LLM for Generating Simulation Inputs to Evaluate Path Planning Algorithms

Chenyang Wang, Jonathan Diller, Qi Han
Department of Computer Science, Colorado School of Mines
Golden, Colorado, USA
{chenyangwang, jdiller, qhan}@mines.edu

***Abstract*—In computer science and robotics research that focuses on algorithm designs, simulation is oftentimes the first step in validating the developed algorithms. However, simulation inputs need to be designed as close as possible to real-world scenarios so that a particular algorithm will perform equally well in simulation as in real-world testing. Designing credible simulation inputs is time-consuming and requires a fair amount of human labor, arguably due to the lack of an efficient way that streamlines the design process while having the capability to provide enough variations. In this study, we present the first-ever exploratory effort to use a Large Language Model (LLM) to facilitate generating simulation inputs. Specifically, we introduce two distributed Multi-Agent Path Finding (MAPF) algorithms and then utilize an LLM to generate variations of warehouse layouts to be used to validate our algorithms. We detail how to effectively prompt the LLM for simulated warehouse designs and compare algorithm performance on both human and LLM-created layouts. Our experimental results show that the LLM-generated layouts find the same algorithm performance trends as inputs designed by humans but require much less time to create, highlighting the LLM's potential to speed up simulation environment generation for algorithm testing.**

***Index Terms*—LLMs, Simulation Generation, Multi-Agent Path Finding, Distributed Robotics, Warehouse Design**

I. Introduction

In the rapidly evolving fields of Computer Science and Robotics, algorithm development and validation are critical steps in advancing technological capabilities. One fundamental step to validate these algorithms is through the use of simulations. These simulations offer a controlled environment where algorithms can be tested, refined, and validated before deploying in real-world scenarios. However, a significant challenge lies in designing simulation worlds (inputs) that are both representative of real-world scenarios and capable of highlighting the strengths and weaknesses of the algorithms.

The fidelity of a simulation directly impacts the confidence in an algorithm's performance during real-world testing. Hence, designing simulation inputs that closely mimic real-world conditions is paramount. This task, however, is often time-consuming and labor-intensive, requiring detailed knowledge of both the specific research domain and the practical aspects of simulation input design. Despite these challenges, efficient simulation input design remains a crucial step, as it can significantly streamline the algorithm development process and reduce the time required for validation.

Take the Multi-agent Path Finding (MAPF) algorithms for logistic robots working in warehouses as an example. Finding more efficient ways to operate a warehouse has been in constant demand since the beginning of the Industrial Revolution but has seen increasing interest given the advent of online shopping and advances in robotic technologies. Current research has proposed various MAPF algorithms for Automated Guided Vehicles (AGVs) working in warehouses, and the approaches can be categorized into two major types: centralized solutions [1]–[3] and distributed solutions [4], [5]. In either category, when testing their proposed algorithms, 2D grids are used as the simulation environment. We also observe that there is no universal design guideline when considering such a 2D layout for warehouses that can facilitate the testing of MAPF algorithms for multiple AGVs. In addition, when testing various MAPF algorithms in a warehouse setting, different warehouse layouts are used depending on various design specifications. Since every warehouse has its own specifications, creating an effective warehouse layout case by case by human designers is time-consuming, and the design cannot be guaranteed to support the full potential of multiple AGVs. On the other hand, we also notice that efforts are made to generate realistic 3D simulated warehouses [6]–[10]. While these simulation environments provide more realistic scenarios for multiple AGVs testing MAPF algorithms, the warehouse layout design process is more time-consuming and lacks flexibility when researchers want to test their algorithms on a large number of warehouse layouts and configurations. Therefore, we need to find a way to facilitate the simulation input design process.

