Boosting Robot Behavior Generation with Large Language Models and Genetic Programming

Aaron Verdaguer Gonzalez[1,2], Luis Merino Cabañas[2], Magí Dalmau-Moreno[1,3] and Néstor García[1]
[1]Eurecat, Centre Tecnològic de Catalunya, Unitat de Robòtica i Automatització, Spain
[2]Universidad Pablo Olavide, Department of Robotics, Spain
[3]Universitat Pompeu Fabra, Spain
aaron.verdaguer@eurecat.org, lmercab@upo.es, magi.dalmau@eureacat.org, nestor.garcia@eurecat.org

Abstract— **Mobile robots are increasingly ubiquitous in modern society, necessitating more human-like interaction capabilities, such as following operator instructions or collaborating with humans. Conventional robot programming methods often fall short in achieving these complex behaviors. Behavior Trees (BTs) offer a promising alternative due to their modularity, scalability and reactivity. We propose using Large Language Model (LLM) assistants to decompose task descriptions into executable BTs. The BTs are then refined using Genetic Programming and a low-resource simulator, eliminating the need for fine-tuning LLMs. Our approach accelerates behavior generation, enhances applicability in diverse environments, and democratizes the process for non-experts. Besides, it enables the generation of adaptable behaviors tailored to various scenarios.**

I. INTRODUCTION

Mobile robots are becoming increasingly prevalent in various domains, including manufacturing, logistics, and healthcare, where they often need to interact with people and operate in dynamic environments [12], [15], [17]. However, the problem that arises with traditional methods of programming robot behaviors, like Hierarchical Finite State Machines, subsumption architectures or Teleo-Reactives [22] is that they often lack the flexibility and adaptability required for these scenarios [5], [2]. Behavior Trees (BTs) offer a compelling solution to this challenge [26]. They have emerged as a powerful tool for representing and executing complex robot behaviors in a structured and modular fashion, offering advantages over their predecessors such as hierarchical organization, reusability of micro behaviors, and ease of readability by users of varying technical expertise [8], [2]. Despite that, while Large Language Models (LLMs) offer promising capabilities for BT generation, there are potential downsides that warrant careful consideration, such as misinterpretations or hallucination, a need of a large dataset for fine tuning or computational resources required for deploying an onboard reliable LLMs in a mobile robot.

The process of generating BTs entails understanding the goal and transforming it into executable sequences of actions and conditions. This task demands a deep understanding of the world, how actions can affect the environment and a high level of linguistic comprehension when the goal is provided as text. Our research aims to address this challenge by proposing an approach that integrates LLMs into the BT generation process. By leveraging the natural language understanding capabilities of LLMs and incorporating post-processing techniques mentioned in Section III, we seek to automate and streamline the task of BT generation. This integration enables mobile robots to adapt quickly to user specified tasks and unpredictable environments.

In recent years, there has been a surge of interest in harnessing LLMs for natural language understanding tasks [19]. These models have showcased remarkable capabilities in understanding and generating human-like text across diverse domains [25]. One of the remarkable aspects of LLMs lies in their capacity to encapsulate vast amounts of world knowledge derived from pre-training on extensive text corpora [28]. This inherent understanding of the world enables LLMs to decipher task descriptions and formulate action plans tailored to the specified objectives [29], [7]. By tapping into their accumulated knowledge, LLMs offer a promising avenue for enhancing the autonomy and problem-solving capabilities of mobile robots, facilitating efficient and contextually appropriate behavior generation in real-world environments [6].

Our proposed methodology can be decomposed in two parts which are the initial creation of a first BT from a natural language task description by a user using three assistants created using ChatGPT 4.0 [23]. And the posterior processing and evaluation using Genetic Programming (GP) and a low resource simulator proposed in [9]. Our paper presents several contributions to the field of autonomous systems and behavior generation. First and foremost, we eliminate the need for fine-tuning LLMs specifically for BT generation, streamlining the process and reducing the associated computational overhead. The fusion of LLMs and GP accelerates behavior generation while facilitating extrapolation to more realistic scenarios, thereby enhancing the applicability of generated behaviors in diverse environments. Our approach is designed to be accessible to non-experts by generating an initial BT and the necessary gene pool for further BT processing just from a given task description.

