Artificial Intelligence and Robotics (AIR)

Winter 2024-2025

Project Report

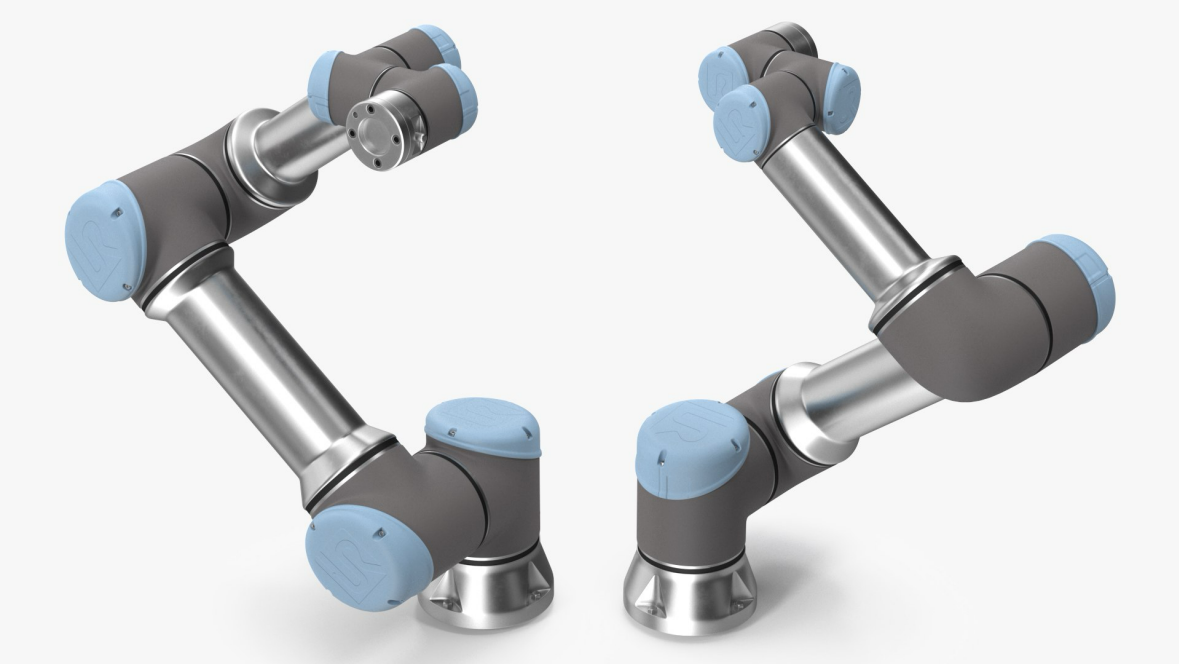
Working code:

Project Code

Submitted by:

Rula Younis 322855651

Kamil Bokaae 211877600



Content Table

PART A - Simulation

Task 1.1 – Stack Function

This task required implementing a function that creates stable cube stacks at a specified location. The function Stack receives two inputs:

- A list of initial cube positions
- A target location for stacking

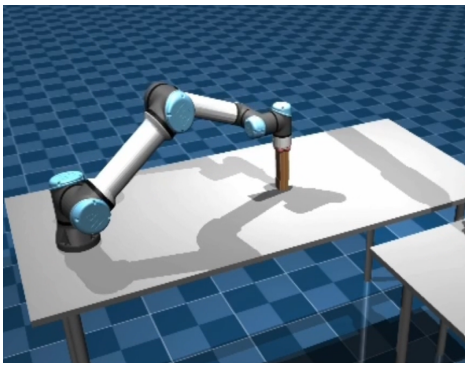
The result is a stable and neatly aligned vertical stack of cubes placed precisely at the target location. A video recording demonstrating the execution is provided as **stackPartA.mp4** in the Videos/partA directory.

