

COMS4444 Project 2 Report

Team 6: Christian Daniel Valadez, Fenglei Gu, Chloe Nguyen

October 23, 2023

Contents

1	Introduction	4
1.1	Problem Specifications	4
1.1.1	Navigating the Flea Market	4
1.1.2	Balancing Digital Engagement and Environmental Awareness	5
1.1.3	The Challenge	5
1.1.4	Success Metric	6
2	Player Evolution	6
2.1	Initial Stage	6
2.1.1	Obstacle Avoidance in Initial Stage	6
2.1.2	Player Avoidance in Initial Stage	9
2.1.3	Navigation in Initial Stage	11
2.2	Intermediate Stage	11
2.2.1	Obstacle Avoidance in Intermediate Stage	11
2.2.2	Player Avoidance in Intermediate Stage	12
2.2.3	Navigation in Intermediate Stage	13
2.3	Final Stage	14
2.3.1	Unifying Obstacle Avoidance and Player Avoidance	14
2.3.2	A Star Search	15
2.3.3	Search Space: Choosing Search Breadth (Branching Factor)	17
2.3.4	Choosing Search Depth	18
2.3.5	Navigation in Final Stage	19
3	Tournament	20
3.1	Tournament and Ranking	20
3.2	Tournament Limitations	20
3.3	Tournament Analysis	21

4	Conclusion	29
5	Future Work	30
6	Summary of Contributions	30

1 Introduction

In today’s fast-paced world, the ubiquitous use of cellphones has revolutionized the way we communicate and access information. It has also created a unique challenge – the ability to stay engaged with our devices while navigating the physical world. In response to this dilemma, we embark on an intriguing journey in ”Cellphone Dodgem,” the second project of COMS-W4444 aimed at simulating the real-world cellphone-induced distractions. By recreating a bustling flea market scenario, we delve into the complexities of multitasking, human behavior, and navigation. The scope of this project is to develop a program that simulates the actions of players attempting to visit a list of stalls while trying to interact with their cellphones as much as possible. Our group, team 6: Christian Daniel Valadez, Fenglei Gu, Chloe Nguyen, will be defining the problem, walking through our strategies, and the efficiency of our solution.

1.1 Problem Specifications

1.1.1 Navigating the Flea Market

The setting is a flea market, spanning a 100-meter by 100-meter field, where stalls are scattered randomly. Each player is entrusted with a mission: move between these stalls to collect items in each stall. The list of stalls to visit and their respective coordinates is provided in advance, and players have the freedom to choose the order in which they visit them.

As they move from one stall to another, players are mostly engaging with their cellphone screens. This means that they remain blissfully unaware of any obstacles or other players in close proximity. With a constant walking speed of 1 meter per second, they can change direction freely, all while immersed in their digital world. However, if a player encounters an obstacle or another player within 0.5 meters, they halt abruptly, causing their cellphone to ”fall”, taking about 10 seconds for the player to recover and resume their journey.

1.1.2 Balancing Digital Engagement and Environmental Awareness

However, players are not entirely helpless; they can choose to momentarily look up from their screens. Looking up, for just one second, interrupts their cellphone interaction but provides valuable information about the immediate environment. A single second of looking up reveals the positions of all obstacles or players within a 10-meter radius. If players look up for two or more seconds, they might be able to track the movement of other players around them. However, players can return to their phones right after, to get as much points as possible interacting with the virtual world.

Satisfaction in this game is earned through continuous interactions with cellphones, rewarding players with $S \log_2 S$ points, where S represents the duration of the interaction. This unique scoring system values prolonged engagement, making one extended interaction more valuable than multiple shorter ones of the same total duration.

1.1.3 The Challenge

The stalls in this bustling flea market are 2-meter by 2-meter squares aligned with the field boundaries. When players approach within 1 meter of a stall's edge, they can execute a transaction to obtain the desired item. Numerous players participate simultaneously, all trying to obtain the same number of items from randomly selected stall locations, as determined by the simulator.

The game doesn't end when players purchase their final item; they can continue to accumulate cellphone satisfaction. The simulator has knowledge of the distance between stalls on the optimal path, allowing it to estimate the time required to complete the circuit without distractions. This estimation sets a simulation time threshold of $T\theta$, where $\theta \geq 1$, representing the level of stringency applied. Smaller values of θ offer less room for delays, while larger values lead to a longer experience.

1.1.4 Success Metric

At the end of the simulation, players will be ranked based on a unique metric: (I, C) , where I signifies the number of items obtained, and C represents the total accumulated cellphone satisfaction. This scoring system awards both efficient navigation and prolonged cellphone interactions. However, visiting all of stalls is still prioritized as a player that visits more stalls will still rank higher than the one with less stalls and more satisfaction points.

Join us as we explore the dynamics of modern life in a digital age, where every step is a challenge to balance between engagement and awareness. This project promises a fun journey into the challenges of completing tasks, navigation, and multitasking in the modern era.

2 Player Evolution

2.1 Initial Stage

2.1.1 Obstacle Avoidance in Initial Stage

When our team first started on the journey of implementing obstacle avoidance for our player, Player 6, we were determined to utilize trigonometry to navigate the obstacle in the virtual flea market effectively. Our primary objective was clear: if Player 6 detected an obstacle within its 10-meter radius and was heading directly towards it, we aimed to employ a strategy that allowed the player to deviate from its collision course.