In response, we believe that LLMs offer a unique opportunity to tackle this challenge. By leveraging their capabilities to process and analyze rich multimodal data, understand complex requirements, and generate solutions accordingly, LLMs can significantly contribute to the design of warehouses. Hence, utilizing LLMs in streamlining simulation input design to thoroughly test the performance of algorithms is a novel and promising area of exploration. Recognizing this need, this paper is motivated by an interesting question: **Can we enable LLMs to automate the design of warehouse layouts that can provide effective and flexible simulations for distributed MAPF algorithms?** Specifically, we first propose two distributed variations of [1], [2] where we find individual plans for each robot and then allow them to replan once stuck without requiring the robots to communicate with one another. We also compare these two algorithms against a distributed variation of [3] where traffic patterns avoid the need for collision avoidance. Then, by applying these MAPF algorithms to LLM-generated designs and those created by humans, we demonstrate the viability of this approach and its potential to revolutionize the simulation design process. Through this exploratory effort, we uncover new avenues for applying LLMs beyond their current uses. In particular, with the help of LLMs, the design process now only involves natural language without requiring professional programming skills and expert knowledge, and the parameters in the prompt can be changed easily according to various warehouse requirements, which automates the entire process. Following the aforementioned discussion, we make the following contributions to this study:

- We present and discuss two distributed MAPF algorithms for AGVs working in warehouses that do not require intra-robot communication.
- We present our approach to utilize LLMs to design simulated warehouses that then are used to test the performance of autonomous robots running distributed MAPF algorithms.
- We evaluate the performance of various distributed MAPF algorithms on warehouses designed by both humans and an LLM.
- We provide insights on the first-ever exploratory effort using LLMs to design simulation environments tailored for computer science or robotics algorithmic research.

II. Related Work

LLMs' Capabilities. With the growth of LLMs' capabilities, various studies have been conducted to test their limits for complex applications. For instance, ChatGPT is being used to generate English reading materials for middle school students in China [11], and the results show that the generated reading comprehension exercises can surpass the quality of human-written ones. ChatGPT is also being used to simulate believable human behavior as independent agents [12], and simulation shows that these generative agents can demonstrate observation, planning, and reflection abilities. Another study is done to investigate the ability to follow natural language instructions by using two LLMs: GPT-4 and PaLM2 [13]. LLMs are also being used to understand real-world raw sensor readings [14]. Results show that LLMs can achieve above 90% accuracy in understanding raw sensor readings from human activity sensing and heartbeat detection with proper guidance. In robotics research, LLMs are being used to provide suggested procedures for a specific task for robots to execute [15]. In addition, LLMs have also been proven useful in designing robots on both the conceptual and technical levels [16]. In short, LLMs have been proven to be powerful and effective in a wide range of real-world tasks.

Generation of Simulation Scenarios. Having a relatively efficient way to generate simulation environments is always an important aspect of research. With the rapid development of LLMs, they are specifically being used to generate simulation scenarios in various domains. To begin with, for robotics research, GPT-4 has been used to extend the task set from the previous tasks by generating codes for robotic arms to execute [17]. Specifically, GPT-4 is used to generate visuomotor policies for robotic arms by creating codes, and it successfully generated over 100 tasks from the original 10 tasks. Moving towards a more abstract level, LLM is being used to interact with digital twin simulations to determine feasible parameters to achieve a given objective [18]. LLM acts as multiple agents within the digital twin simulations to search for a feasible parameter setting for simulated physical processes. Combined with the concept of a knowledge graph, LLM is also being used to generate textual descriptions of simulated scenarios [19]. Several studies have also focused on generating simulation components for autonomous driving applications. For example, LLMs are used to generate synthesized pedestrian movements for realistic autonomous driving simulators [20]. LLMs can also edit photo-realistic 3D driving simulations by natural language commends [21]. However, all these efforts only considered using LLMs to generate some components for simulation environments without trying to enable LLMs to streamline the process of creating simulations themselves, which is the motivation of this study.

III. Background Problem: Distributed Multi-Agent Path Finding

To motivate the need for LLM-generated simulation inputs, we first introduce the MAPF problem and propose two distributed algorithms. Evaluating these algorithms via numerical simulation will require a large data set of warehouse designs.