This process democratizes the process of behavior generation and fostering broader adoption in the society. Furthermore, our approach enables the generation of adaptable behaviors tailored to different scenarios, enhancing the flexibility and robustness of autonomous systems. Through our research, we aim to push the boundaries of BT generation for mobile robots, ultimately paving the way for more intuitive and user-friendly robot programming interfaces.

The rest of the paper is structured as follows. First, we delve into a comprehensive review of the literature surrounding the utilization of LLMs as behavior generators, as well as the application of GP in learning BTs. Then, we provide a detailed description of our proposed methodology, elucidating the intricacies of combining LLMs and GP for behavior generation in autonomous systems. We described the used benchmarking scenarios and engage in a thorough discussion of the results obtained. Finally, we draw insightful conclusions from our experiments and propose directions for future research.

II. RELATED WORK

As mentioned, our BT generation pipeline can be decomposed in two parts which are the initial creation of a first BT using LLMs and the posterior fine tuning and evaluation of the BT. The usage of LLMs as BT generators is starting to thrive. For example, in [1] the authors propose a Phase-Step prompt design that facilitates hierarchical-structured robot task generation, integrated with a behavior-tree-embedding-based search. This approach enables automatic and cross-domain behavior-tree task generation without the need for pre-defined primitive tasks but with the need of a previous task decomposition examples. Their main downside is that the generated BTs are restricted to a sequence of actions resulting more in a sequential plan than a BT, lacking to exploit the main advantages of the different control nodes in BTs.

Other examples of using LLMs for BTs generation are [16], [14]. They introduce different methodologies of training LLMs to become BT generators. Their downside is the need for large text corpora for fine tuning such models. For instace, LLM-BRAIn generates robot BTs from operator commands but needs fine-tuning on 8.5k instruction-following demonstrations in the style of self-instruct using *text-davinchi-003*. Although previously cited papers employ LLMs for generating BTs, they lack detailed scenarios that would allow for a direct comparison of our methodology with theirs [16], [14], or they do not provide the necessary knowledge base for testing their methodologies in our specific scenario [1].

Evolutionary algorithms, such as GP, operate by treating valid BTs as population within a generation. The nodes within the BTs represent the genetic diversity within this population, and processes analogous to those found in nature, such as mutation or crossover, occur to facilitate the evolution of the population into the next generation [21]. As previously mentioned, in our case we have used the GP algorithm proposed in [9] as well as their low resource simulator approach for fine tuning the generated BT. The fitness of each candidate program, which represents how well it solves the problem or performs the task, is evaluated using their fitness function $J = R - \big(\alpha\|s_d - s\|^2 + \beta b + \gamma T\big)$, where R is the reward obtained for completing the task, s is the world state after the BT execution, s_d is the desired goal state, b is the the number of nodes in the BT, and T is the execution time.

III. METHODOLOGY

In our research, we focus the generation of a valid functional BT involving a systematic methodology that integrates user interaction, assistance from specialized tools and a final BT post processing GP approach as depicted in Figure 1. The three assistants are created using OpenAI's Chat GPT 4.0 [23] and their assistants API, starting with the Task Understanding Assistant (TUA).

The TUA extracts key details from user-provided task descriptions, such as manipulable objects and relevant locations, and confirms these with the user before proceeding. This streamlined description shortens the methodology section and reduces the redundancy of explaining every single assistant's function in detail.

Following user confirmation, the task description and associated data are used as input for to the Node Parametrization Assistant (NPA). The associated data includes the addition of previous examples as text files enabling the Retrieval-Augmented Generation (RAG) [4] of the assistant to be triggered if it considers it necessary. This phase involves the parametrization of fundamental action nodes (i.e. move_to_{location}, pick_{object} and place_{object}) and condition nodes (i.e. have_{object}?). The NPA empowers the user to filter out or adding in nodes that may not align or might be needed within the task requirements, thereby refining the node pool for subsequent stages. Parametrized nodes, along with the task description, are stored for future reference and utilization.