Task 1.2 – Transfer Function

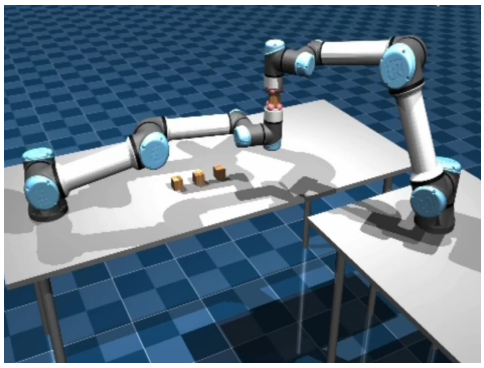
This task involved implementing a function that simulates the transfer of a cube between two robots at a shared handover point. The function TransferCubeBetweenRobots receives two inputs:

- A start position for the cube
- A handover location where the transfer takes place

The process begins with one robot picking up the cube from the given start position, moving to the handover point, and orienting its gripper to face the second robot. The second robot then approaches the same handover position and prepares for the transfer. This coordination simulates a collaborative exchange between two robotic arms.
A video recording demonstrating this behavior is provided as **transferPartA.mp4** in the Videos/partA directory.



Stack



Transfer

PART B – Lab Implementation

Task 2.1 – Stack Function

This task required adapting the stacking logic from simulation to the physical robots in the lab. The Stack function receives two inputs:

- A list of real-world cube positions (in meters)
- A target position on the table for stacking

Using the robot identified as UR5e_1, the function performs a sequence of pick-and-place actions to create a vertical stack at the specified target location. Each cube is picked from its initial location and carefully placed to maintain stability and alignment. Acceleration and speed parameters were tuned for precise control, and the robot was returned to its home position after completing the task.
A video recording demonstrating the lab execution is provided as **stackLab.mp4** in the Videos/partB directory.

Task 2.2 – Transfer Function

This task involved implementing the cube transfer operation between two physical robots in the lab. The TransferCubeBetweenRobots function takes two inputs:

- The start position of the cube
- The handover location where the transfer occurs

The process begins with UR5e_1 picking up the cube from the start position and moving to a predefined joint configuration suitable for handover. Simultaneously, UR5e_2 moves to its own handover posture. The TCP pose and joint angles of both robots are monitored to ensure proper alignment during the interaction. This setup enables a safe and coordinated robot-to-robot handover.
A video recording of the task being performed in the lab is available as **transferLab.mp4** in the Videos/partB directory.

PART C

Task description and motivation

This project involves programming a UR5e robotic arm to perform a structured manipulation task: picking up a set of blocks from predefined positions and placing them on a surface to spell out the word "AI".
The task is carried out within the MuJoCo simulation environment, using a custom setup developed in the CLAIR lab's sim_ur5 framework.

The choice of this task is motivated by its symbolic and practical significance - spelling out "AI" showcases how a robotic system can convert a high-level symbolic goal into a series of precise physical actions. It serves as a compelling example of how motion planning, spatial reasoning, and object manipulation can be integrated to achieve a meaningful arrangement.

To explore different strategies for assigning start and target positions, the project implements and compares four heuristics: a greedy nearest-neighbor approach, the Hungarian algorithm for optimal assignment, a Euclidean distance-based sorter, and a randomized matcher. This multi-heuristic setup allows for an in-depth analysis of the trade-offs between computational efficiency and path optimality, enriching the experiment's educational and practical value.

Ultimately, this project serves as a foundational platform for robotic control and symbolic task execution, while also setting the stage for future extensions involving multi-agent collaboration, object-based communication, and higher-level cognitive planning.

