The National Higher School of Artificial Intelligence

Introduction to AI Course Project

Spring 2025

Introduction to AI Project Paper

Project Title: Warehouse Inventory Management Optimization

Student List:

Full Name	Group
LEFKAIER Yacine	1
GASMI Redhoua Ghezlene	6
MOUHOUN Cilia	6
AIT BELKACEM Ryme	6
MANAA Mohaned (Team Leader)	8
LASSEL Asma	12

Abstract

This project portrays the problem of warehouse optimization, suggests an approach to solve it through search algorithms with their different types; local and general, and applies some aspects of constraints satisfaction implicitly within the code. The main objectives are minimizing path finding time for items placement and retrieval by the agent, finding the best placement of the item according to several constraints, and solves one of the crucial issues of real-life

warehouses which is reordering problem, showing in this way modular solutions by dividing the big problem into three sub-problems (path finding, best placement, layout optimization) and solving them separately then combining them together. This allows simplifying some of the huge complexity that warehouses problems own.

The system we built, **AtlasWMS**, is as relevant as possible to real life warehouses, by accounting each of its particularities and constraints.

Keywords: Warehouse optimization, Search Algorithms, Constraints Satisfaction, layout optimization, placement and retrieval path finding.

Table of contents

1. Introduction:

The problem of warehouse optimization is an issue that is commonly known and extensively defined in the context of decision-making problems. The classical issue of warehouse layout optimization and travel time minimization is defined in literature as the "**Storage Location Assignment Problem**" that is classified as an **NP-HARD** problem, meaning there is no known polynomial time search algorithm that can solve this problem.

Warehouse optimization is key to the efficient operation of warehouses of all sizes. It is about defining a reliable automation system that saves time of placement and retrieval, ensures an efficient storage space for the better exploiting of resources, while reducing errors and improving flexibility, and reduce humans' intervention, as the system fully relies on the agents and their assigned tasks. It takes into consideration the flow of tasks, ordered and managed with precision by the system, the products placement and the retrieval, and the periodic reordering to ensure the maintenance of storage efficiency overtime.

Warehouse optimization is one of the impactful factors of the company's success, and the redirection toward the automation is a step ahead to formalize the procedures and rules of the company as it ensures the least possible errors and improving the overall financial performance.

The approach to solve this problem was by following the **divide and conquer** methodology as a modular solution, warehouse optimization and problem definition was divided into three sub-problems:

Path finding: Either the task is a placement or a retrieval or when the agent is moving around the warehouse, search algorithms are being used to find the optimal path.

Product Placement: The agent will run search algorithms that optimize the selection of the best insertion slot in the warehouse respecting many constraints in the process.

Reordering problem: The layout of the warehouse will get messed up overtime after several operations on the racks done by the agent, so the algorithms are run to find a best new layout that minimizes storage inefficiency in similar aspects with the product placement problem.

2. State of the Art

The **Storage Location Assignment Problem (SLAP)** is fundamental problem for warehouse efficiency, and it got researchers working on it for decades.

Key Approaches

1. **Basic Storage Policies**
    a. *Random Storage*: Simple but inefficient for correlated picks Fumi, A., Giordano, F., & Schiraldi, M. (2010).[1]
    b. *Dedicated Storage*: Fixed locations based on turnover (Kofler, M., Beham, A., Wagner, S., & Affenzeller, M. (2014).) [2]
    c. *Class-Based Storage*: ABC zoning (Hausman et al., 1976)[3]
2. **Demand-Aware Methods**:
    a. *Demand Correlation Patterns (DCP)*: Zhang & Wang (2019) reduced travel distance by 22% by co-locating frequently ordered items [4]
    b. *Order-Oriented Slotting*: Mantel, R.J., Schuur, P.C. and Heragu, S.S. (2007) optimized for predefined order sets[5]
3. **Advanced Techniques**:
    a. *Simheuristics*: Leon et al. (2023) combined simulation with metaheuristics for realistic constraints [6]

---

[1] Fumi, A., Giordano, F., & Schiraldi, M. (2010). *Storage Location Assignment Problem: Implementation in a Warehouse Management System*. Retrieved from https://art.torvergata.it/retrieve/handle/2108/53970/69005/DRAFT%20-%20Battista%20Fumi%20Giordano%20Schiraldi%20-%20SLAP%20Tool.pdfTor Vergata Art

[2] Kofler, M., Beham, A., Wagner, S., & Affenzeller, M. (2014). *Affinity Based Slotting in Warehouses with Dynamic Order Patterns*. In *Advanced Methods and Applications in Computational Intelligence* (pp. 123–143). Springer. DOI: 10.1007/978-3-319-01436-4_7ResearchGate

[3] Hausman, Warren & Schwarz, Leroy & Graves, Stephen. (1976). Optimal Storage Assignment in Automatic Warehousing Systems. Management Science. 22. 629-638.
10.1287/mnsc.22.6.629.https://www.researchgate.net/publication/227444439_Optimal_Storage_Assignment_in_Automatic_Warehousing_Systems

[4] Zhang, R.Q.; Wang, M.; Pan, X. New model of the storage location assignment problem considering demand correlation pattern. *Comput. Ind. Eng.* **2019**, *129*, 210–219. [**Google Scholar**] [**CrossRef**]