A. Multi-Agent Path Finding Problem Definition

In the MAPF problem, we are given a warehouse layout with robot start locations, shelves, and goal locations, and we are tasked with finding collision-free paths through the warehouse to complete product retrieval tasks. A task consists of a starting location, a designated shelf where an object must be retrieved from, and a drop-off location. Tasks arrive randomly over the considered time horizon and are unknown a priori. We assume that the robots have complete state knowledge of the warehouse (i.e., the layout of the warehouse, the robot's location, and the location of all other robots) but do not know the goal or planned path of other robots and are not able to communicate with one another.

We assume we are given a set of n warehouse robots that can complete tasks. Due to the limited number of robots, we cannot service more than n tasks at once. If a new task arrives while all n robots are still completing earlier tasks, then we queue the task and wait for the next available robot. We consider the operation of the warehouse over a predefined time horizon T and study two variations of the MAPF problem:

Problem 1: **Multi-Agent Path Finding Problem with Maximum Completed Tasks (MAPF-MCT)**: Given a warehouse, n robots, and time horizon T, plan collision-free robot paths from the start point to the designated task shelf, and from the task shelf to the drop-off location such that the number of completed tasks is maximized.

Problem 2: **Multi-Agent Path Finding Problem with Minimum Time (MAPF-MT)**: Given a warehouse, n robots, and time horizon T, plan collision-free robot paths from the start point to the designated task shelf, and the task shelf to the drop-off location such that the average time to complete all tasks over T is minimized.

B. Algorithms

We evaluate two distributed algorithms for solving the MAPF-MCT and MAPF-MT problems. Both approaches are divided into two phases: an offline initial planning phase and an online adaptive phase. The algorithms both use the same first phase to find an initial plan but differ in the second phase.

In the first phase, we find a "collision-agnostic" optimal path, P, for the robot using A* search from the start to the task location, and from the task location to the drop-off location while ignoring all other robots. If a robot gets stuck while following P, the online adaptive phase kicks in. Below are our two proposed adaptive algorithms for avoiding robot-on-robot collisions.

1) Dynamic-Replanning: Our first collision avoidance heuristic is termed *Dynamic-Replanning* (*DR*). In this algorithm, we follow P until a collision occurs. With some probability, we attempt to plan a new route that will go around the obstacle that the robot collided with (in this case, a different robot). The algorithm for *DR* is given in listing 1.

Algorithm 1 Dynamic-Replanning

```
Require: $P$ : Given robot path
1: $waiting \leftarrow False$
2: while $P \neq \emptyset$ do
3:     $\alpha \leftarrow P.pop()$
4:     if performing $\alpha$ doesn't cause collision then
5:         Perform move $\alpha$
6:         $waiting \leftarrow False$
7:     else
8:         if $waiting$ then
9:             if $rand() \leq \epsilon$ then
10:                $P \leftarrow dynamicReplan()$
11:            end if
12:        else
13:            $waiting \leftarrow True$
14:        end if
15:    end if
16: end while
```

The DR algorithm takes movement command α from path P, which tells the robot to move in one of the cardinal directions. If performing the movement command will not lead to a collision with another robot, then the robot performs α. Otherwise, the robot waits a single time step and attempts α once more. After the second attempt, if performing the command would still lead to a collision, then the robot replans P using the *dynamicReplan()* function, with some probability ϵ. In the *dynamicReplan()* function, we first attempt to find a collision-free path from the robot's current location to the target location using A* search while considering cells that do not currently contain a robot. If we are unable to find a valid path, then we select a random location behind the robot and use a collision-agnostic A* search to find a route that first visits the random location behind the robot and then proceeds with the robot's assigned task.

2) Continuous-Replanning: The second heuristic-based algorithm that we consider is *Continuous-Replanning* (*CR*). In this approach, we re-plan the robot's route every few time steps in an attempt to prevent collisions before they happen. The algorithm for *CR* is given in listing 2.