Planning Strategy

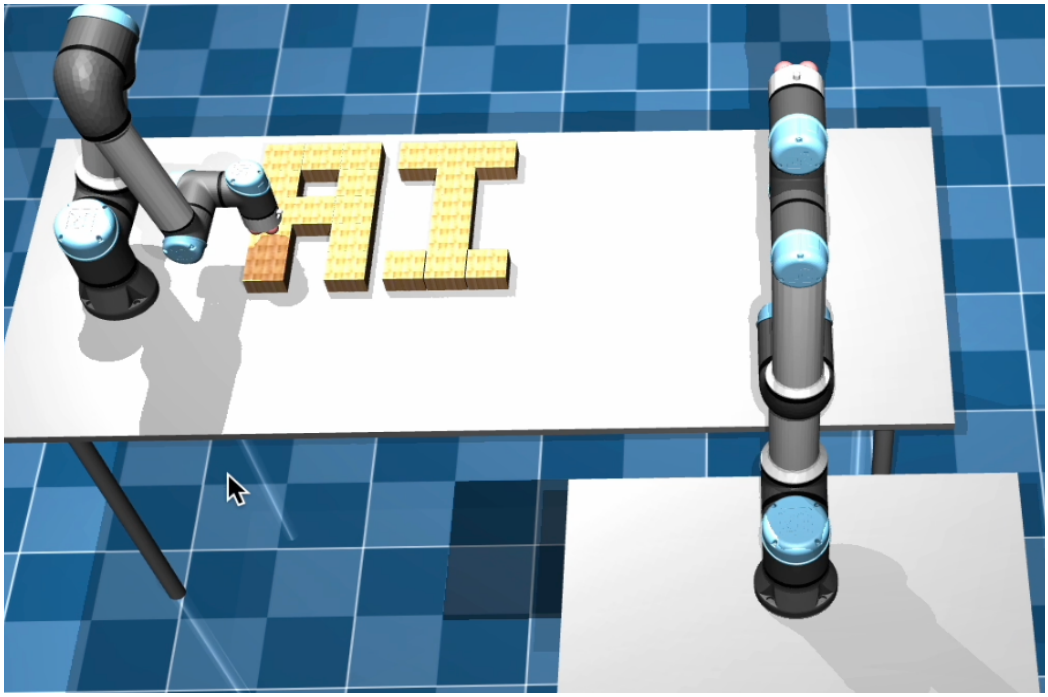
The planning process combined symbolic reasoning with geometric feasibility to accomplish the high-level objective of spelling out "AI" through precise block placements. Each letter was represented by a predefined set of target coordinates selected to form a clear and recognizable shape while remaining within the reachable workspace of the UR5e robot. These target positions were arranged with consistent spacing to account for the physical size of the blocks and to avoid overlaps.

To map start positions to target locations, we implemented and compared four different assignment strategies:

- **Greedy Matching**: Each block was paired with its nearest unassigned target, minimizing local distances in an iterative fashion.
- **Hungarian Algorithm**: A globally optimal assignment approach that minimized the total distance across all pairings.
- **Euclidean Sorted Matching**: Blocks were matched based on a sorted list of pairwise distances, favoring close assignments while ensuring uniqueness.
- **Random Matching**: A baseline method where target assignments were randomized to test the importance of heuristic design.

These strategies allowed us to study the trade-offs between computational complexity and total movement cost. Once assignments were computed, the robot executed a sequence of pick-and-place operations accordingly.

Motion execution was handled via high-level task-space commands, abstracting away joint-space planning. Each pick-up and placement action internally relied on inverse kinematics, allowing us to focus on spatial planning and heuristic evaluation. Minor adjustments were included to accommodate stacking and collision avoidance, ensuring smooth and safe block manipulation.



Implementation strategy

The project was implemented in the MuJoCo simulation environment using the UR5e robotic arm setup provided by the CLAIR lab framework. The scene was initialized with 21 blocks arranged in a fixed grid of predefined source positions. A separate set of 21 target coordinates was manually selected to form a clear visual layout of the letters "A" and "I".

The implementation, written in Python, uses the SimEnv and MotionExecutor modules to control the robot. Four heuristics: Greedy, Hungarian, Euclidean, and Random were used to assign each block to a target. For every assignment, the robot performed a pick-and-place action using high-level motion commands with internal inverse kinematics.

Simulation runs were recorded to visualize and compare the performance of each strategy.

Challenges and Constraints

The simulation environment supported a maximum of 22 blocks, which meant we needed to use larger blocks to represent the letters "A" and "I" without compromising their readability. This required thoughtful layout adjustments to make the best use of the available space.

During placement, we observed occasional slight misalignments between blocks. These were likely due to minor inaccuracies in the robot's control or actuation. To avoid blocks interfering with each other, we introduced small gaps between them in the target configuration. This simple adjustment helped maintain the stability and clarity of the final structure.

The robot's motion was also constrained by the size and layout of the table, as well as by the physical limitations of its joints and angles. In some cases, target positions that were too far from the base of the robot couldn't be reliably reached. To ensure all motions were feasible, we had to carefully place both the starting grid and the final layout within the robot's reachable workspace.

Due to overlapping between some of the start and target positions, we could not spread all the starting blocks flat on the surface. Instead, we stacked them to avoid conflicts and fit all blocks within the allowed area.

Algorithmic description

As mentioned earlier, four heuristic approaches were used to assign the 21 blocks to target positions: Greedy, Hungarian, Euclidean and Random. Each heuristic produced a unique block-to-target assignment based on its matching strategy.