[5] Mantel, R.J., Schuur, P.C. and Heragu, S.S. (2007) Order Oriented Slotting: A New Assignment Strategy for Warehouses. *European Journal of Industrial Engineering*, 1, 301-316.
https://www.researchgate.net/publication/5171307_Order_oriented_slotting_A_new_assignment_strategy_for_warehouses

[6] Leon, J. F., Li, Y., Peyman, M., Calvet, L., & Juan, A. A. (2023). A discrete-event simheuristic for solving a realistic storage location assignment problem. Mathematics, 11(7), 1577. https://doi.org/10.3390/math11071577

b. *Integrated Approaches*: [7] Bolaños-Zuñiga et al. (2023) jointly optimized storage and routing

Algorithmic Progress

Method	Advantages	Limitations
Genetic Algorithms	Global optimization	Slow convergence (>300s)
Simulated Annealing	Handles NP-hard problems	Parameter sensitivity
FlexSim Simulation	Real-world validation	High implementation cost

Simulated annealing has been proven to be a powerful tool for **SLAP**, particularly for its ability to handle **NP-hard problems**. **Muppani and Adil (2008)** [8] have shown SA's superiority over GA, by achieving 15-20% greater cost reductions in complex warehouse configurations.

Optimization Techniques:

Modern solutions address SLAP holistically:

Route Integration:

- Shetty et al. (2020) achieved 27% faster picking by combining:
    - Vehicle routing models
    - Discrete-event simulation

Dynamic Adaptation:

- Keung et al. (2024) used IoT-enabled robots with A* pathfinding [9]
- Xu & Ren (2021) incorporated real-time congestion avoidance[10]

Research Gaps:

1. **Practical Deployment**:
    a. Most simheuristics (Leon et al., 2023) require FlexSim expertise
    b. Excel-based solvers lack scalability (Montanari et al., 2021)
2. **Dynamic Demand**:

[7] Bolaños-Zuñiga, J.; Salazar-Aguilar, M.A.; Saucedo-Martínez, J.A. Solving Location Assignment and Order Picker-Routing Problems in Warehouse Management. *Axioms* **2023**, *12*, 711. https://doi.org/10.3390/axioms12070711

[8] Muppani, Venkata Reddy & Adil, Gajendra. (2008). Efficient formation of storage classes for warehouse storage location assignment: A simulated annealing approach. Omega. 36. 609-618. 10.1016/j.omega.2007.01.006. https://www.sciencedirect.com/science/article/abs/pii/S030504830700076X?via%3Dihub

[9] Wang, D.; Liu, Q.; Yang, J.; Huang, D. Research on Path Planning for Intelligent Mobile Robots Based on Improved A* Algorithm. *Symmetry* **2024**, *16*, 1311. https://doi.org/10.3390/sym16101311

[10] Jiang, Huiling & Li, Qing & Jiang, Yong & Shen, GengBiao & Sinnott, Richard & Tian, Chen & Xu, Mingwei. (2021). When machine learning meets congestion control: A survey and comparison. Computer Networks. 192. 108033. 10.1016/j.comnet.2021.108033.

    a. Static approaches (Guo et al., 2022) fail with volatile orders
3. **Computational Limits**:
    a. ILP methods (Chen et al., 2020) struggle beyond 10,000 SKUs

Our Contribution:

- **Lightweight Python implementation:** Highlights every aspect of the project.
- **Demand-triggered optimization:** By building a flexible system.
- **Use of search algorithms:** SA, GA, Greedy BFS, A*, Local Beam Seach, Steepest Ascent.

