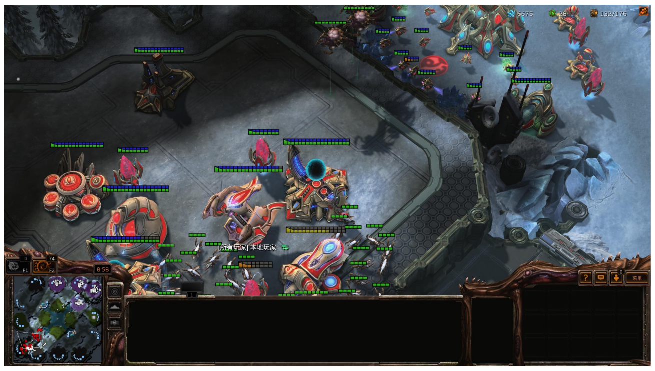
Table 1 shows the MMR (Matchmaking Rating) distribution of the participants.

5.1.2 Task Design. Each participant played a total of 6 games as Zerg against the built-in AI (Protoss) opponent set to level 6 (Very Hard). The games were divided into two conditions:

- Control condition: 3 games played using standard StarCraft II gameplay
- Experimental condition: 3 games played with the assistance of Adaptive Command


(a) Initial strategy


(b) Transition to air units


(c) Balanced army composition

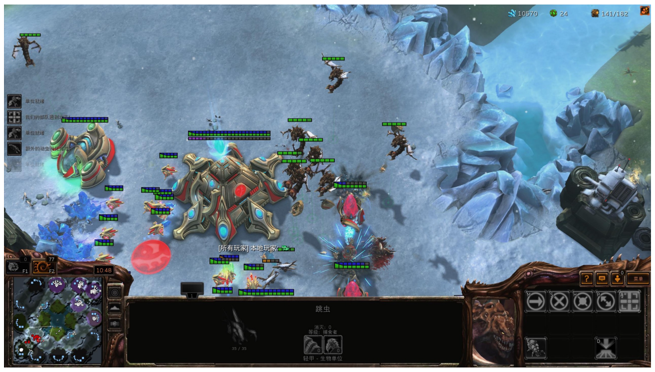

(d) Final victory

Figure 2: Adaptive Command in action: Real-time strategy adjustment through human-AI collaboration

Phase	Human Input	LLM Response
Strategy Init. Phase (Fig. 2a)	"Given the opponent is using heavily armored units, what is the best counter-strategy?"	Recommends Roach-Hydra composition for balance of survivability and damage against armored units.
Adaptive Exec. Phase (Fig. 2b, 2c)	"I want to play a sky army style"	Suggests transitioning to Mutalisks and Corruptors to counter Protoss air units and control the map.
	"I think we should also produce some ground army, right?"	Advises maintaining a small ground force for versatility and defense against various Protoss ground units.

Table 1: Player MMR in Experiment

Player ID	Skill Level	MMR
1-4	Novice	–
5	Intermediate	1923
6	Intermediate	2599
7	Intermediate	3094
8	Intermediate	3496
9	Expert	5600
10	Expert	5513
11	Expert	5314
12	Expert	5005

The order of conditions was counterbalanced across participants to mitigate learning effects.

Games were played on three 2023 ladder maps: 'Altitude LE.SC2Map', 'Gresvan LE.SC2Map', and 'Babylon LE.SC2Map'. These maps were chosen to represent current competitive play environments. There was no time limit imposed on the games.

5.1.3 System Configuration.

- LLM: Gemini 1.5 Flash, DeepSeek-V2-0628, GPT-4-turbo-2024-04-09, GPT-4o-2024-08-06. These models were selected for their high performance and availability as commercial APIs.

- Behavior Tree: Custom implementation based on TextStarCraft2, python-sc2[3], and the SC2 AI community[4] resources

5.2 Evaluation Metrics

We employed both quantitative and qualitative measures to evaluate the effectiveness of Adaptive Command:

5.2.1 Quantitative Metrics.

(1) Win Rate: Percentage of games won against the AI opponent in each condition.
(2) Instruction Following: Degree to which the system accurately implemented player instructions in the experimental condition, rated on a 1-5 scale by the research team. A score of 1 indicates complete failure to understand instructions, while 5 indicates perfect comprehension and corresponding adjustments.