In the final assistant phase, the parametrized action nodes, the refined task description and previous examples if apply are forwarded to the Behavior Tree Assistant (BTA). The BTA, just like the NPA, is also doted with RAG. Leveraging the provided information, the BTA orchestrates the synthesis of a simplified action-based BT tailored to address the specified task objectives. The resultant behavior tree encapsulates a hierarchical structure of actions designed to facilitate task resolution within the defined environment. The idea behind this assistants is to decompose a big prompting task into smaller and easier to understand subtasks, similarly to [27].

To ensure that the system is able to work even if the LLMs hallucinate [23], following the generation of a BT by the Behavior Tree Assistant (BTA), a crucial step involves ensuring its validity for its posterior evolution and evaluation. For this, a validation engine is employed to transform the output BT into a structurally and functionally sound tree. The validation engine scrutinizes the BT to identify and rectify any inconsistencies or syntactical errors that may compromise its effectiveness in task execution (see Figure 2).

Once validated, the BT undergoes evaluation and fine-tuning using Genetic Programming techniques within a simulator environment proposed in [10]. The researchers utilize a simplified, deterministic, low-resource simulator that can be interpreted as a state machine. This speeds up the learning process by not requiring the detailed modeling of the robot's physical attributes and interactions, thereby reducing computational costs. The final BT is tested in Gazebo [13].

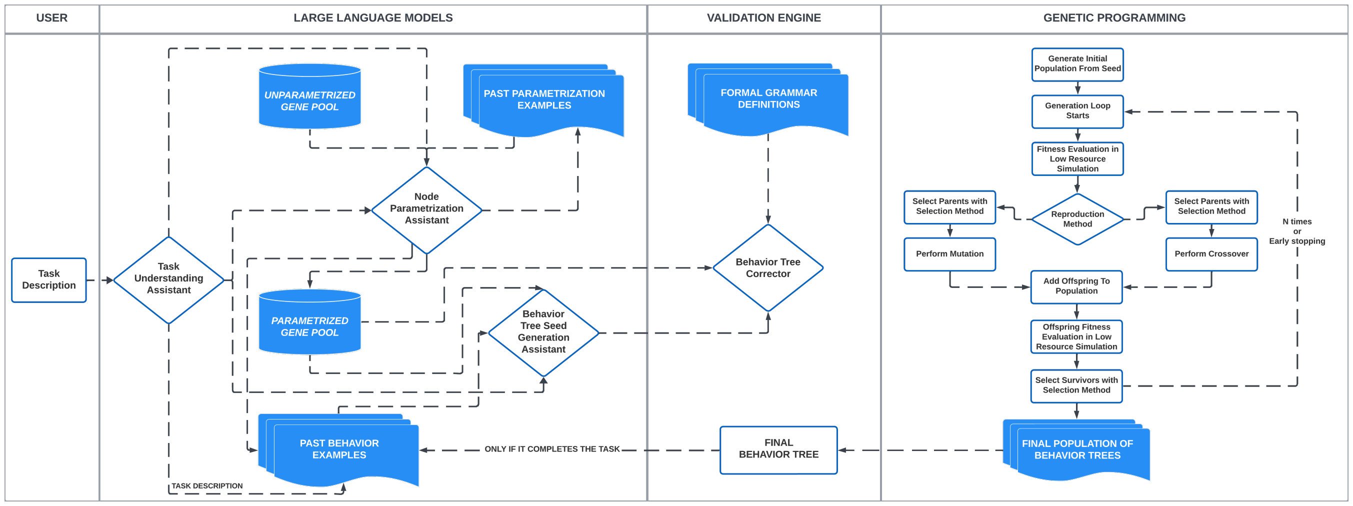
USER
LARGE LANGUAGE MODELS
VALIDATION ENGINE
GENETIC PROGRAMMING
Task Description
Task Understanding Assistant
UNPARAMETRIZED GENE POOL
PAST PARAMETRIZATION EXAMPLES
Node Parametrization Assistant
PARAMETRIZED GENE POOL
Behavior Tree Seed Generation Assistant
PAST BEHAVIOR EXAMPLES
TASK DESCRIPTION
ONLY IF IT COMPLETES THE TASK
FORMAL GRAMMAR DEFINITIONS
Behavior Tree Corrector
FINAL BEHAVIOR TREE
Generate Initial Population From Seed
Generation Loop Starts
Fitness Evaluation in Low Resource Simulation
Reproduction Method
Select Parents with Selection Method
Select Parents with Selection Method
Perform Mutation
Perform Crossover
Add Offspring To Population
Offspring Fitness Evaluation in Low Resource Simulation
Select Survivors with Selection Method
N times or Early stopping
FINAL POPULATION OF BEHAVIOR TREES

Fig. 1: Workflow of the proposed approach.