3. Data Set building:

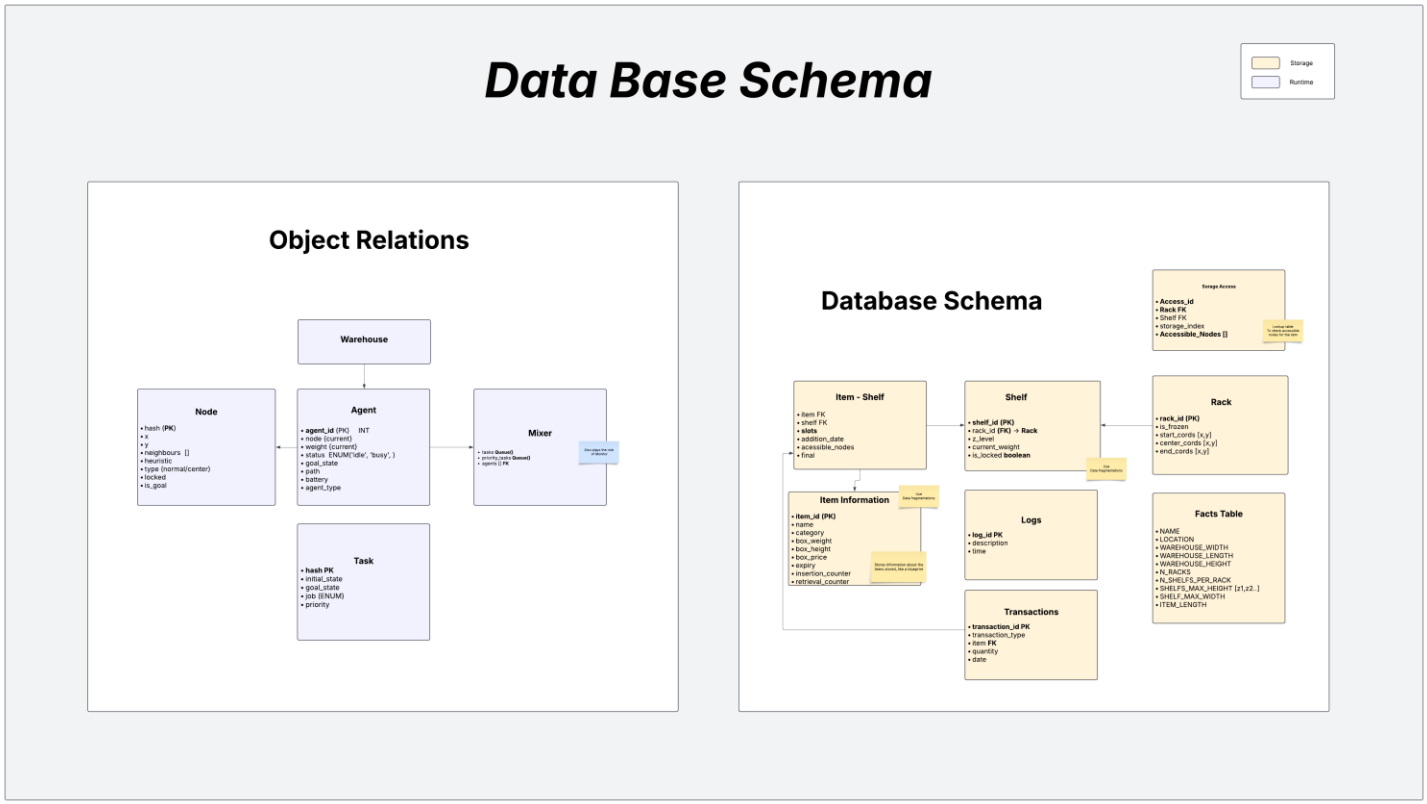
This project did not rely on actual datasets, rather the data it works on is completely user input. The system we built, AtlasWMS, specifically has a section called **inventory** where our user specifies the needed operation, whether a retrieval or a placement, then data is saved in the database, and specific transactions and logs are saved as well to make the **statistics** section be more accurate.

The database followed a dedicated study of real-life warehouses, encompassing the mixture of business rules and the automation requirements. The database design was inspired by the **Snowflake Schema** that is being used in **data warehouses**, given to the huge advantages it has, to name but a few: Storage efficiency, data integrity, and structured hierarchical representation.

The database schema has to main sections:

- Object Relations: As stated in the *figure* bellow, the objects in purple are runtime objects, thus they are not being saved in the database, they operate fully in runtime, and this allows more flexibility and dynamicity to the system. The dominating entity is the **warehouse**, that has many **Agents** who are the spine of our system, and the ones being assigned tasks. Each of the agents has a map of the warehouse that was modelized in **nodes** for the path, so he is aware of each node and its neighbors through what we called a **look-up table**, the latter facilitates the path finding problem enormously and was suggested by us to reduce the complexity. The agent can be assigned three types of **tasks**: **Retrieval**, **placement**, **reordering**. And the **mixer** is the orchestrator that ensures the smooth run of the system and the one assigning those tasks.

Database Schema: The database's core, has entities of racks that are composed of shelves, that in return contain the items that are being stored. It has the entities of transactions and logs to ensures safety and security of the system as everything is being registered to make it easy for backtracking actions and operations done in the database.



4. Problem Formulation:

As mentioned previously, the problem of warehouse optimization is didvided into three main problems to simplify its complexity:

4.1. Item insertion(placement)

When new products are brought to the warehouse the system issues a queue of tasks to the agents. To find the best placement of items respecting all the constraints from weight specifications heaviest items are prioritized to be stored in lower shelves, and lighter ones in top shelves, items compatibility as some items are preferred to be stored with ones to optimize storage efficiency and fasten the retrieval time later. And most importantly items accessibility and frequency, as some items are more accessed than others, thus they are prioritized in being stored in nearer shelves to the entrances, that are starting points for our agents.

Algorithms that are used are **Steepest Ascent** and **Simulated Annealing** to establish a comparison between them.

As for problem definition, both algorithms take the current layout of the warehouse as an argument, or an initial state precisely. Both operate to find the best spot where an item should be placed, they take one item at a time to reduce the complexity of the operation. The goal state here is clear, and reaching it must preserve all constraints, the resulting state is a new layout of the warehouse.

Objective function of Simulated Annealing: The function selects the best placement based on the score that is determined by the satisfaction of constraints, best is always chosen.

4.2. Path Finding:

When the agent is assigned a task, it must find the optimal path to the goal. In this regard two pathfinding algorithms were implemented A* and Greedy BFS. Both algorithms start from a starting point, assumed to be the entrance of the warehouse, if the task is for instance a retrieval it will fetch from the database the place of the item and will run to find the best path that takes to this goal state. Else, if it is about placement, the agent takes the output of the algorithm run to find the best placement for the item, then it runs to find the path to that goal. The environment is only affected in the aspect of the agent moving around the warehouse, the layout does not change.

Heuristic and Cost function of A*: Heuristic is the **straight-line distance.** The cost function is the sum of all Euclidean distances between each node and its neighbor.

Heuristic of Greedy BFS: The **Manhattan distance heuristic** estimates the cost from a node to the goal by summing the absolute differences of their x and y coordinates.