Once the assignments were computed, the robot followed a consistent execution loop:

1. A pick-up motion is performed at the block's initial location.
2. The assigned target is retrieved from the computed assignment.
3. The block is moved and placed at the target with a slight vertical offset to ensure clearance.
4. The robot proceeds to the next block and repeats until all blocks are placed.

This algorithmic structure enabled consistent evaluation of the different heuristics under the same motion sequence.

Future Improvements

Several extensions could significantly enhance the system:

- **Obstacle Avoidance and Path Validation**: Introducing obstacles would create a more realistic environment and require the use of path planning algorithms (e.g., RRT*, A*) and reactive control strategies to ensure safe, collision-free trajectories.

- **Vision-Based Feedback**: Integrating a camera system would enable the robot to verify placements and adapt in real time.

- **Cooperative Manipulation**: Extending the setup to support two robotic arms would allow for more advanced tasks such as handovers and coordinated actions.

- **Randomized Initialization with Constraints**: Randomizing block starting positions would test the flexibility of planning. However, given the limited reachable area, it's important to enforce spatial constraints and detect if a block is initialized inside the target area, adjusting the plan accordingly.

These enhancements were beyond the current scope due to time and resource constraints but can be naturally integrated into the existing framework in future iterations.

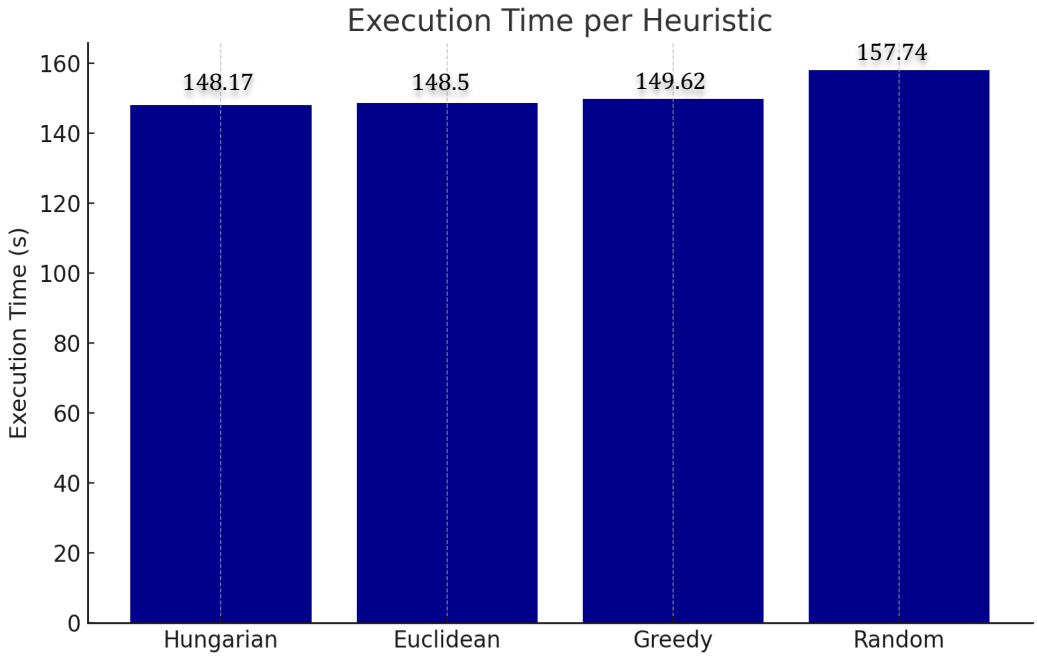
Performance analysis

To evaluate the effectiveness of the four heuristics, we collected data for two key metrics:

1. Execution Time (in seconds)

This measures how long the robot took to complete the full task — picking up all blocks in order and placing them at their assigned target positions. The only variable was the target selection heuristic.
The pickup order remained constant across all runs.

Execution Time per Heuristic:



This chart shows that:

- **Hungarian** was the fastest.
- **Euclidean** was slightly slower but still efficient.
- **Greedy** took more time due to local decisions.
- **Random** was the slowest, likely due to inefficient movements and higher path redundancy.