Before all, we need to know whether an obstacle is in our way. Denote our current position as O , our next stall as S , and an obstacle $ABCD$ (whose coordinates are just $(x \pm 1, y \pm 1)$ if their center is (x, y)). A point P is said to be **in front of us** if

$$\vec{OP} \cdot \vec{OS} > 0$$

where \vec{OS} is our directional vector. The distance of a point P to our directional vector \vec{OS}

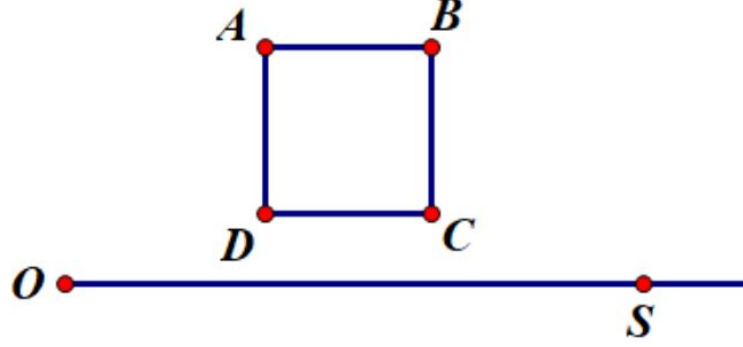


Figure 1: The extreme case an obstacle is not in our way. The two closest vertices are 0.5m away; the other two are 2.5m away.

is

$$h_P := |\vec{OP}| \sin(\arccos \frac{\vec{OP} \cdot \vec{OS}}{|\vec{OP}| |\vec{OS}|})$$

It's easy to see that an obstacle is **in our way** if and only if: any of its four vertices A, B, C, D is in front of us, and (see Figure 1)

$$h_A + h_B + h_C + h_D \leq 6$$

Our initial concept was to consider a 90-degree field of vision ahead of the player, ensuring that any potential obstacles in this critical zone would be addressed. To achieve this, we set out to calculate the angles necessary for Player 6 to make subtle left or right turns, effectively avoiding a head-on collision. The foundation of our approach was to determine these turning angles by finding the arctangent of a fixed distance (2 meters) divided by the distance between Player 6 and the detected obstacle, adjusted by 1 meter to account for the boundary. This trigonometric strategy appeared promising, but it quickly encountered a roadblock in the form of our checkpoint system.

In figure 2, we mapped out how an obstacle is avoided, knowing the black arrow is the original trajectory of the player, the constants $c1$ and $c2$, which should be a number greater

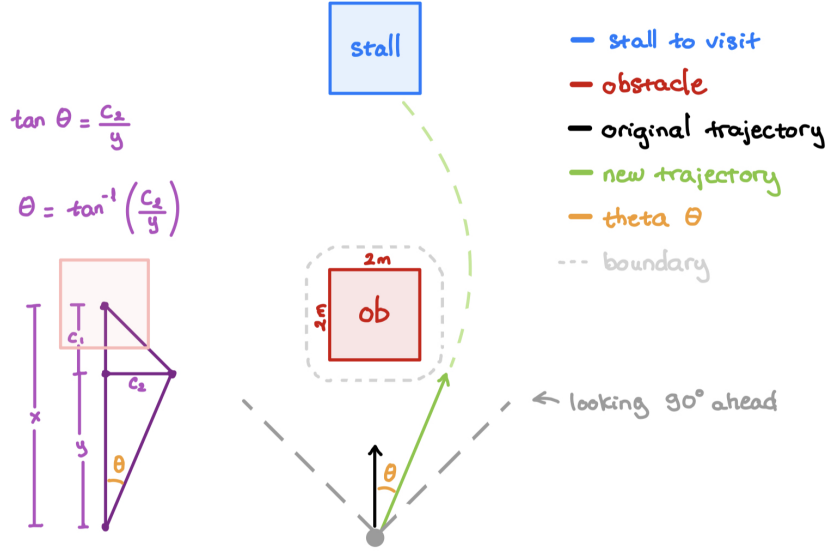


Figure 2: Using Trigonometry for Obstacle Avoidance

than 1.5 meters to account for a safe location to be in, and the distance from the player to the center of the obstacle x , angle theta θ can be calculated. By calculating y , we can calculate the arctangent of c_2/y which gives the an angle θ to shift by. Hence, we get the new green trajectory, safely avoiding the obstacle.

At this stage in our project’s development, we had implemented a checkpoint system, an essential component of Player 6’s navigation strategy. The idea behind this system was to make Player 6 look up at certain predefined checkpoints along its path within the flea market. These checkpoints allowed the player to periodically assess its surroundings and adapt to the dynamic environment. However, the trigonometric approach we had initially envisioned presented a significant challenge. The intricacies of constantly recalculating angles to avoid obstacles and the requirement for precise adjustments to stay within the checkpoint system clashed. Moving Player 6 too far off its intended course, as dictated by the checkpoint system, rendered the approach impractical. The two strategies seemed fundamentally incompatible, and a new direction was needed to reconcile these conflicting priorities.

While our initial approach with obstacle avoidance using trigonometry encountered ob-

stacles of its own, it was not without its merits. The concept of adjusting Player 6's trajectory based on calculations of angles demonstrated the potential for a more dynamic and adaptable approach. As we continued to develop our project, we would need to address this dilemma, ensuring that our obstacle avoidance mechanisms could work in harmony with the checkpoint system while maintaining the safety and efficiency of our player.

2.1.2 Player Avoidance in Initial Stage

Players are similar to obstacles in the sense that we also need to keep away from them. However, players move. So unlike obstacle avoidance, we need frequent look-ups to update their location information in order to avoid hitting them. Meanwhile, we want to look up as less frequently as possible, since we want to gain more satisfaction.

At time t , let our position be O_t , and suppose we looked up, finding the **closest** (in terms of L_2 distance) player **in front of us** is at position P . In the worst case, the player will move towards point C on our directional vector OS such that $|OC| = |PC|$. In this case, we will effectively hit them not at C , but earlier at O' such that $|O'P'| = 0.5$ due to the safety distance of 0.5m (where $|OO'| = |PP'|$ due to same speed of players). The furthest distance we can walk without look-up is (see Figure 3)

$$|O\vec{O}'| = \frac{|\vec{OP}| - 0.5|\vec{OP}||\vec{OS}|}{\vec{OP} \cdot \vec{OS}}$$

Since our simulation is discrete, we will take the flooring and walk for $\lfloor |O\vec{O}'| \rfloor$, and then stop, look-up to see if the player is indeed walking towards us as supposed. If this is the case, wait a round to see if the other player will move away (L_2 distance to us at least 1m); if they don't move, we will back up by take a random unit step to X such that $O'\vec{X} \cdot O'\vec{P}' \leq 0$ and $O'\vec{X} \cdot O'\vec{S} \geq 0$.