Algorithm 2 Continuous-Replanning

```
Require: $P$ : Given robot path
1: while $P \neq \emptyset$ do
2:     if $i^{th}$ iteration then
3:         $P \leftarrow semiAgnstcReplan()$
4:     end if
5:     $\alpha \leftarrow P.pop()$
6:     if performing $\alpha$ doesn't cause collision then
7:         Perform move $\alpha$
8:     else
9:         Wait
10:    end if
11: end while
```

At the start of every i^{th} iteration, the robot discards the current tour P and dynamically re-plans P using the *semi-AgnstcReplan()* function. This function first attempts to find a collision-free route using A*, then uses an agnostic A* if no such collision-free route exists. The algorithm then observes the next move α in path P. If performing α will not cause a collision, then the robot performs move α, otherwise, the robot simply waits.

To evaluate the performance and trade-offs of these two algorithms, we require a large data set of simulation inputs. However, designing such inputs manually is time-consuming.

IV. Warehouse Design with Engineered Prompt for LLMs

In this section, we discuss the process of using an LLM to design a warehouse layout that can facilitate the testing of the aforementioned MAPF algorithms. Specifically, given its capability to understand textual and image inputs, we choose ChatGPT-4 [22] as the LLM.

A. The Prompt

ChatGPT needs a detailed description with some background to understand the task we ask it to complete. Therefore, providing as many details as possible is important to facilitate the design process. Inspired by Penetrative AI [14], we crafted a fixed prompt consisting of an Objective, multiple Expert Knowledge, multiple Design Specifications, and multiple Example Layouts. The overview of this LLM-based design process is shown in Figure 1: the Objective, Expert Knowledge, and Design Specifications are combined as a fixed prompt and given to ChatGPT, then ChatGPT's understanding of the task is combined with two examples to form another prompt, which leads to the final design generated by ChatGPT. We next provide details on each component.

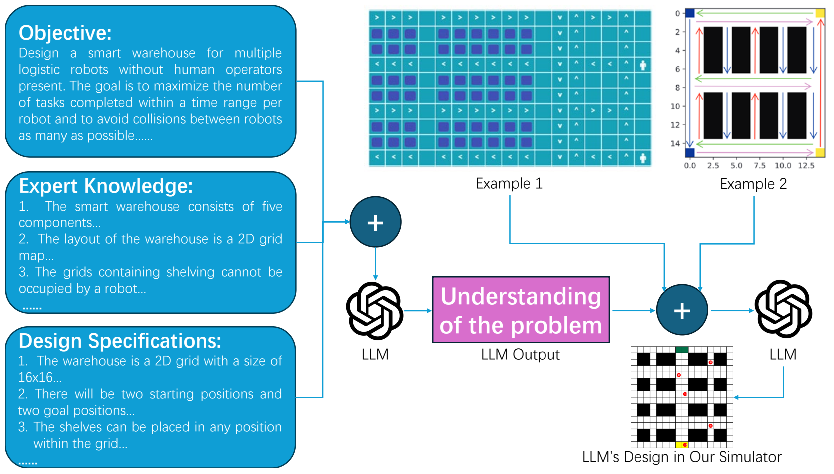

Objective:
Design a smart warehouse for multiple logistic robots without human operators present. The goal is to maximize the number of tasks completed within a time range per robot and to avoid collisions between robots as many as possible......
Expert Knowledge:
1. The smart warehouse consists of five components...
2. The layout of the warehouse is a 2D grid map...
3. The grids containing shelving cannot be occupied by a robot...
......
Design Specifications:
1. The warehouse is a 2D grid with a size of 16x16...
2. There will be two starting positions and two goal positions...
3. The shelves can be placed in any position within the grid...
......
Example 1
Example 2
LLM
Understanding of the problem
LLM Output
LLM
LLM's Design in Our Simulator