5.2.2 Qualitative Metrics.

(1) Helpfulness: Players' assessment of how helpful the Adaptive Command system was during gameplay, rated on a 1-5 scale. A score of 1 indicates not helpful at all, while 5 indicates extremely helpful.

5.3 Limitations

While our study provides valuable insights into the effectiveness of Adaptive Command, several limitations should be noted:

- **Sample Size:** The study involved only 12 participants, which may limit the generalizability of our findings. A larger sample size could provide more robust and representative results.
- **AI Opponent:** Our experiments were conducted against the built-in AI of StarCraft II. While this ensures consistency across trials, it may not fully capture the complexity and unpredictability of human opponents in real competitive scenarios.
- **Game Version Discrepancy:** There exists a potential mismatch between the game version used during the pre-training of the large language models and the version used in our current experiments. This discrepancy may affect the relevance and accuracy of the AI's strategic advice, as the game's meta and balance may have evolved.
- **Framework Performance Limitations:** The Adaptive Command system relies on a behavior tree for execution, which may inherently limit its performance compared to more advanced reinforcement learning (RL) approaches. While our framework offers interpretability and ease of modification, it may not achieve the same level of optimization and adaptability as RL-based systems.

6 Result Analysis

6.1 Quantitative Results

6.1.1 Win Rates. We observed significant differences in win rates between the control group (standard gameplay) and the experimental group (using Adaptive Command). The percentages represent the proportion of games won against the level 6 (Very Hard) AI opponent:

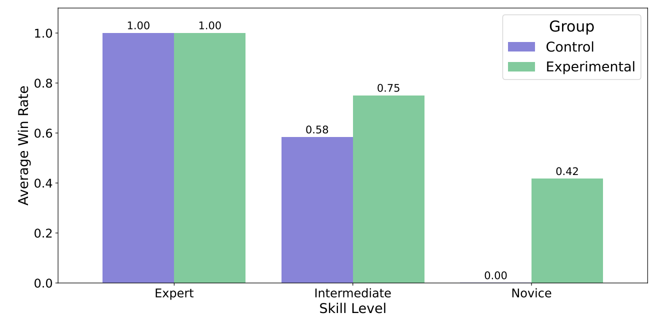


Figure 3: Average Win Rate by Skill Level and Group

- Novice players: 0% (control) vs. 42% (experimental)
- Intermediate players: 58% (control) vs. 75% (experimental)
- Expert players: 100% (control) vs. 100% (experimental)

These results suggest that Adaptive Command provides the most substantial benefit to novice players, with a 42 percentage point increase in win rate. The impact on intermediate players is also notable, with a 17 percentage point improvement, while expert players maintained their perfect performance in both conditions.

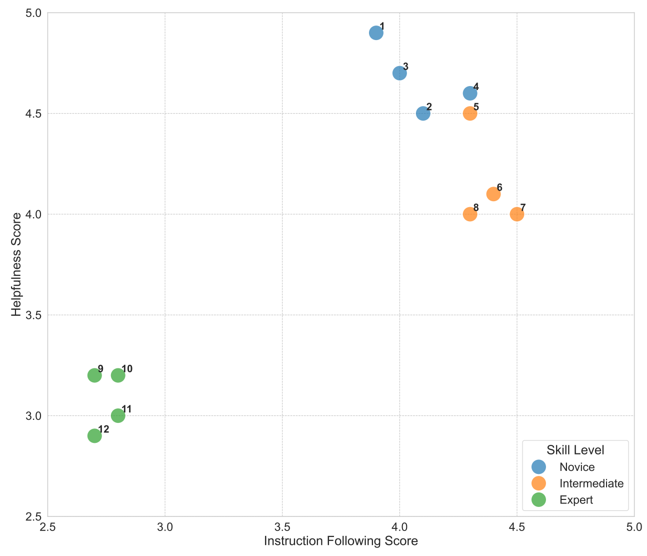


Figure 4: Individual participant scores for Instruction Following and Helpfulness. Each point represents a participant, with numbers indicating participant IDs. Point size and color denote skill level (Novice, Intermediate, Expert). Instruction Following measures the system's ability to accurately implement player instructions, while Helpfulness reflects the perceived usefulness of the Adaptive Command system.