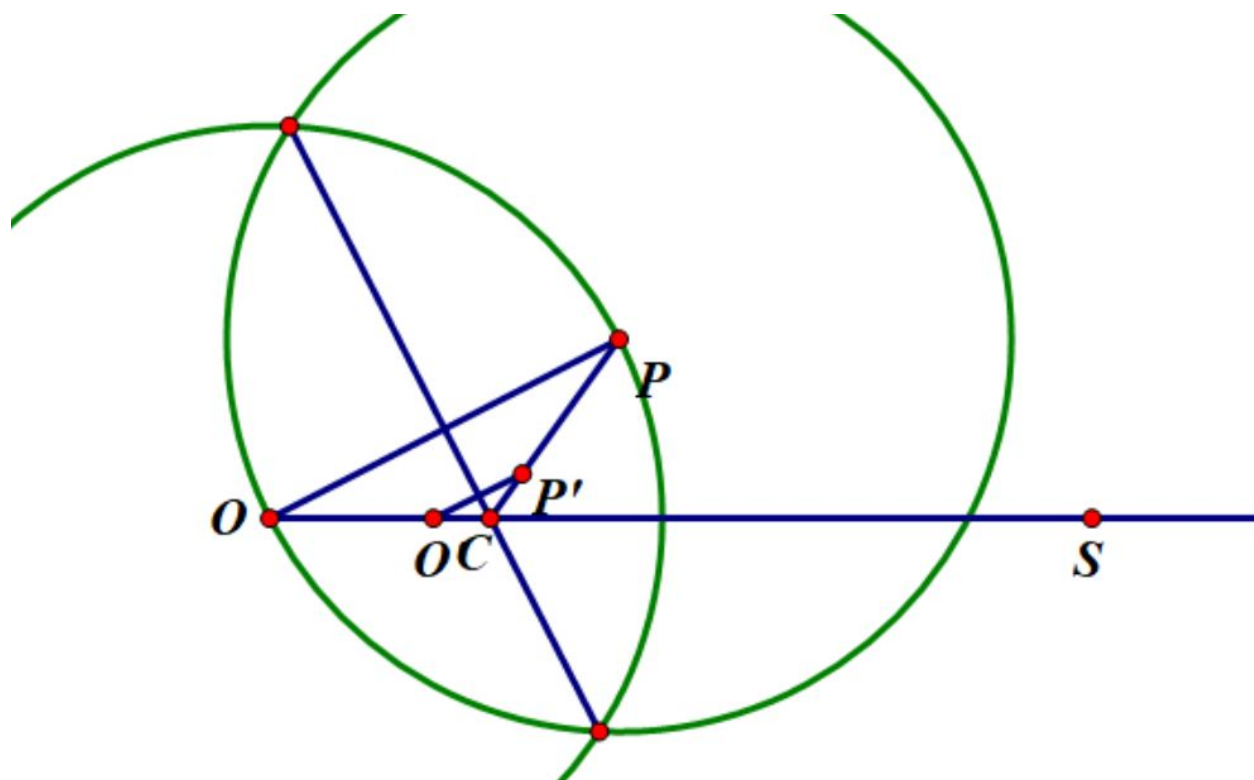


Figure 3: How far we can safely walk without lookup: We can walk up to O' . (Some auxiliary lines were left on the figure)

2.1.3 Navigation in Initial Stage

Initially, our player uses the python package `fast-tsp` to calculate the best order with regard to navigating the stalls expected to visit. We chose to use this package for a variety of reasons. First, we noted that the game code uses this package to calculate the `tsp_path` for each player, in an effort to estimate how many moves are needed for the game. Secondly, the traveling salesman problem is an NP-Hard problem which means that this problem is currently unsolvable computationally in an efficient amount of time. Thirdly, the `fast-tsp` package solves the tsp problem in a relatively fast time and although it does not provide the best, most optimal solution 100% of the time, we found it to give desired results. This tsp path for the player is computed once during the initialization of the player and is used during the player's journey from stall to stall. The path acts as a queue, where next stall is removed from the queue once the player reaches it. After completing the tsp path, the player will remain at the final stall. By using `fast-tsp`, this allowed us to focus and dive deeper on other aspects of this project that required more deep, complex thought such as player and obstacle avoidance.

2.2 Intermediate Stage

2.2.1 Obstacle Avoidance in Intermediate Stage

As we progressed to the intermediate stage of our project, our team recognized the need for a more sophisticated obstacle avoidance strategy than just randomly find a way out whenever we hit an obstacle for our player. Building upon our initial experiments with trigonometry, we sought a solution that allowed the player to adapt to dynamic obstacles effectively. Our approach involved the creation of a set of *known_obstacles*. Whenever Player 6 detected an obstacle within its 10-meter radius, it was promptly added to this list, and since it's a set, the obstacles that are already known will not be added. Then, at regular intervals during its movement, Player 6 would iterate over this list, examining each known obstacle. If an

obstacle appeared within 2.5 meters of the center of its path, it would initiate a 90-degree turn to the left, navigating around the obstacle, all while keeping its ultimate destination, the next stall, in mind. This strategy represented a significant leap forward in terms of obstacle avoidance, particularly when dealing with clusters of obstacles.

The strength of this intermediate strategy is its adaptability to a variety of obstacle scenarios. Whether Player 6 encountered isolated obstacles or clumps of them, the system worked very well. When a single obstacle or multiple obstacles were detected in close proximity, the player adeptly executed the 90-degree turn and continued its journey towards the target stall. This dynamic approach not only enhanced Player 6's ability to avoid obstacles efficiently but also increased its chances of successfully reaching its destination. The adaptability of this system allowed Player 6 to traverse the flea market, even amidst congested areas.