Fig. 1: The Overall Design Process using LLM

- *Objective:* The objective for ChatGPT is to design a 2D grid layout of the smart warehouse for multiple logistic robots without human operators present. We define the design goal as maximizing the number of tasks completed within a time duration per robot and avoiding collisions between robots as many as possible. To facilitate ChatGPT's understanding, we also provide an analogy of designing a city layout using one-way streets.
- *Expert Knowledge:* The expert knowledge provides definitions of different terms when designing the layout. For example, ChatGPT needs to know which components should be considered during the design, what kind of movements are allowed for the robots, and the specific definition of a robot's task. We also define constraints during the design, such as the path between shelves may only have the width of one grid.
- *Design Specifications:* By informing ChatGPT of the design specifications, the generated design will follow as close as possible to the user's request. In this study, we asked ChatGPT to generate a 16x16 2D grid layout with 80 shelves. However, these parameters can be any number according to a specific user's scenario. In addition, we enforced ChatGPT to generate a design that satisfies the following conditions: there must be a path from every starting position to every shelf and from every shelf to every goal position. The final design is in a CSV file.
- *Examples:* After ChatGPT understands the problem by considering the three aforementioned components, we supply it with two examples of 2D grid layouts of a warehouse designed by humans. The first example is from a previous work [3]. In each example, the shelf placement and traffic patterns are clearly indicated. The purpose of providing these examples is to further enhance ChatGPT's understanding of the problem visually in addition to the textual inputs.

B. Insights

We obtained several insights from this process. 1) For Expert Knowledge and Design Specifications, ChatGPT can hardly infer the underlying meaning of some constraints and specifications, which, on the other hand, a human can easily infer. In addition, ChatGPT makes its own assumptions about some definitions and constraints, which do not align with the design specifications. Therefore, after several iterations with ChatGPT trying to understand its limit, we added several very detailed instructions in the final prompt to ensure ChatGPT can follow the prompt. As a result, in the final prompt, Expert Knowledge has 17 very detailed items, and Design Specifications has eight very detailed items. 2) Even though the prompt contains detailed instructions, ChatGPT can still make minor mistakes in the final design. Therefore, it is necessary to check the generated design and fix these obvious errors manually. 3) Providing the visual examples is beneficial. Given the image-reading and logical reasoning capabilities of ChatGPT, providing images of warehouse layouts with traffic patterns designed by humans can improve ChatGPT's understanding of the rationale behind a specific design and apply it to its own design. This can be proven by ChatGPT's response to the aforementioned examples:

- Aisles that allow for two-way traffic, with main thoroughfares for the robots' primary routes.
- Secondary paths between the shelves where robots can move in one direction to reach specific areas.
- A looped system ensuring robots can move continuously without the need for U-turns, which increases efficiency and reduces the change of bottlenecks or collisions.

Next, we asked two engineering students with no prior knowledge of the study to complete the same task by following the prompt provided to the LLM. Specifically, we asked the students and the LLM for three types of warehouses: 16×16 in size with 80 shelves, 20×20 with 100 shelves, and 30×30 with 300 shelves. Due to space limits, we only show the visualizations of the first two designs, which are shown in Figure 2 and 3. All the designs will be evaluated in the following section.

V. Performance Evaluation

To evaluate the effectiveness of the LLM-designed warehouses as simulation inputs for validating our MAPF algorithms, we ran our algorithms with two baselines on both the human and LLM designs to determine if we see the same algorithm performance trends across the various designs. Specifically, all the designs are stored in CSV files, which are used as inputs to our simulation environment. This section summarizes our results.

A. Experiment Setup

We simulated 500-time steps and set a 25% chance that a new task would be generated at each time step. Each task consisted of a starting point, a target location, and a goal location. We ran the simulation 50 times for each warehouse design, starting each run with a unique random seed that was applied to each approach to ensure that all algorithms received the same tasking load. We consider the average number of tasks completed each run (the MAPF-MCT problem), the average number of time steps required per task (the MAPF-MT problem), and the longest that a robot had to wait as our experiment metrics.