Genetic Programming offers a systematic approach to iteratively refine the BT's structure and parameters, leveraging evolutionary principles to optimize its functionality and adaptability to diverse scenarios. Through this iterative process, the BT evolves to better align with the task requirements, removing redundancies and enhancing the overall autonomy and efficiency of the system.

In summary, the workflow for BT generation embodies a collaborative endeavor between users and specialized assistants followed by BT validation and polishing. By iteratively refining task descriptions, parametrizing action and condition nodes, and orchestrating BT synthesis, the workflow eases the creation of tailored solutions to complex tasks.

IV. EXPERIMENTS

A. Scenarios

Our experimental evaluation tests the BT generation on three controlled scenarios to assess adaptability, efficiency, and robustness across 3 scenarios outlined in [9]. The consistent setup involves three tables where cubes appear (namely *Table 0*, *Table 1*, and *Table 2*) and a target table (*Table 3*) for cube transportation. The scenarios progress from a fixed cube location on *Table 0* in scenario 1, introducing cube spawn randomness in scenario 2, and culminating in simultaneous spawns across all tables in scenario 3. This setup effectively demonstrates our approach's capability to handle uncertainty and multiple objectives efficiently.

B. ALFRED

The ALFRED (Action Learning From Realistic Environments and Directives) [24] is a benchmark for learning a mapping from natural language instructions and egocentric vision to sequences of actions for household tasks. ALFRED provides a collection of environments with a natural language goal description, a set of instructions and corresponding demonstration trajectories. These instructions outline step-by-step tasks, guiding agents on how to accomplish various household activities such as cooking, cleaning, or organizing objects. The dataset is categorized into different task types, delineating distinct challenges and objectives for agents to tackle. For our experimentation, we randomly selected two tasks from the "pick_and_place_simple" category where agents are tasked with the fundamental action of picking up objects from one location and placing them elsewhere. We randomly chose the tasks of placing a baseball bat on a bed and moving a watch to a glass table. Thereby, we aim to assess the ability of our approach to extrapolate from more hard-coded scenarios to more realistic tasks.

C. Genetic Programming Modifications

In order to refine the generated BT by the assistants and being able to evaluate the generated tree we have used the approach proposed in [9] (using the same GP parameters). Their approach incorporates structural constraints inspired by [18] to evolve the BT. Unlike [3], mutations are not restricted to nodes of the same type, increasing diversity. Conditions are chosen by the Genetic Programming (GP) algorithm, with essential ones included at an atomic level. Several modifications have been made to the *original* approach to try to improve its performance:

- *Change Mutation Probs.:* Adjusted mutation probabilities to favor changes within nodes of the same type, focusing the likelihood of mutation towards similar nodes without deeply affecting diversification.
- *Allow Condition:* Allowed the last child node of a (sub)tree to be a condition, providing more flexibility.
- *Global Alignment (GA):* Implementation of a global alignment matrix computation using the Needleman-Wunsch algorithm [20]. This promotes crossover between the most similar parts of the BT by aligning parent BTs and selecting the two most resembling parts.
- *Custom GA:* Developed a custom global alignment approach utilizing a custom similarity matrix based on node proximity based on their type, their core function and their parametrization. By computing their distances, a similarity score was assigned to nodes, facilitating crossover between similar nodes.