While the intermediate strategy showed significant promise in obstacle avoidance, it was not without its occasional drawbacks. One notable issue came up in cases where Player 6 would exhibit a behavior where it seems to be circling an obstacle or a cluster of obstacles. This circling behavior could occur up to two times as the player attempted to make its way towards the target stall. This unintended behavior, while not detrimental, resulted in the player making additional movements that often led to inefficiencies, potential delays, and the risk of accumulating fewer satisfaction points. Addressing this circling behavior became a focal point for further refinement as we aimed to strike a balance between effective obstacle avoidance and optimal navigation efficiency within the flea market.

2.2.2 Player Avoidance in Intermediate Stage

In terms of avoiding players in the intermediate stage, our team focused on a simple yet effective strategy to avoid players. This strategy was not perfect and contained flaws but was a better implementation than our initial player. Similar to the obstacle avoidance, the player will now constantly turn right when in the direction of a player. This ensures that our

player will avoid colliding with any other players as we turn right instead of colliding with a player. Our methodology behind this approach stems from our discussion in how people avoid bumping into others. Our first instinct is to turn right (or left) and continue to walk forward. This was the basis and starting idea as we began to explore the best way to avoid players. Although this ensured that we avoided players, this methodology combined with a similar methodology for obstacles was not effective. In cases where our player encountered another player in the presence of other obstacles, our player tended to turn right for too many moves and was thrown off track with regard to the next stall. The player drifted too far off course and thus led to our player taking a longer time to finish the tsp stall path. In another case where the player encountered a cluster of 3 to 4 players/obstacles, the player would circle around this cluster 2 to 3 times before making progress to the next stall. We believe this is due to the fact that the player would constantly be in the presence of colliding with a new player/obstacle when turning right. Overall, this strategy ensured we would not collide with a player but in many cases set us back with regard towards our progress towards each stall.

2.2.3 Navigation in Intermediate Stage

In the previous iteration of our project, our initial player's navigation to collect items from the stalls was primarily guided by the Fast-TSP (Traveling Salesman Problem) algorithm. Leveraging the Fast-TSP package, the player efficiently calculated the optimal path to visit the specified stalls, ensuring that it could maximize its item collection within the given time constraints. However, as we transitioned to the intermediate player version, we recognized the need for a more adaptive approach. We aimed to integrate obstacle avoidance into the navigation strategy by dynamically updating the TSP solution. The fundamental concept was to reorganize the sequence of nearby stalls when obstacles were encountered, thereby streamlining the player's path to make it more efficient.

As the player confronted obstacles and executed 90-degree left turns when obstacles

came within 2.5 meters, there was a need to adjust its intended path to reduce the potential inefficiencies caused by these obstacles. The updated TSP feature allowed the player to recalculate its path on the fly, ensuring that it could find another way around obstacles while also reorganizing the sequence of stall visits to regain lost time. This revised approach worked decently with the obstacle avoidance strategy, keeping the player on track towards its goal of collecting satisfaction points efficiently.

While the intermediate player’s navigation strategy was a significant improvement over the initial version, it was not without its challenges. One notable issue that persisted from obstacle avoidance was the player occasionally circling around obstacles or clusters of obstacles. Despite the incorporation of dynamic TSP updates, there were instances where the player’s path planning didn’t fully eradicate the circling behavior. This residual behavior, though relatively infrequent, led to moments where the player expended more time navigating rather than collecting satisfaction points. As we continued our journey towards optimizing player efficiency within the virtual flea market, addressing this circling behavior and fine-tuning the interplay between navigation and obstacle avoidance became key areas of focus for our team’s ongoing efforts. Therefore, this approach, along with the current obstacle avoidance was not kept for the final player stage.

2.3 Final Stage

2.3.1 Unifying Obstacle Avoidance and Player Avoidance

For our final player’s algorithm, we chose to overhaul our previous approaches and use the **A-Star (A*) Algorithm** to calculate the best path to the next stall, effectively avoiding both the obstacles and the players.

Before formally describing the A Star, we first discuss how we unified obstacle avoidance and player avoidance.

We know that players move around. When we lookup at time t , we know their positions P_t . Then, they will diffuse away. For players that are far way, it is very likely that they

will no longer be near there when we reach that region. For players that are in the vicinity of our current position, we will treat them as obstacles located at that point, as they will likely be around there in the next step. We need to take extra caution as we’re likely to hit them soon. Specifically, at a later time T , we could hit them if our distance to their previous position P_t is less than or equal to $T - t - 0.5$. To be safe, we will look up once this distance is less than or equal to $T - t - 1.5$.

To wrap up, our algorithm for players work as follows:

- When we look up, we treat nearby (empirically determined as within 3 meters) players treated as “obstacles” and calculate the A* path (details described in the section below).
- Far away players not considered for A*.
- Look up again when we reach within $T - t - 1.5$ to a last seen player location P_t . Update the path if the players moved.
- Otherwise look up when moved 4.75 meters away from last look up, to avoid unseen players outside horizon. If there are new players nearby, update the path accordingly.

2.3.2 A Star Search

Figure 4 showcases an representation of the A* search algorithm, which is used for our final player. When the A* search algorithm is called, the player searches 12 possible moves from its current position. From these 12 moves, the move that is closest to the next stall is then searched next. This is our heuristic, distance of the player to the next stall. This algorithm continues until an ideal path is found to the next stall or the path is longer than 20 moves. The A star search algorithm will not add any moves that collide with the obstacle which is represented by the dark blue dots in the image. This is an overarching explanation of the A star search algorithm but more granular choices such as path depth, move search space, and when to call the A star search algorithm will be explained further.

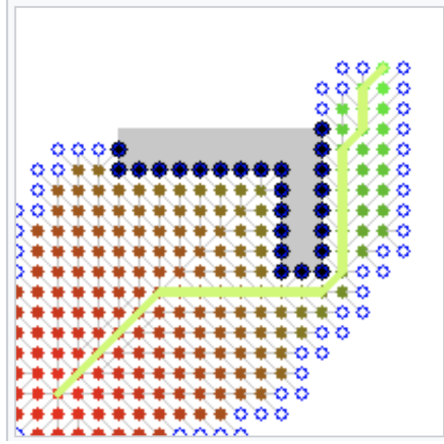


Figure 4: Example A star search algorithm visualized (Source: Wikipedia). The heuristic function is the L2 distance to the next stall.