4.3. Reordering Problem (Layout Optimization)

The warehouse reordering is one of the complex problems. The **Genetic Algorithms**, one of the bio-inspired Algorithms that has been gaining popularity in the recent year for its effectiveness and robustness in finding successful solutions to them, specifically in this context **warehouse optimization**. It has two main advantages: fast intelligent search, used to find the best layout with nearly perfect placement of the item following many criteria and constraints of the respective problem, to name but a few: efficiency of storage that is relative to the accessibility of the item, compatibility of a category with other ones, weight constraints,...etc. Although GA does not guarantee an optimal solution in this regard, it is found often that the output provided is close enough to an optimal solution, and in practice the difference is not that significant. The second advantage that GA provides is that it does not require defining rules of a good solution; it is only based on intuitive knowledge (e.g. The previously mentioned constraints), which makes it fairly easy to formulate the problem, as in not having to determine the complex mathematical formula that englobes the problem, which is impossible in most times, and that makes it the perfect choice for such problems like warehouse reordering.

The problem is encoded using arrays called **chromosomes**, such that one chromosome is **one rack** of the warehouse, represented in an array of three sub-arrays being the three levels of the shelves. The **gene** was selected to be **one shelf of the rack**. The **tournament selection** was used to select the **best individuals** from the population since it is proven to be the most robust method. The **order crossover** is eventually going to be performed on shelves of the same level on one hand, as it shows better results in **minimizing the travel distance or time** to pick items in an efficient order. On the other hand, mutation is performed on random items of the shelf, swapping them between the racks, and **elitism** was used to select the outstanding resulting individuals.

The Genetic Algorithm takes as input the current layout of the warehouse, along with other parameters that defines some metrics for the operations that are later to be performed, and it returns as an output the best layout for a better reordered warehouse. The operation of reordering happens periodically to ensure storage efficiency.

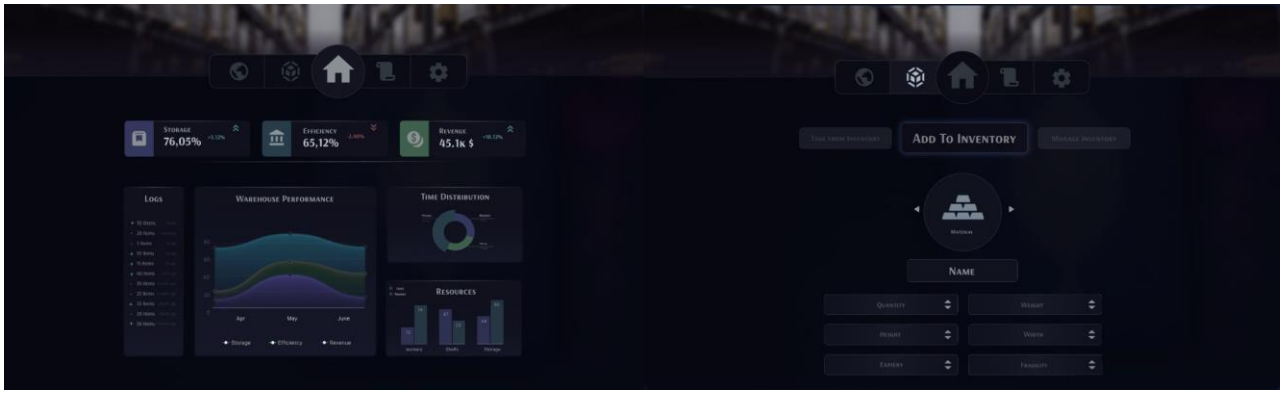
As for the comparative Algorithm, we have picked the **Local Beam Search Algorithm**.

CSP Model for layout optimization

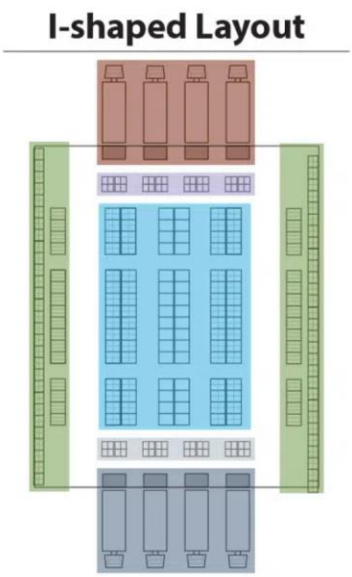
For the constraints' satisfaction, we have the Simulated Annealing Algorithm that was run with all the algorithms of the layout optimization (placement and reordering problems), since it takes into consideration all the possible constraints that were mentioned earlier, results are going to be shown in the Results and Analysis sections.

5. Application Design:

A website was designed to accommodate all the needs of the system built, while providing a user-friendly user interface. It had sections for the statistics, inventory sections were operations and transactions are managed efficiently. Here are some sections:



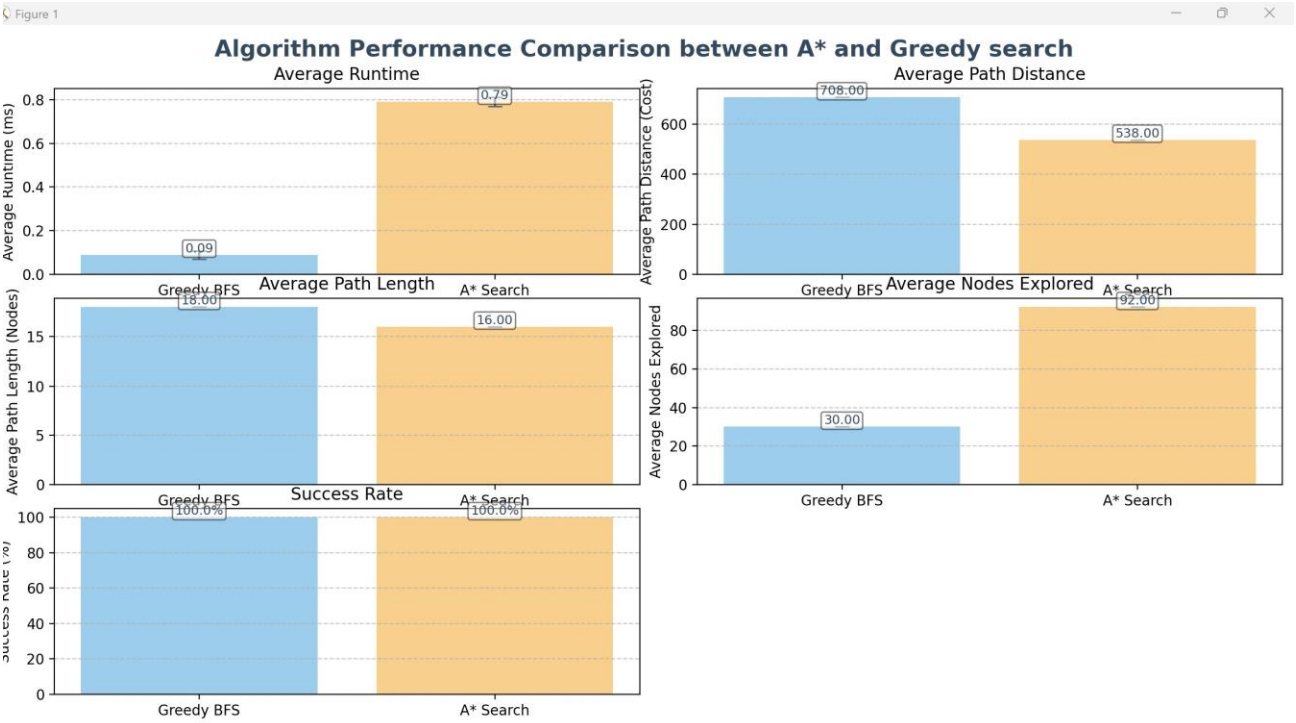
Further, we have built a 3D model to simulate the process of the tasks.



6. Results and Analysis:

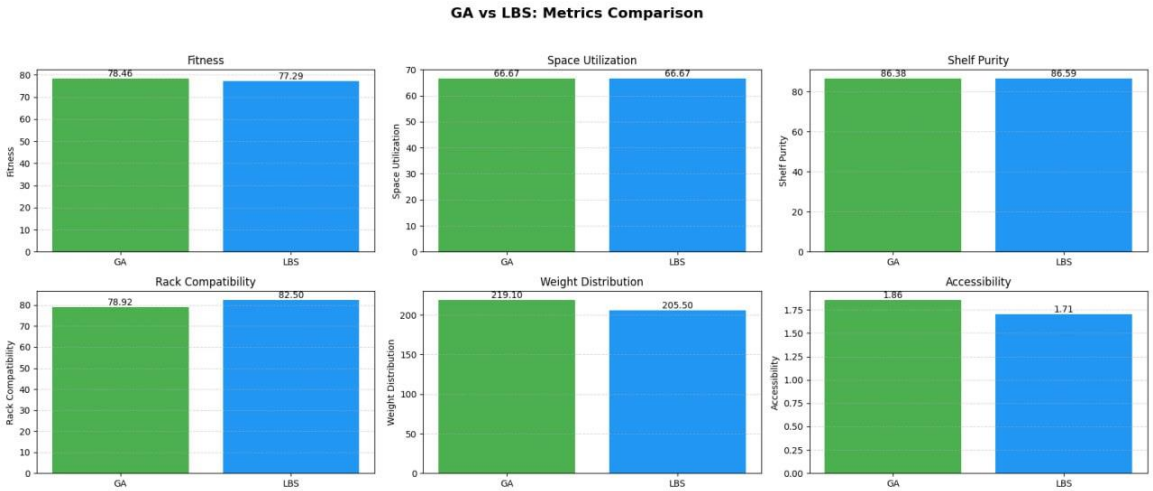
6.1. Greedy BFS vs A*

Greedy BFS has overall shown better results than A* in terms of path finding. It had better runtime and minimized the distance of finding the goal and it explored less nodes. However, A* and Greedy BFS were almost as equal when it came to success rate, since A* is a complete Algorithm, and Greedy BFS has demonstrated great results in path finding despite not being a complete algorithm.