In addition to our two proposed algorithms from Section III-B, we also experimented with two baseline methods. Our first baseline uses traffic patterns assigned to the warehouse and uses the A* search algorithm to find a single path for each robot before deployment. We term this approach the Traffic Patterns (TP) approach [3]. Our second baseline removes robot-on-robot collisions in the simulation and follows the optimal path found using A*. Although this approach does not provide valid plans for the MAPF problem, it does provide a theoretical bound on performance, finding a lower bound on the average number of steps required per task (LB) and an upper bound on the number of tasks that can be completed in the considered time frame (UB).

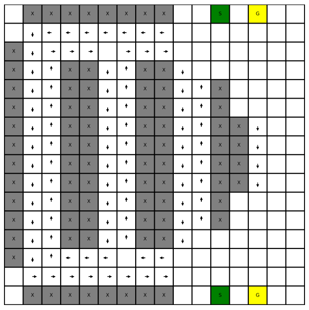
B. Simulation Results

Figures 4, 5, and 6 show the simulation results on the 16×16, 20×20, and 30×30 warehouse layouts, respectively. For clarity, a higher average task completion is better, while a lower time-steps per task and longest wait time are better.

What we are looking for in the graphs are repeating trends in algorithm performance between the human-designed inputs and the LLM inputs. Although the algorithms performed differently between the human and LLM design, the CR and DR algorithms perform comparably in Tasks Completed on all input sizes, with CR slightly better on human inputs than what is seen on the LLM. However, the two algorithms perform closely enough that we cannot claim one is better than the other in Tasks Completed. Both algorithms greatly outperform the TP baseline on both human and LLM inputs.

For the Time Steps Required metric, we see that the human inputs show the CR algorithm clearly outperforming DR but do not see this same trend on all LLM inputs. Notably, the LLM designed a 16×16 warehouse where the DR algorithm reduced the number of time step per task by 13.0% compared to the CR algorithm, breaking from the trend. However, both the human and LLM inputs show that the TP baseline performs comparable or better than our two proposed algorithms in Time Steps Required, leading to the same conclusion that the proposed algorithms trade-off total tasks complete with time steps required per task.

Both the human and LLM inputs show that the two proposed algorithms avoid deadlocks, while the TP algorithm can lead to deadlocks. This is seen in the Stuck Time metric, where the two proposed algorithms average near zero time steps without moving, while the TP algorithm clearly allows the robots to get stuck for both human and LLM designs.


(a) Human's design

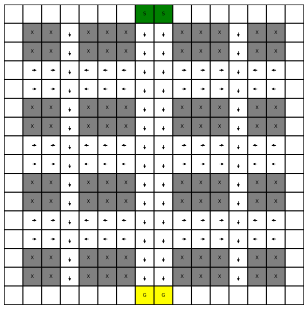
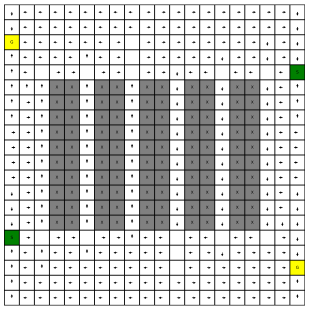

(b) LLM's design

Fig. 2: A 16×16 warehouse with 80 shelves. Grey cells are the shelves; white cells are the empty floor space with an omnidirectional traffic pattern; green cells are the starting positions; and yellow cells are the goal positions.


(a) Human's design

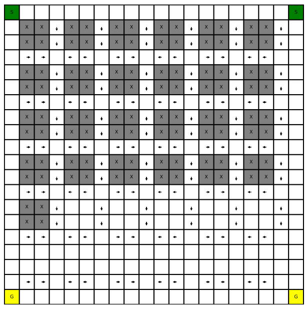

(b) LLM's design

Fig. 3: A 20×20 warehouse with 100 shelves.