Originally, the A star search algorithm was called when our player was within 3 units of an obstacle on their path to the next stall. This meant that the A star search algorithm led us to have a relatively small avoidance space and thus potentially fewer moves to avoid the obstacle since the algorithm is called when the player is close to the obstacle. Also, in games where the number of obstacles exceeded 100 the player collided with obstacles quite often. This is due to the fact that our lookup rate is every 3 units, yet we only avoid obstacles that are within 3 units thus allowing for possibilities of colliding with other obstacles due to the player looking up to late. After further testing with the A star algorithm and realizing just how effective the algorithm is, we decided to test the idea of using the algorithm much more often. The A star search algorithm will never add vectors that collide with other obstacles and our player "remembers" obstacles seen. These two facts allow our player to navigate spaces effectively if the player ends up in a area previously explored. The algorithm improves as the player progresses throughout the game due to the knowledge of previously seen obstacles. Thus, we decided to call our A star algorithm much more often: whenever we lookup thus allowing us to compute the best path given any possible new information from looking up. Basically, the A star path is our path to the next obstacle and it gets updated at every lookup. With this change, we found significant improvements in our player. The player now nearly never collides with obstacles/players.




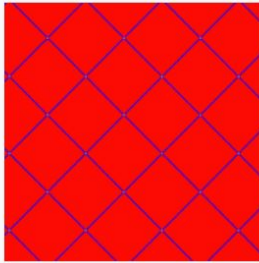
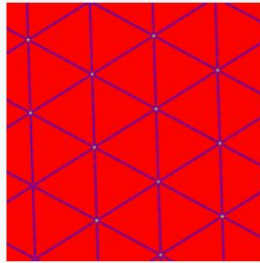
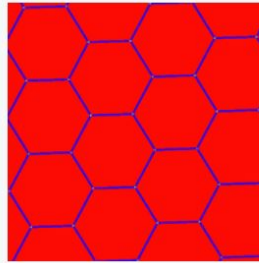
Name	Square tiling (quadrille)	Triangular tiling (deltille)	Hexagonal tiling (hextille)
Symmetry	p4m, [4,4], (*442)	p6m, [6,3], (*632)	
Schläfli {p,q}	{4,4}	{3,6}	{6,3}
Coxeter diagram			
Image			

Figure 5: There are three regular tessellations of the plane. All three have an Euler characteristic (χ) of 0. (Source: Wikipedia)

2.3.3 Search Space: Choosing Search Breadth (Branching Factor)

When searching in the Euclidean plane, one common way to expand is to search the eight nearby integer points (two horizontal $x \pm 1, y$, two vertical $x, y \pm 1$, four (anti-)diagonal $x \pm 1, y \pm 1$ when our current point is x, y). (As seen in Figure 4) Doing so would include four diagonal expanding steps that have length of $\sqrt{2}$. However, we want our search steps correspond to actual moving steps which have length of 1.

However, if we simply expand in 8 directions to $(x + \cos \frac{i\pi}{4}, y + \sin \frac{i\pi}{4})(i \in \{0, \dots, 7\})$, we will see that we will end up with a lot of points in our search space: Let's say we expanded to $(x + 0.5\sqrt{2}, y + 0.5\sqrt{2})$, 7 of the 8 directions from there will be unvisited. By contrast, in the way above, we would only have 3. This is because our expansion is not forming a crystal/**lattice structure** that has a unit cell with periodic, **regular tiling** (tessellation).

On the Euclidean plane, there are three (proper) regular tessellations. (See Figure 5). With side length fixed at 1m, it's easy to see that the **Triangular Tiling** has the most dense distribution of vertices (i.e., our search points), which is why it's chosen for our searching.

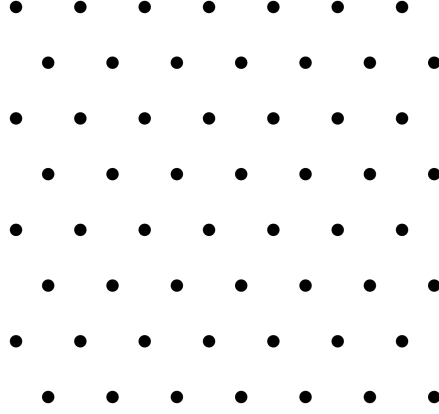


Figure 6: Search Space: Equilateral Triangular Lattice

These vertices form our search space, mathematically the **Equilateral Triangular Lattice**. (See Figure 6)

Hence, we will be expanding in 6 directions to $(x + \cos \theta + \frac{i\pi}{3}, y + \sin \theta + \frac{i\pi}{3})$ ($i \in \{0, \dots, 5\}$, θ is a random angle initialized before running A*) from x, y . From the new point, at most 3 neighbors are unvisited, which significantly decreased our searching computation. The reason to have a random θ is to avoid "dancing/oscillation" when two players with same behavior (e.g. two members in our team) meet trying to avoid each other. By reducing the branching factor, our speed performance was significantly improved.

2.3.4 Choosing Search Depth

The A star path algorithm returns the path after searching for more than 20 units, or the player has reached the stall. The latter case was very obvious to include as if the stall happens to be on the other side of the obstacle and the player found the stall, this is clearly the ideal path. But what about when the stall is very far away from the obstacle? This unique case forced us to do rigorous testing with regard to move breadth and path search depth. Originally, the player included 24 possible moves on a 360 degree space (so a move towards every 15 degree rotation) and search depth of 10. This was effective in games with fewer than 100 obstacles. In games where there was more than 100 obstacles, the player

would be unable to progress past clusters of obstacles. The player would get stuck moving back and forth due to the fact that the player was not able to "see" far enough an ideal path out of the cluster. This led us to increase the search depth to 30 units. Now, the player was able to compute a long ideal path away from obstacle clusters with ease. But, there was a significant trade off with regard to computation time. The player now took over 5 seconds to compute moves when encountering clusters (more than 3) obstacles. To compensate for this, we decided to reduce the move search range to prioritize search space depth over move breadth. Now, the player will search for only 12 possible moves with a search space depth of 30. We still found this computation time to be long, so with further testing a search depth of 20 was ideal. The player was now able to effectively navigate tight clusters and progress around any obstacle with ease in a relatively short time.