6.2. Genetic Algorithms vs Local Beam Search

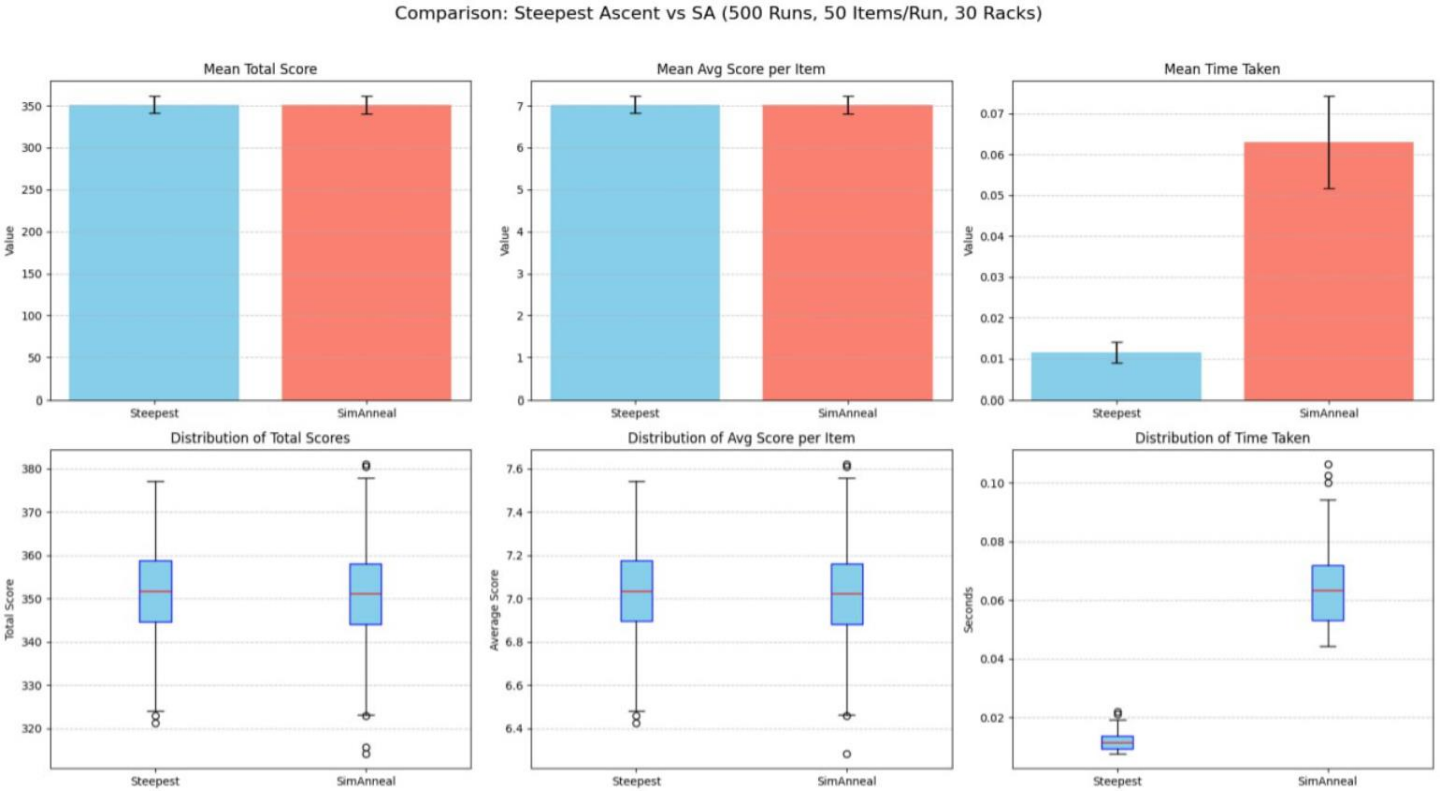
The Genetic Algorithm were run for 500 generations, the same with Local Beam Search, k chosen as 8 (k = 8), results show that they are almost similar in results, and they have shown that they are both suitable for layout optimization. Both algorithms respected the constraints and gave good results. The metrics of comparison as strict as they were, the layout of our warehouse was optimized significantly by running both algorithms.