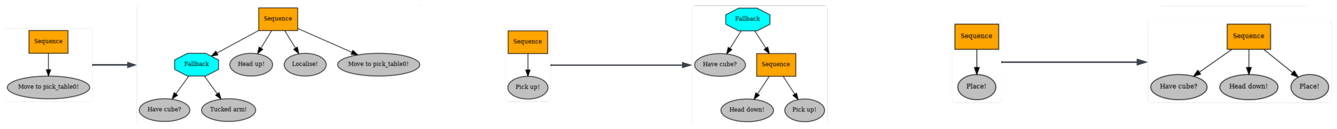

Fig. 2: Validation output for nodes *move_to_location/object* (left), *pick_object* (middle), and *place_object* (right).

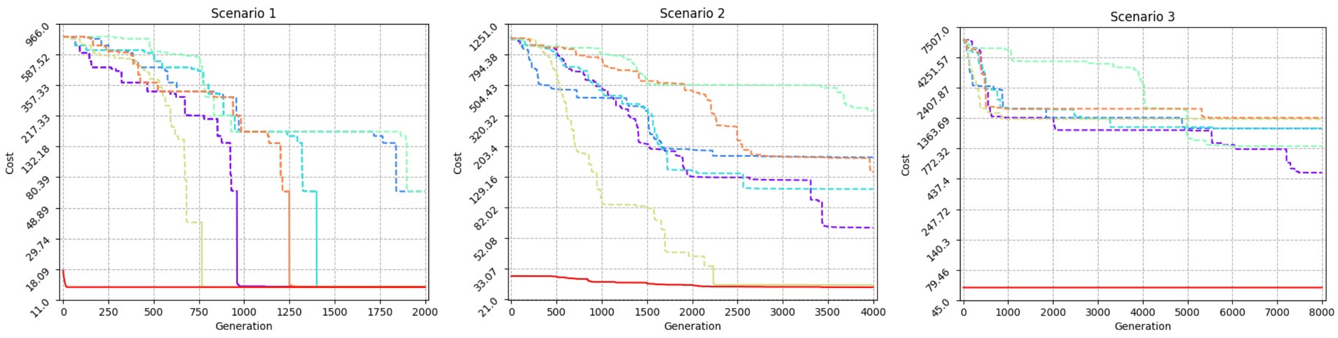

Fig. 3: Average results from *Scenarios 1-3* (lower cost implies better performance): *Change Mutation Probs.* (violet), *Global Alignment (GA)* (orange), *Custom GA and Change Probs.* (dark blue), *Custom GA* (blue), *Allow Condition* (light blue), *Original* (light green), and ***Ours*** **(red)**. Note that lines appear dashed when the task is not solved yet.

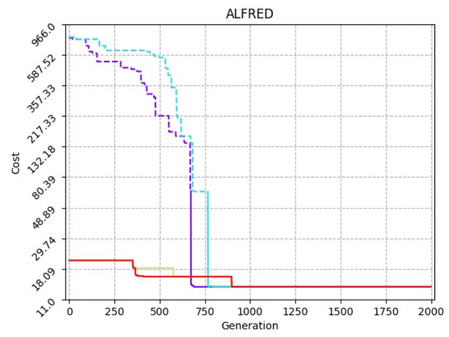

Fig. 4: Average results from different *ALFRED* tasks (lower cost implies better performance): *Original* (violet) and ***Ours*** **(red)**. Note that dashed lines represent BTs not solving the task, while solid lines represent BTs completing the task.

V. Discussion

The modifications to the GP algorithm did not yield any discernible improvements compared to the original methodology. Furthermore, a combination of various modifications, as illustrated in Figure 3, produced outcomes similar to those observed with the original approach. Notably, the modified algorithm failed to solve Scenario 2 within 4000 generations, a task successfully solved by the original approach.

We believe that the specific modifications of GA and changing mutation probabilities undergone to the original GP algorithm had a negative impact in the space exploration of possible BTs by reducing diversity in the population. On the other hand, allowing the last child of a sub tree to be a condition might result in larger trees without any effect on the environment, obtaining low fitness scores and inserting noise in the population. A future modification could be a guided search over the possible BTs interacting with an LLM assistant. Consequently, we retained the original GP algorithm for fine-tuning the assistant BTs.