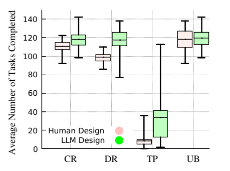
C. Experiment Takeaways

Our simulation results give us two major takeaways from this study. Firstly, we see that, for the most part, the human-designed inputs and the LLM-designed inputs lead to the same performance trends across the considered algorithms. This was particularly true for the Tasks Completed metric and the Stuck Time metric. For the Time Steps Required metric, we get conflicting performance trends between the CR and DR algorithms but see that both average more time steps per task when compared to the TP baseline.

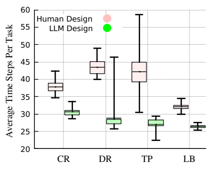
Secondly, our results show that both the human designers and the LLM struggled to design traffic patterns in the warehouse. Previous work has shown that traffic patterns in warehouses and the TP algorithm can perform well if the warehouse is designed correctly [3]. However, both the LLM and the human designer created warehouses that allowed for deadlock scenarios in the traffic patterns. This indicates that our prompts were unclear on how to add traffic patterns to the warehouse, and asking a human designer to follow our prompts shows us that the prompt itself needs to be addressed.

In terms of time consumed for the design process, the students reported that it took them roughly 50 minutes to design one variation. On the other hand, for each variation, LLM only took within a minute to produce. This strongly indicates the potential of our approach to speed up the validation process.

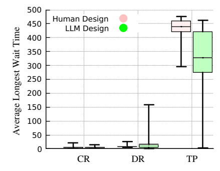
In summary, these results show that an LLM can be used to generate simulation inputs to evaluate algorithm performance, but we also recommend validating results found on LLM inputs with a small selection of human-designed inputs.



(a) Tasks Completed

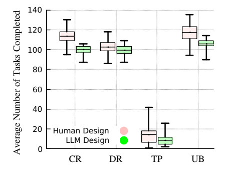


(b) Time Steps Required

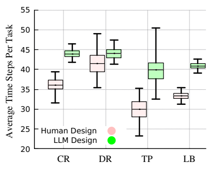


(c) Stuck Time

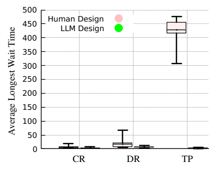
Fig. 4: Simulation results on 16×16 warehouse designs.



(a) Tasks Completed

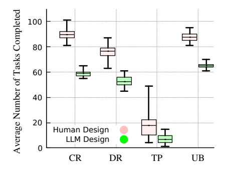


(b) Time Steps Required

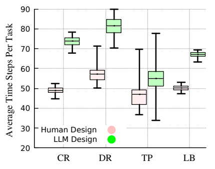


(c) Stuck Time

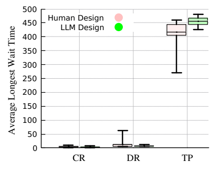
Fig. 5: Simulation results on 20×20 warehouse designs.



(a) Tasks Completed



(b) Time Steps Required



(c) Stuck Time

Fig. 6: Simulation results on 30×30 warehouse designs.

VI. Conclusion

This study explores the feasibility of utilizing currently available LLMs to facilitate the simulated warehouse layout design for MAPF algorithms testing. By changing the parameters in the prompt, we show that LLMs can design warehouse layouts on various scales for algorithm validation. By comparing the performance of MAPF algorithms for distributed robots on human-designed and LLM-designed layouts, we show that LLMs' designs can produce similar performance trends compared to human designs but with much less time consumed. However, we also found out that LLMs are not as creative as humans when designing the layout; in addition, LLMs occasionally create deadlocks. To enhance the creativity and the robustness of the generated layouts, future research could focus on specific prompt engineering to avoid deadlocks and provide more visual examples to increase LLM's creativity. Overall, considering both the benefits and the limitations, using LLM to generate simulation inputs for algorithm validation is promising.