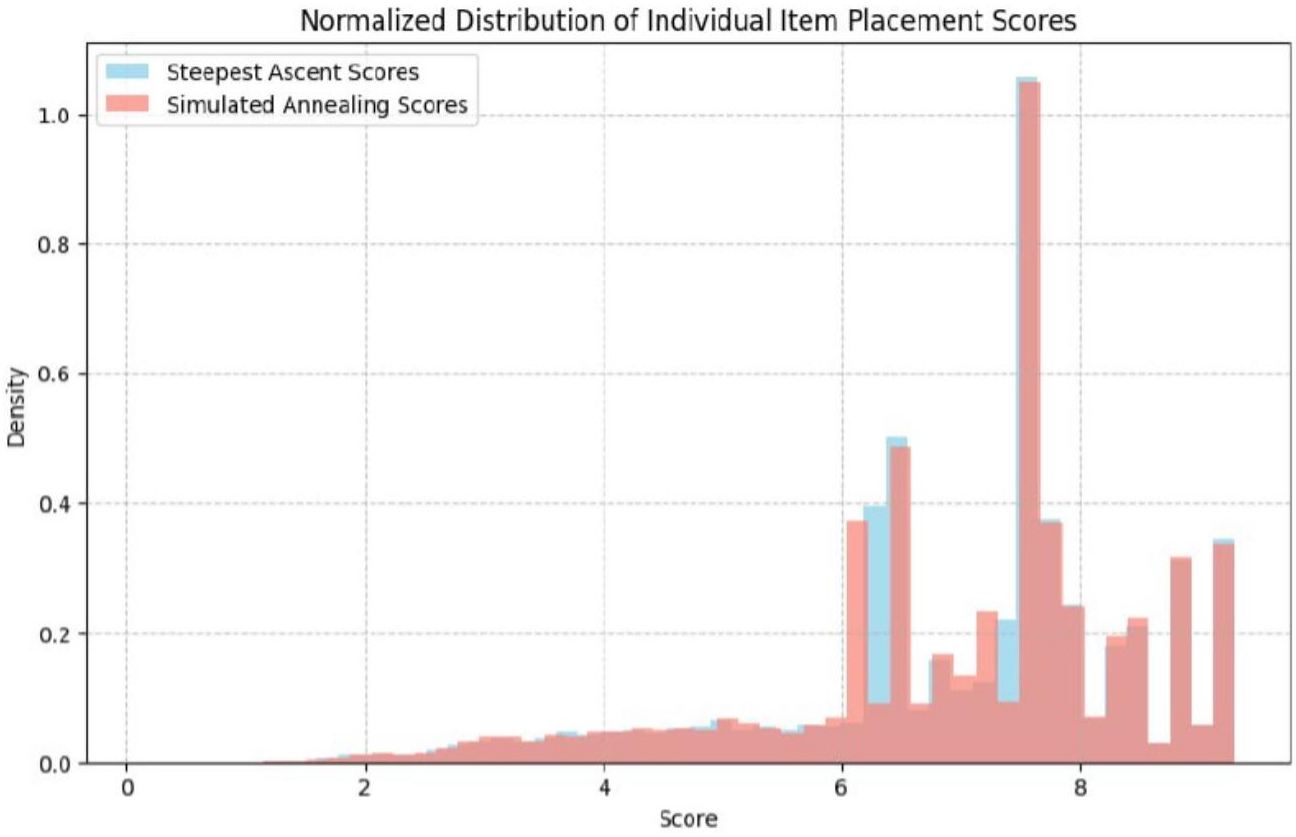


6.3. Simulated Annealing vs Steepest Ascent

The results that are shown below suggests that Steepest Ascent is better than Simulated Annealing in terms of item placement. Overall, the first one showed that is it more reliable, faster, and has more stable results.

Metric	Steepest Ascent	Simulated Annealing	Remarks
Mean Total Score	~355	~355	No significant difference
Mean Avg Score per Item	~7.1	~7.1	Equal per-item placement quality
Mean Time Taken	~0.01 seconds	~0.065 seconds	Steepest is much faster
Distribution of Total Scores	Tight, some outliers	Slightly wider, more outliers	Similar trend, SA more variable
Distribution of Avg Score per Item	Centered near 7.1	Centered near 7.1	Nearly identical
Distribution of Time Taken	Very tight	Higher variance, some outliers	Steepest is more stable
Overall Efficiency	High (fast & stable)	Lower (slower & variable)	Steepest preferred for speed





7. Discussion:

The results we had after running all these algorithms for the three problems were promising, the problem was approached in a way that reduced its complexity significantly.

Warehouses are indeed complex, resources must be preserved, and costs must be reduced, all with minimizing the retrieval time. We managed however to get good results, by trying our best to mimic the real warehouses.

However, portraying the actual behavior is still impossible, as this problem is NP-Hard, it is known that it has heavy computations, especially when all real-life constraints are needed to be satisfied. This is why we tried to select the most important ones that will make our simulation easier. And that was one of the limitations of our solution.

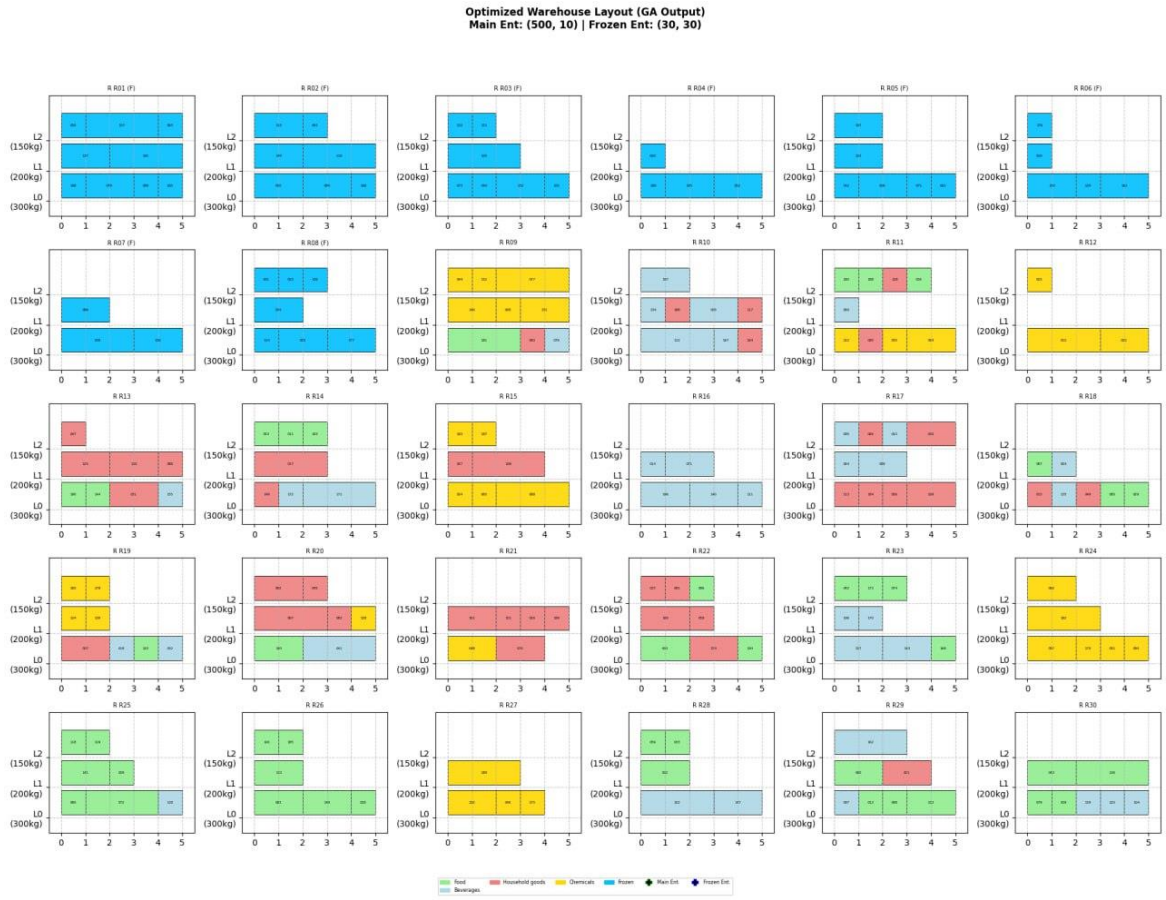
As for the possible improvements, we discussed that we would try more reinforcement learning techniques as they provide better approach for such problems.