6.1.2 Instruction Following and Helpfulness. The average instruction following and helpfulness ratings (on a 1-5 scale) for Adaptive Command were:

[3]https://github.com/BurnySc2/python-sc2
[4]https://aiarena.net/

- Novice players: Instruction Following 4.1, Helpfulness 4.7
- Intermediate players: Instruction Following 4.4, Helpfulness 4.2
- Expert players: Instruction Following 2.8, Helpfulness 3.1

Novice and intermediate players reported high scores for both metrics, indicating that the system accurately implemented their instructions and was perceived as highly helpful. Interestingly, expert players reported significantly lower scores for both metrics, suggesting that the system may not have met their more advanced needs or expectations.

6.2 Discussion of Results

The results demonstrate that Adaptive Command provides varying benefits across different skill levels:

- For novice players, the system serves as a crucial learning tool, providing basic automated strategies that significantly improve their performance. The dramatic increase in win rate (from 0% to 42%) and high helpfulness score (4.7) reflect the system's effectiveness in bridging the knowledge gap for beginners.
- Intermediate players also benefit substantially, with improved win rates (from 58% to 75%) and high instruction following (4.4) and helpfulness (4.2) scores. This suggests that the system effectively helps translate their theoretical knowledge into practical gameplay strategies.
- Expert players, while maintaining perfect win rates, reported lower scores in instruction following (2.8) and helpfulness (3.1). This indicates challenges in meeting the complex needs of high-level play, possibly due to the sophisticated nature of expert strategies and instructions.

These findings highlight the system's potential as a skill development tool, particularly for novice and intermediate players, while also revealing areas for improvement in supporting expert-level gameplay.

7 Discussion, Future Work, and Conclusion

7.1 Discussion

Our study on Adaptive Command reveals significant insights into the potential of Large Language Models (LLMs) in enhancing human-AI collaboration in complex gaming environments:

- **Skill Development Tool**: The system proves highly effective for novice and intermediate players, serving as both a learning platform and a performance enhancer. This demonstrates the potential of AI assistants in accelerating skill acquisition in complex games.
- **Cognitive Load Reduction**: Across all skill levels, Adaptive Command helps reduce the cognitive load associated with macro management. This allows players to focus more on tactical decision-making and micro-management, potentially leading to more engaging and strategic gameplay experiences.
- **Language-Conditioned Policy Effectiveness**: The high instruction following ratings, particularly for novice and intermediate players, showcase the system's ability to translate natural language instructions into game actions. This opens up new possibilities for intuitive human-AI interaction in gaming.
- **Scalability Challenges**: The disparity in effectiveness across skill levels, particularly the lower scores from expert players, highlights the challenges in creating AI systems that can cater to all levels of expertise. This points to the need for more advanced, flexible AI models that can adapt to varying levels of player sophistication.

7.2 Future Work

Building upon these insights, our future research will focus on:

- Integrating LLMs with reinforcement learning models to create a more adaptive and high-performance system.
- Developing more sophisticated natural language processing capabilities to better interpret and execute complex, expert-level instructions.
- Exploring ways to dynamically adjust the level of AI assistance based on real-time assessment of player skill and needs.
- Investigating the application of this technology to other complex, decision-intensive environments beyond gaming.

We anticipate that these advancements will not only enhance gaming experiences but also contribute to broader applications of human-AI collaboration in fields such as education, business strategy, and defense.

7.3 Conclusion

Adaptive Command represents a significant step forward in creating language-conditioned AI assistants for complex decision-making environments. While current limitations exist, particularly in matching expert-level play, the system demonstrates remarkable potential in:

- Accelerating skill development for novice and intermediate players.
- Providing a novel approach to human-AI collaboration in real-time strategy games.
- Offering insights into how language models can enhance human decision-making in complex, time-sensitive scenarios.

This research not only contributes to the field of game AI but also opens up new avenues for exploring human-AI collaboration in various industries. As we continue to refine and expand this technology, we anticipate significant advancements in our understanding and application of AI-assisted decision-making in complex, real-world environments.