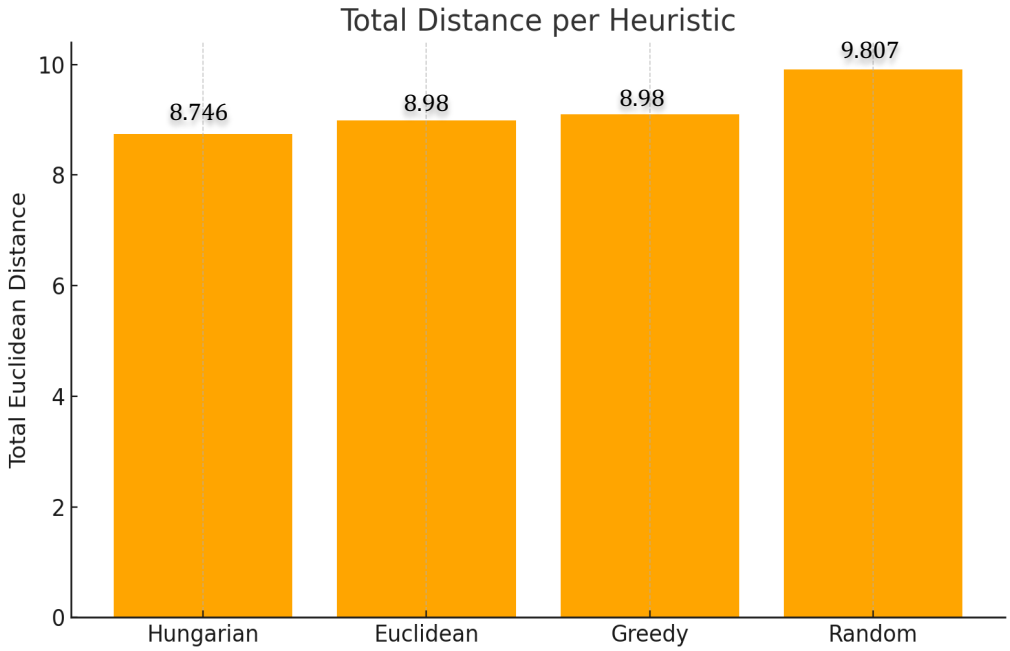
2. Total Euclidean Distance

This metric represents the sum of 2D Euclidean distances traveled by each block between its fixed start position and its target position, calculated as:

$$Distance = \sum_{i=1}^{n} \sqrt{(x_i^{start} - x_i^{target})^2 + (y_i^{start} - y_i^{target})^2}$$

Only the horizontal (x, y) plane is considered, as vertical movement (z) is constant across all heuristics.

Chart: *Total Distance per Heuristic*:

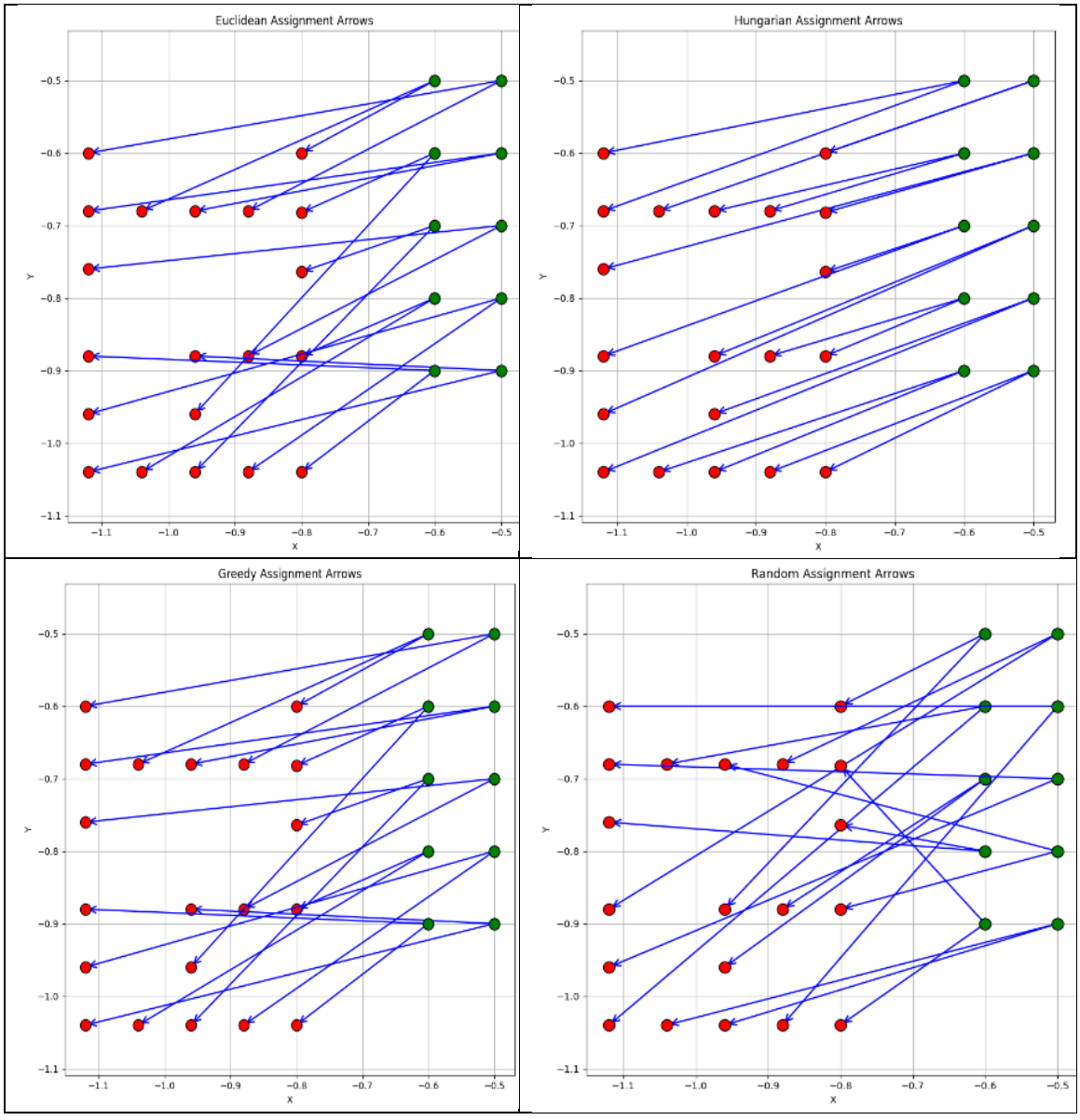


This chart shows that:

- **Hungarian** produced the shortest total distance (globally optimal).
- **Euclidean** and **Greedy** were slightly worse but still competitive.
- **Random** had the worst performance, with significantly more distance traveled.

Heuristic-Based Path Visualization

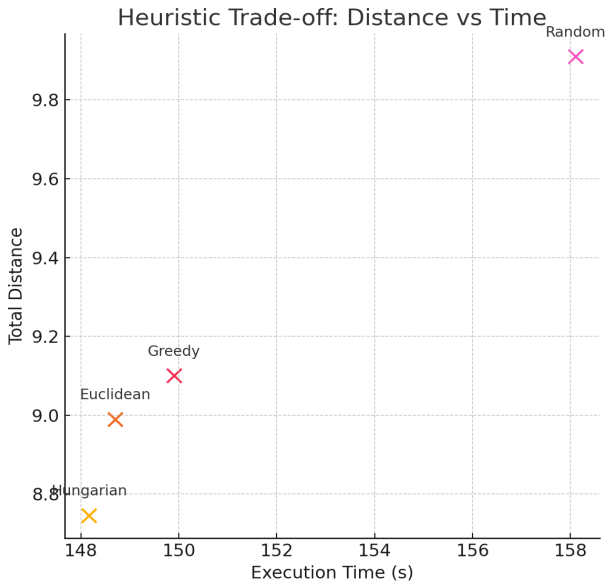
The following figures present a visual comparison of four different heuristics used to assign blocks from initial positions to target locations. Each arrow represents a planned robot movement from a start (green circle) to a target (red circle) position.



The key difference lies in how each heuristic determines the target location for each block. The Euclidean and Greedy methods prioritize proximity, selecting the nearest unassigned target at each step.
The Hungarian method optimizes the overall assignment globally, leading to more structured movement.

In contrast, the Random method assigns targets without spatial logic, resulting in disorganized paths and significantly less efficient block placements.

Performance Summary:



The chart above illustrates the trade-off between execution time and total distance for each heuristic. The Hungarian algorithm achieved the shortest path with relatively low execution time, while the Random heuristic performed the worst in both metrics.

Part C Videos:

1. Video recording of the Greedy strategy.
2. Video recording of the Hungarian strategy.
3. Video recording of the Euclidean strategy.
4. Video recording of the Random strategy.