Alternatively, the assistant pipeline generated BTs that effectively solved the proposed tasks from their initial generation. Subsequent post-processing was applied to refine the assistant BTs, aiming to create the most efficient, readable, and concise structures for the given tasks. Notably, as suggested in [11], our approach demonstrates superior performance. Figure 3 illustrates how the original approach failed to solve the Scenario 3 task within 8000 generations, while leveraging our LLM-based assistants enabled the generation of BTs that successfully solved the task since the first generation, hence using GP only for fine tuning the BT.

Additionally, we evaluated the original approach using simple ALFRED pick-and-place tasks, as depicted in Figure 4. Although these tasks are straightforward, our approach generated valid trees suitable for more realistic scenarios, assuming the agent possesses complete or generable knowledge of its surroundings. While further research is necessary to validate our approach for deployment in real-life robots and more complex tasks, our findings demonstrate the efficacy of integrating LLMs with subsequent post-processing techniques without the need of fine tuning it, resulting in the generation of valid and efficient BTs for simulated scenarios.

VI. Conclusion

This work introduces a complete BT generation pipeline combining LLM-based assistants and Genetic Programming (GP) without requiring LLM fine-tuning. We enable non-expert users to generating complex robot behaviors from narrated task descriptions. The combined approach outperforms BT-generation methodologies that just relay on GP. The potential versatility of this approach has been shown by extending the experimentation to a significant variety of simple instances of the ALFRED dataset.

Future research focuses on exploring automatic domain knowledge extraction, fitness function J generation , and testing the pipeline in stochastic environments, to enhance applicability and robustness in real-world scenarios.

References

[1] Y. Cao and C. S. G. Lee, "Robot Behavior-Tree-Based Task Generation with Large Language Models," Feb. 2023, arXiv:2302.12927 [cs]. [Online]. Available: http://arxiv.org/abs/2302.12927

[2] M. Colledanchise and P. Ogren, *Behavior Trees in Robotics and AI: An Introduction*. CRC Press, July 2018, arXiv:1709.00084 [cs]. [Online]. Available: http://arxiv.org/abs/1709.00084

[3] M. Colledanchise, R. Parasuraman, and P. Ögren, "Learning of Behavior Trees for Autonomous Agents," *IEEE Transactions on Games*, vol. 11, no. 2, pp. 183–189, June 2019, arXiv:1504.05811 [cs]. [Online]. Available: http://arxiv.org/abs/1504.05811

[4] Y. Gao, *et al.*, "Retrieval-Augmented Generation for Large Language Models: A Survey," Jan. 2024, arXiv:2312.10997 [cs]. [Online]. Available: http://arxiv.org/abs/2312.10997

[5] R. Ghzouli, T. Berger, E. B. Johnsen, A. Wasowski, and S. Dragule, "Behavior Trees and State Machines in Robotics Applications," *IEEE Transactions on Software Engineering*, vol. 49, no. 9, pp. 4243–4267, Sept. 2023, conference Name: IEEE Transactions on Software Engineering. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10106642

[6] Z. Hu, F. Lucchetti, C. Schlesinger, Y. Saxena, A. Freeman, S. Modak, A. Guha, and J. Biswas, "Deploying and Evaluating LLMs to Program Service Mobile Robots," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2853–2860, Mar. 2024, conference Name: IEEE Robotics and Automation Letters. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10416558

[7] X. Huang, W. Liu, X. Chen, X. Wang, H. Wang, D. Lian, Y. Wang, R. Tang, and E. Chen, "Understanding the planning of LLM agents: A survey," Feb. 2024, arXiv:2402.02716 [cs]. [Online]. Available: http://arxiv.org/abs/2402.02716

[8] M. Iovino, J. Förster, P. Falco, J. J. Chung, R. Siegwart, and C. Smith, "On the programming effort required to generate Behavior Trees and Finite State Machines for robotic applications," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 5807–5813. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10160972

[9] M. Iovino, J. Styrud, P. Falco, and C. Smith, "Learning Behavior Trees with Genetic Programming in Unpredictable Environments," Nov. 2020, arXiv:2011.03252 [cs]. [Online]. Available: http://arxiv.org/abs/2011.03252

[10] ——, "Learning Behavior Trees with Genetic Programming in Unpredictable Environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 4591–4597, iSSN: 2577-087X. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9562088