8. Conclusion:

Search algorithms have shown impressive results for such a complex problem like warehouse optimization. It effectively improved travel time, items placement and warehouse layout. The move toward automation for warehouses now became a must, and it is clearly justified by this project. Automation with optimization does reduce humans intervening and it helps focusing on instructing more than doing, which saves the effort and errors committed in such sensitive environments that requires precision. The repeated experiments and their output confirm that digitalization is for the best, and it simplifies whatever complexities we face.

9. Appendix A:

An example of running Genetic Algorithm for the layout, it significantly optimized the scattered items and improved storage efficiency.



10. Appendix B: Tasks done by each member

Note: The algorithms were explained in depth and a better way in the notebook to respect the size of the report.

Yachine LEFKAIER	• Problem formulation • A* algorithm • 3D models • Implementation of lookup table • Testing and debugging
Ryme AIT BELKACEM	• Problem formulation (path finding) • 3D model (Blender) design and implementation • Database design and implementation • Greedy BFS search • Comparative analysis between A* and Greedy • Supervision • Testing and debugging • Result visualization Report writing : 3D section and comperative anlysis
GASMI Redhoua Ghezlene	• Database Schema • Problem formulation of the placement problem • State of the art section in the report • Simulated Annealing • Comparative analysis between Simulated Annealing and Steepest Ascent

	• Notebook building • Dataset collection and cleaning (CSV) • Testing and debugging
Cilia MOUHOUN	• Problem formulation • Frontend (the whole frontend using React, TailwindCSS, TypeScript, ShadCN library, and Motion library) • Notebook building • Map of the warehouse • A* algorithm • Collecting data (map.json)
Mohaned MANAA (team leader)	• Problem formulation • UI/UX • Database Schema • Flask Backend setup • Software setup • Steepest Ascent and Local Beam Search • Frontend and algorithms supervision
Asma LASSEL	• Reordering problem formulation • Database implementation (PostgreSQL) and design • Genetic Algorithm • Report writing (everything except state of the art) • Testing and debugging

The National Higher School of Artificial Intelligence
Introduction to Artificial Intelligence

Statement on Plagiarism and Academic Dishonesty

We, the undersigned, each declare having understood what plagiarism and academic dishonesty are, and we understand that we must use research conventions to quote and make exact references to other people's or even generative AI systems ideas and phrases/words in our reports and articles. We understand that plagiarism is an act of intellectual dishonesty. We understand that it is unethical and unacceptable to commit any of the following acts:

- submit an essay/report written in whole or in part by another student or other person as if it were my own;
- download an essay/report from the Internet and then take extracts or paraphrase them, in whole or in part, without precisely mentioning the original reference;
- reuse a sentence from another writer as is (without quotation marks and) without mentioning the source;
- paraphrase part of another author's work without reference to the source;
- reproduce the substance of another author's argument without reference to the source;
- take work originally done as an assignment or project given to a teacher and resubmit it to another teacher;
- cheating during an assessment or any other exam by using hidden notes or other unauthorized documents, by looking at another student's document, by passing my own answers to another student, by verbal or textual communication, by signs, or other means of storing and communicating information, including any electronic device, recording device, cell phone, headphones and/or laptop computer, etc.

We understand that the consequence of committing any of the aforementioned acts of academic dishonesty may include a zero on an assignment/project or exam, failure at the module, or even disciplinary action (see the Charter of Ethics and University Deontology of the Ministry of Higher Education and Scientific Research).

For this mini-project, we did not cheat or commit any of the aforementioned acts.

Last and first names: LASSEL Asma Date: 10/05/2025 Signature	Last and first names: Mohaned Manaa Date: 10/05/2025 Signature	Last and first names: GASMI Redhoua Rheglene Date: 10/05/2025 Signature
Last and first names: TEFKAIER Yacine Date: 10/05/2025 Signature	Last and first names: Mouhoun Cilia Date: 10/05/2025 Signature	Last and first names: AIT BELKACEM Ryne Date: 10/05/2025 Signature