References

[1] S. D. Han and J. Yu, "Effective heuristics for multi-robot path planning in warehouse environments," in *IEEE International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2019.

[2] ——, "Ddm: Fast near-optimal multi-robot path planning using diversified-path and optimal sub-problem solution database heuristics," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, 2020.

[3] A. Bolu and Ö. Korçak, "Path planning for multiple mobile robots in smart warehouse," in *IEEE International Conference on Control, Mechatronics and Automation (ICCMA)*, 2019.

[4] S. Dergachev and K. Yakovlev, "Distributed multi-agent navigation based on reciprocal collision avoidance and locally confined multi-agent path finding," in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2021.

[5] B. De Wilde, A. W. Ter Mors, and C. Witteveen, "Push and rotate: cooperative multi-agent path planning," in *ACM International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, 2013, pp. 87–94.

[6] M. Bučková, M. Krajčovič, and D. Plinta, "Use of dynamic simulation in warehouse designing," in *International Conference on Intelligent Systems in Production Engineering and Maintenance*, 2018.

[7] A simulation of basic robot localization (slam) and navigation in warehouse environment. [Online]. Available: https://github.com/wh200720041/warehouse_simulation_toolkit

[8] Aws robomaker small warehouse world. [Online]. Available: https://github.com/aws-robotics/aws-robomaker-small-warehouse-world

[9] Dynamic logistics warehouse. [Online]. Available: https://github.com/belal-ibrahim/dynamic_logistics_warehouse

[10] L1br: a warehouse robot simulation. [Online]. Available: https://github.com/LASER-Robotics/Warehouse_Gazebo

[11] C. Xiao, S. X. Xu, K. Zhang, Y. Wang, and L. Xia, "Evaluating reading comprehension exercises generated by llms: A showcase of chatgpt in education applications," in *18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, 2023.

[12] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *36th Annual ACM Symposium on User Interface Software and Technology*, 2023.

[13] J. Zhou, T. Lu, S. Mishra, S. Brahma, S. Basu, Y. Luan, D. Zhou, and L. Hou, "Instruction-following evaluation for large language models," *arXiv preprint arXiv:2311.07911*, 2023.

[14] H. Xu, L. Han, Q. Yang, M. Li, and M. Srivastava, "Penetrative ai: Making llms comprehend the physical world," in *25th International Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2024.

[15] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," in *Conference on robot learning (CoRL)*, 2023.

[16] F. Stella, C. Della Santina, and J. Hughes, "How can llms transform the robotic design process?" *Nature machine intelligence*, vol. 5, 2023.

[17] L. Wang, Y. Ling, Z. Yuan, M. Shridhar, C. Bao, Y. Qin, B. Wang, H. Xu, and X. Wang, "Gensim: Generating robotic simulation tasks via large language models," in *The International Conference on Learning Representations (ICLR)*, 2024.

[18] Y. Xia, D. Dittler, N. Jazdi, H. Chen, and M. Weyrich, "Llm experiments with simulation: Large language model multi-agent system for process simulation parametrization in digital twins," *arXiv preprint arXiv:2405.18092*, 2024.

[19] J. Zhang, Y. Zhang, M. Chu, S. Yang, and T. Zu, "A llm-based simulation scenario aided generation method," in *IEEE Information Technology and Mechatronics Engineering Conference (ITOEC)*, 2023.

[20] M. Ramesh and F. B. Flohr, "Walk-the-talk: Llm driven pedestrian motion generation," in *IEEE Intelligent Vehicles Symposium*, 2024.

[21] Y. Wei, Z. Wang, Y. Lu, C. Xu, C. Liu, H. Zhao, S. Chen, and Y. Wang, "Editable scene simulation for autonomous driving via collaborative llm-agents," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[22] OpenAI, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.