[11] S. Kambhampati, K. Valmeekam, L. Guan, K. Stechly, M. Verma, S. Bhambri, L. Saldyt, and A. Murthy, "LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks," Feb. 2024, arXiv:2402.01817 [cs]. [Online]. Available: http://arxiv.org/abs/2402.01817

[12] K. Kawaharazuka, T. Matsushima, A. Gambardella, J. Guo, C. Paxton, and A. Zeng, "Real-World Robot Applications of Foundation Models: A Review," Feb. 2024, arXiv:2402.05741 [cs]. [Online]. Available: http://arxiv.org/abs/2402.05741

[13] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3. Sendai, Japan: IEEE, 2004, pp. 2149–2154. [Online]. Available: http://ieeexplore.ieee.org/document/1389727/

[14] F. Li, X. Wang, B. Li, Y. Wu, Y. Wang, and X. Yi, "A Study on Training and Developing Large Language Models for Behavior Tree Generation," Jan. 2024, arXiv:2401.08089 [cs]. [Online]. Available: http://arxiv.org/abs/2401.08089

[15] A. Loganathan and N. S. Ahmad, "A systematic review on recent advances in autonomous mobile robot navigation," *Engineering Science and Technology, an International Journal*, vol. 40, p. 101343, Apr. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2215098623000204

[16] A. Lykov and D. Tsetserukou, "LLM-BRAIn: AI-driven Fast Generation of Robot Behaviour Tree based on Large Language Model," May 2023, arXiv:2305.19352 [cs]. [Online]. Available: http://arxiv.org/abs/2305.19352

[17] A. Mazumder, *et al.*, "Towards next generation digital twin in robotics: Trends, scopes, challenges, and future," *Heliyon*, vol. 9, no. 2, p. e13359, Feb. 2023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2405844023005662

[18] P. McClarron, R. Ollington, and I. Lewis, "Effect of Constraints on Evolving Behavior Trees for Game AI," in *CGAT 2016*, Mar. 2016.

[19] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao, "Large Language Models: A Survey," Feb. 2024, arXiv:2402.06196 [cs]. [Online]. Available: http://arxiv.org/abs/2402.06196

[20] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, Mar. 1970. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0022283670900574

[21] M. Nicolau, D. Perez-Liebana, M. O'Neill, and A. Brabazon, "Evolutionary Behavior Tree Approaches for Navigating Platform Games," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 9, no. 3, pp. 227–238, Sept. 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7435292/

[22] N. Nilsson, "Teleo-Reactive Programs for Agent Control," Dec. 1993, arXiv:cs/9401101. [Online]. Available: http://arxiv.org/abs/cs/9401101

[23] OpenAI, *et al.*, "GPT-4 Technical Report," Mar. 2024, arXiv:2303.08774 [cs]. [Online]. Available: http://arxiv.org/abs/2303.08774

[24] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks," Mar. 2020, arXiv:1912.01734 [cs]. [Online]. Available: http://arxiv.org/abs/1912.01734

[25] R. Tang, Y.-N. Chuang, and X. Hu, "The Science of Detecting LLM-Generated Texts," June 2023, arXiv:2303.07205 [cs]. [Online]. Available: http://arxiv.org/abs/2303.07205

[26] Z. Wang, W. Tongyu, and G. Hang, "A Survey: Development and Application of Behavior Trees," in *Communications, Signal Processing, and Systems*, June 2021, pp. 1581–1589.

[27] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," Jan. 2023, arXiv:2201.11903 [cs]. [Online]. Available: http://arxiv.org/abs/2201.11903

[28] J. Yu, *et al.*, "KoLA: Carefully Benchmarking World Knowledge of Large Language Models," July 2023, arXiv:2306.09296 [cs]. [Online]. Available: http://arxiv.org/abs/2306.09296

[29] Z. Zhao, W. S. Lee, and D. Hsu, "Large Language Models as Commonsense Knowledge for Large-Scale Task Planning," *Advances in Neural Information Processing Systems*, vol. 36, Feb. 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/65a39213d7d0e1eb5d192aa77e77eeb7-Abstract-Conference.html