2.3.5 Navigation in Final Stage

In our final player, the navigation has two parts:

- Macroscopic path based on stalls: TSP.
- Microscopic path adjustment to avoid obstacles/players: A star.

For the TSP, the final stage basically keeps what was given by the simulator, just the same way initially. We no longer updates TSP path regularly, because we find TSP not really changing, and thus not much performance improvement, whereas the time/computation was significant. However, in some cases we find that the TSP is not really shortest. In the future, we consider using Google's OR Tools which may provide a better TSP solution than fast-tsp.

Our A star search algorithm was fine tuned after much testing to be effective for the current task. These values may differ for other problems but we believe our method of testing will now allow us to create an ideal A star search algorithm for any situation thanks the work done in this project.

3 Tournament

3.1 Tournament and Ranking

The Tournament is where 6 players from 6 different teams with different implementations get to play against each other. The core metric for ranking players within this tournament was their ability to efficiently visit the stalls and collect items. The primary criterion for ranking was the number of stalls successfully visited by each player. The player with the highest count of visited stalls would receive the top ranking. However, in scenarios where two or more players had visited the same number of stalls, creating a tie in stall count, a secondary ranking metric came into play. To break the tie and establish a clear hierarchy, the satisfaction score was employed. The satisfaction score, calculated based on the player's interactions with their cellphones, added a layer of complexity to the competition. Higher satisfaction scores were indicative of more effective and continuous engagement with the cellphone. In these situations, the player with the higher satisfaction score would be positioned ahead of others with the same number of stalls visited. This unique approach to ranking added depth and intrigue to the tournament, ensuring that performance wasn't solely dependent on stall count, but also on the player's ability to maintain satisfaction through digital interaction. The tournament fostered competition and strategic innovation, pushing players to explore diverse tactics and refine their approaches in pursuit of the top-ranking position.

3.2 Tournament Limitations

During the execution of the tournaments, an unexpected bug within our code was found, affecting the composition of participating teams. Regrettably, player 6 was inadvertently excluded from Tournaments 2, 3, 4, and 5. As a result, our report is primarily based on the results of the 84 runs in Tournament 1, where Team 6 was included as a participant.

This unanticipated bug presented a unique challenge in our project's development, as it prevented us from observing the interactions, strategies, and performance of all six player

teams competing against each other. While this limitation significantly reduced the variety of strategies and outcomes we originally aimed to analyze, it also offers an opportunity for future investigations. We found that every time we hit another player, we would be stuck in a loop as the parameters in the loop was incorrect. This bug was quickly fixed, but because of it, team 6 was not a part of tournament 2, 3, 4, and 5.

Despite the limitation imposed by the bug, our report remains a valuable exploration of the strategies, challenges, and innovations employed by us in navigating the virtual flea market. If anything, it highlights the importance of rigorous testing and code maintenance in such projects, as unexpected issues can impact the scope and depth of analysis.

3.3 Tournament Analysis

In Tournament 1, our project featured a diverse range of variable combinations, encompassing the following key parameters:

- Number of Obstacles: The Tournament experimented with various obstacle scenarios, including 0, 50, and 200 obstacles. This wide range of obstacle densities allowed us to assess Player 6's adaptability and obstacle avoidance strategies under different levels of complexity.
- Number of Stalls to Visit: To simulate a spectrum of scenarios, we considered different numbers of stalls for the players to visit, ranging from 2 stalls to 3, 20, and 30 stalls. This allowed us to evaluate the player's performance under varied levels of task load.
- Number of Theta Values: represented by θ , was explored with three different values: 1, 2, and 3. These different values of θ influenced the threshold for simulation time, impacting the tolerance for delays and inefficient paths.

With some runs of the same combination, our tournament featured a total of 84 runs, which became the basis for our analysis. One particular observation from this extensive data

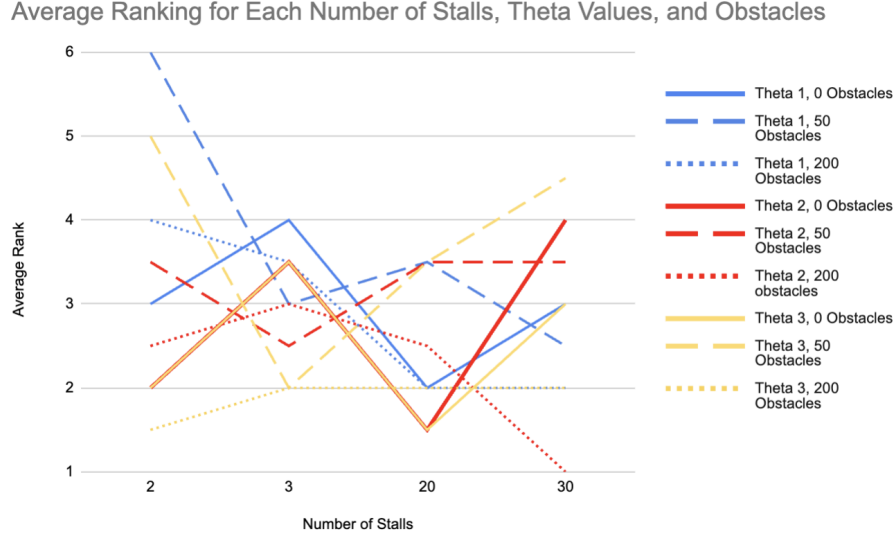


Figure 7: Average Ranking of Each Number of Stalls, Theta, and Obstacle

set is illustrated in Figure 7, which provides insights into the average rankings of Player 6 across these diverse variable combinations.