8 Appendix: Introduction to StarCraft II

StarCraft II (SC2) is a real-time strategy (RTS) game developed by Blizzard Entertainment, renowned for its strategic depth and complexity. It serves as both a popular e-sport and a benchmark for AI research in strategic gameplay.

8.1 Gameplay Overview

Players choose from three distinct species: Terrans, Protoss, and Zerg, each offering unique units and strategies. The game revolves around four key elements:

(1) **Resource Management**: Gathering minerals and vespene gas to fund operations.
(2) **Base Construction**: Expanding and fortifying positions efficiently.
(3) **Army Composition**: Building and controlling a balanced mix of units.
(4) **Strategic Planning**: Adapting strategies in response to opponent moves.

9 Introduction to StarCraft II

StarCraft II (SC2) is a real-time strategy (RTS) game developed by Blizzard Entertainment, renowned for its strategic depth and complexity. It serves as both a popular e-sport and a benchmark for AI research in strategic gameplay.

9.1 Gameplay Overview

Players choose from three distinct species: Terrans, Protoss, and Zerg, each offering unique units and strategies. The game revolves around four key elements:

(1) **Resource Management**: Gathering minerals and vespene gas to fund operations.
(2) **Base Construction**: Expanding and fortifying positions efficiently.
(3) **Army Composition**: Building and controlling a balanced mix of units.
(4) **Strategic Planning**: Adapting strategies in response to opponent moves.

9.2 Competitive Scene and Research Platform

Since its 2010 launch, SC2 has been a cornerstone of e-sports, featuring in major tournaments worldwide. Its demanding mechanics and strategic complexity have also made it a valuable platform for AI research, particularly in areas of real-time decision making, strategic planning, and human-AI collaboration.

The game's rich strategic landscape, coupled with its clear victory conditions and quantifiable performance metrics, makes it an ideal testbed for developing and evaluating AI systems capable of complex, multi-step reasoning and execution in dynamic environments.