In Figure 7, a noteworthy trend to point out is the choice of theta θ greatly influences Player 6’s performance. It is evident that, in general, Player 6 tends to achieve lower rankings when with lower θ . With theta 1 (blue lines), the average ranking shown to be positioned above the other thetas (red and yellow lines), which means lower rankings. As we transition to θ values of 2 and 3 (red and yellow lines), the player’s performance improves. These results highlight the impact on Player 6’s ability to adapt, navigate efficiently, and collect satisfaction points in the dynamic environment of the virtual flea market, especially with longer time.

The correlation between θ values and rankings underscores the need for careful consideration when determining the stringency of the simulation time threshold. A possible explanation for this is while player 6 is on its way to visit stalls, it looks up quite frequently, which is every 3 meters, so the satisfaction point doesn’t accumulate too much in that process. This drawback magnifies when the amount of time given is short. However, when we have a large amount of time, we rank higher, which highlights player 6’s ability to collect all

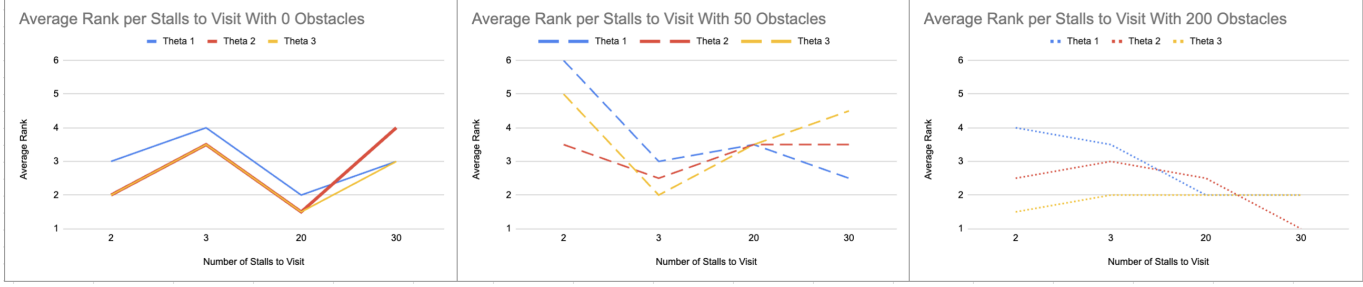


Figure 8: Average Ranking per Stall to Visit, from left to right: 0 obstacle, 50 obstacles, 200 obstacles

stalls fast, and then stop to accumulate satisfaction points when all stall are visited. Because player 6 is faster to collect all items compared to most teams, this also draws attention to its ability to avoid obstacles and players, giving it a longer time span at the end of the game to interact with the phone. This finding not only informs our understanding of the player’s behavior but also offers valuable insights for refining and optimizing our strategies in future iterations of the project.

We observed three key scenarios based on the number of obstacles, i.e., 0, 50, and 200, and their corresponding impacts on the player’s rankings. These scenarios were further differentiated by the number of stalls to visit in Figure 8. In the absence of obstacles, the player’s rank showcased a unique trend, where its performance was optimal with visiting 20 stalls. Conversely, the player faced its most challenging situations when the number of stalls to visit was set at 3 and 30. This trend is consistent across different θ values.

Conversely, when faced with a moderate obstacle density (50 obstacles), Player 6 demonstrated a contrasting trend. In this scenario, the player displayed its optimal performance when visiting either 3 or 30 stalls. On the contrary, the player encountered more challenging conditions when faced with either 2 or 20 stalls to visit. Similarly, the variation in θ values did not significantly alter this trend.

The presence of a high obstacle density, 200 obstacles, prompted Player 6 to excel in scenarios where it was tasked with visiting a higher number of stalls. In this situation, the player displayed the best performance when the number of stalls to visit was increased. In

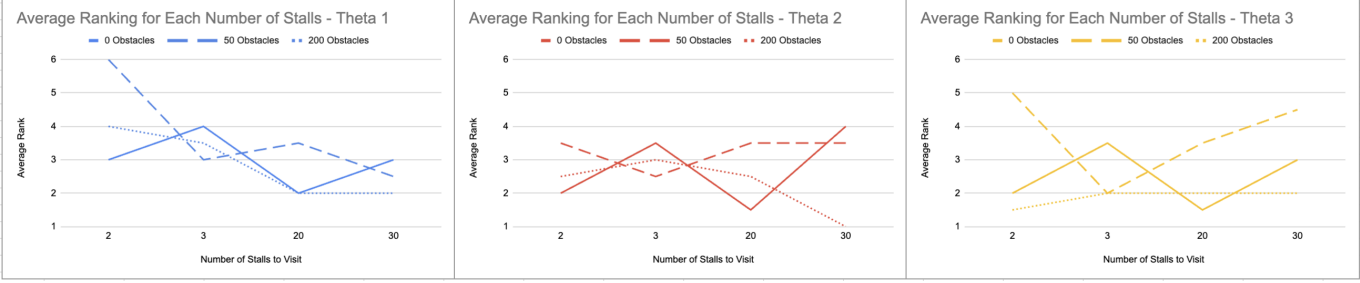


Figure 9: Average Ranking per Stall to Visit, from left to right: theta 1, theta 2, theta 3

particular, the player showcased its optimal capabilities when visiting 20 or 30 stalls, experiencing an upward-trending improvement in its ranking as the number of stalls increased. This trend remained consistent across various θ values, underscoring Player 6’s ability to thrive in demanding scenarios with a higher obstacle density.

These observations underscore the complex relationship between obstacle density, the number of stalls to visit, and the player’s ranking. They offer valuable insights into the adaptability and performance nuances of Player 6 under a variety of challenging and dynamic conditions within the virtual flea market.

To take even a deeper look into the relationship between the player’s ability to navigate through the flea market and the density of the market, we decided to group the data into 3 groups, each group having a different value of theta θ to see if there’s any trend for each θ .

For θ equals 1, in Figure 9, there’s a subtle trend in an improvement of performance as the number of stall and obstacles increases. Interestingly, across all three graphs, runs with higher number of obstacles, i.e. 200 obstacles, tend to give the player higher ranking consistently. This can be observe the best in the third graph, θ equals 3 in Figure 9. This may be attributed to Player 6’s capacity to thrive in challenging and dense scenarios, potentially due to its dynamic obstacle avoidance strategies and its ability to utilize the opportunities presented by a greater number of stalls.