References

[1] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609* (2023).
[2] Filippos Christianos, Georgios Papoudakis, Muhammad A Rahman, and Stefano V Albrecht. 2021. Scaling multi-agent reinforcement learning with selective parameter sharing. In *International Conference on Machine Learning*. PMLR, 1989–1998.
[3] Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. 2019. Textworld: A learning environment for text-based games. In *Computer Games: 7th Workshop, CGW 2018, Held in Conjunction with the 27th International Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, July 13, 2018, Revised Selected Papers 7*. Springer, 41–75.
[4] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).
[5] Xueyang Feng, Zhi-Yuan Chen, Yujia Qin, Yankai Lin, Xu Chen, Zhiyuan Liu, and Ji-Rong Wen. 2024. Large Language Model-based Human-Agent Collaboration for Complex Task Solving. *arXiv preprint arXiv:2402.12914* (2024).
[6] Ran Gong, Qiuyuan Huang, Xiaojian Ma, Hoi Vo, Zane Durante, Yusuke Noda, Zilong Zheng, Song-Chun Zhu, Demetri Terzopoulos, Li Fei-Fei, et al. 2023. MindAgent: Emergent Gaming Interaction. *arXiv preprint arXiv:2309.09971* (2023).
[7] Lei Han, Jiechao Xiong, Peng Sun, Xinghai Sun, Meng Fang, Qingwei Guo, Qiaobo Chen, Tengfei Shi, Hongsheng Yu, Xipeng Wu, et al. 2020. Tstarbot-x: An open-sourced and comprehensive study for efficient league training in starcraft ii full game. *arXiv preprint arXiv:2011.13729* (2020).
[8] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. 2024. Learning multi-agent communication from graph modeling perspective. *arXiv preprint arXiv:2405.08550* (2024).
[9] Ruozi Huang, Xipeng Wu, Hongsheng Yu, Zhong Fan, Haobo Fu, QIANG FU, and Yang Wei. 2023. A Robust and Opponent-Aware League Training Method for StarCraft II. In *Thirty-seventh Conference on Neural Information Processing Systems*.
[10] Hao Jiang, Dianxi Shi, Chao Xue, Yajie Wang, Gongju Wang, and Yongjun Zhang. 2021. Multi-agent deep reinforcement learning with type-based hierarchical group communication. *Applied Intelligence* 51 (2021), 5793–5808.
[11] Xuanfa Jin, Ziyan Wang, Yali Du, Meng Fang, Haifeng Zhang, and Jun Wang. 2024. Learning to Discuss Strategically: A Case Study on One Night Ultimate Werewolf. *arXiv preprint arXiv:2405.19946* (2024).
[12] Ruo-Ze Liu, Haifeng Guo, Xiaozhong Ji, Yang Yu, Zhen-Jia Pang, Zitai Xiao, Yuzhou Wu, and Tong Lu. 2021. Efficient reinforcement learning for starcraft by abstract forward models and transfer learning. *IEEE Transactions on Games* 14, 2 (2021), 294–307.
[13] Ruo-Ze Liu, Zhen-Jia Pang, Zhou-Yu Meng, Wenhai Wang, Yang Yu, and Tong Lu. 2022. On efficient reinforcement learning for full-length game of starcraft ii. *Journal of Artificial Intelligence Research* 75 (2022), 213–260.
[14] Ruo-Ze Liu, Wenhai Wang, Yanjie Shen, Zhiqi Li, Yang Yu, and Tong Lu. 2021. An Introduction of mini-AlphaStar. *arXiv preprint arXiv:2104.06890* (2021).
[15] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688* (2023).
[16] Weiyu Ma, Qirui Mi, Yongcheng Zeng, Xue Yan, Yuqiao Wu, Runji Lin, Haifeng Zhang, and Jun Wang. 2024. Large Language Models Play StarCraft II: Benchmarks and A Chain of Summarization Approach. arXiv:2312.11865 [cs.AI] https://arxiv.org/abs/2312.11865
[17] Michael Mathieu, Sherjil Ozair, Srivatsan Srinivasan, Caglar Gulcehre, Shangtong Zhang, Ray Jiang, Tom Le Paine, Konrad Zolna, Richard Powell, Julian Schrittwieser, et al. 2021. Starcraft ii unplugged: Large scale offline reinforcement learning. In *Deep RL Workshop NeurIPS 2021*.
[18] OpenAI. 2023. GPT-4 Technical Report. *ArXiv* abs/2303.08774 (2023). https://api.semanticscholar.org/CorpusID:257532815
[19] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. 2017. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069* (2017).
[20] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768* (2020).
[21] Weihao Tan, Ziluo Ding, Wentao Zhang, Boyu Li, Bohan Zhou, Junpeng Yue, Haochong Xia, Jiechuan Jiang, Longtao Zheng, Xinrun Xu, et al. 2024. Towards general computer control: A multimodal agent for red dead redemption ii as a case study. *arXiv preprint arXiv:2403.03186* (2024).
[22] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.
[23] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. 2017. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782* (2017).
[24] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291* (2023).
[25] Xiangjun Wang, Junxiao Song, Penghui Qi, Peng Peng, Zhenkun Tang, Wei Zhang, Weimin Li, Xiongjun Pi, Jujie He, Chao Gao, et al. 2021. SCC: An efficient deep reinforcement learning agent mastering the game of StarCraft II. In *International conference on machine learning*. PMLR, 10905–10915.
[26] Xiangjun Wang, Junxiao Song, Penghui Qi, Peng Peng, Zhenkun Tang, Wei Zhang, Weimin Li, Xiongjun Pi, Jujie He, Chao Gao, et al. 2021. SCC: An efficient deep reinforcement learning agent mastering the game of StarCraft II. In *International conference on machine learning*. PMLR, 10905–10915.
[27] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171* (2022).
[28] Nicholas Waytowich, Sean L Barton, Vernon Lawhern, Ethan Stump, and Garrett Warnell. 2019. Grounding natural language commands to StarCraft II game states

for narration-guided reinforcement learning. In *Artificial intelligence and machine learning for multi-domain operations applications*, Vol. 11006. SPIE, 267–276.

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

[30] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671* (2024).

[31] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629* (2022).

[32] Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, et al. 2023. Ghost in the Minecraft: Generally Capable Agents for Open-World Enviroments via Large Language Models with Text-based Knowledge and Memory. *arXiv preprint arXiv:2305.17144* (2023).