Another surprising point is player 6 does not perform as well especially in runs with 50 obstacles. It’s possible that the player’s performance may be influenced by how its obstacle avoidance mechanisms are tuned to different obstacle densities. In cases with 50 obstacles,

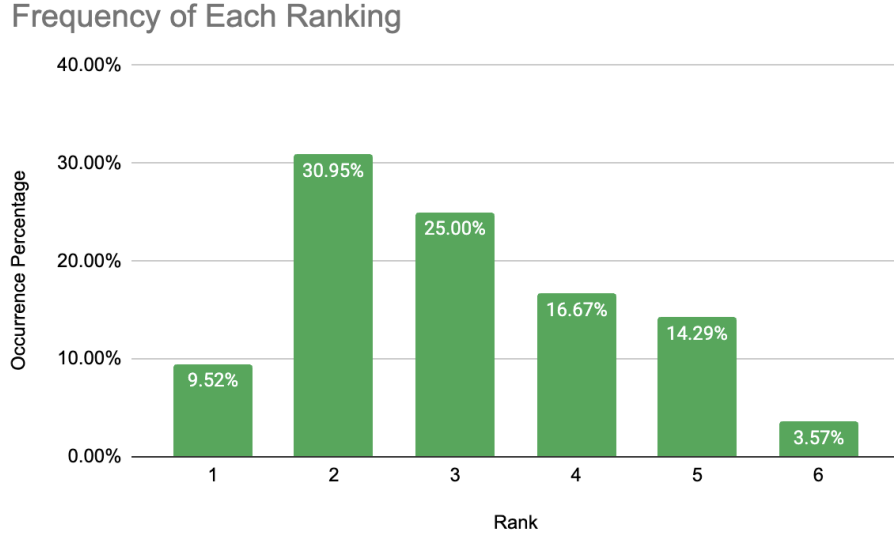


Figure 10: Frequency of Each Ranking

the player may face challenges or limitations that have a unique impact on its performance because it's too sparse for the current settings and algorithms we have in place. The existing parameters and strategies may not be fully optimized to navigate efficiently in scenarios with lower obstacle density, which indicates that the player's ability to navigate in a high density map does not translate to its ability to navigate in a sparse map.

Figure 10, which showcases the frequency of Player 6's rankings across the 84 runs, offers valuable insights into the player's overall performance and consistency. It's notable that Player 6 achieved top rankings quite frequently. With a combined frequency of 40.47%, the player secured either 1st or 2nd place in a significant proportion of the runs. This indicates a level of robustness and competitiveness in its strategies. The player also performed respectably in the mid-tier rankings. Rankings of 3rd and 4th place accounted for 41.67% of the time, showcasing Player 6's ability to consistently secure solid rankings, even when not at the top. While Player 6's performance was generally strong, there were instances where it obtained 5th or 6th place, amounting to a combined 17.86% of the time. These lower rankings may be attributed to the inherent complexity of the virtual flea market and the player's adaptability challenges in certain scenarios. The relatively low frequency of 6th-place rank-

Percentage of Occurrences of Each Ranking for Each Theta Value

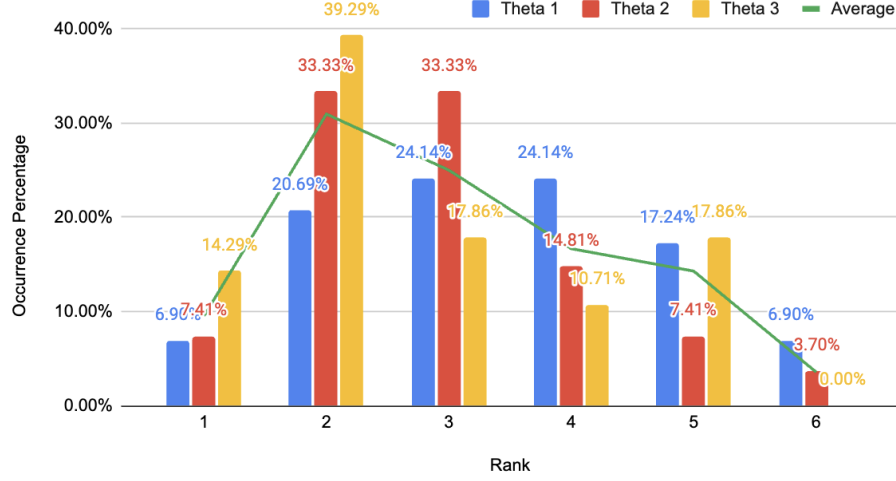


Figure 11: Percentage of Occurrences of Each Ranking for Each Theta Value

ings (3.57%) suggests that Player 6 remained a competitive competitor throughout the 84 runs. Its strategies and adaptability allowed it to avoid the last place in the majority of the trials.

For each theta value in Figure 11, we can observe a notable frequency at $\theta = 3$. First and second place occur most often at $\theta = 3$, which means when we have more time to run, our player can accumulate satisfaction points at the end because we have the ability to finish quite early. However, looking at the lower frequency for first and second place at $\theta = 1$, it shows that either our player didn't do very well with small amount of stalls to visit or when we do finish visiting the small amount stalls required, our frequent look up strategy caused us lower satisfaction points, which led to mostly 3rd and 4th ranking. For $\theta = 2$, we did a bit better than $\theta = 1$ as we had more time to interact with our phone.

When looking at the ranking through the obstacles particularly, in Figure 12, we can observe that player 6 does well with about a third of the time, we get 2nd place in a run with 200 obstacles. It is clear that we did well with either 0 obstacle or 200 obstacles. In between with 50 obstacles, we didn't do very as the red bar is especially high at 5th place. Even though the player did do well at 2nd place as well, but the frequency of being in 5th

Percentage of Occurrences of Each Ranking per Amount of Obstacles

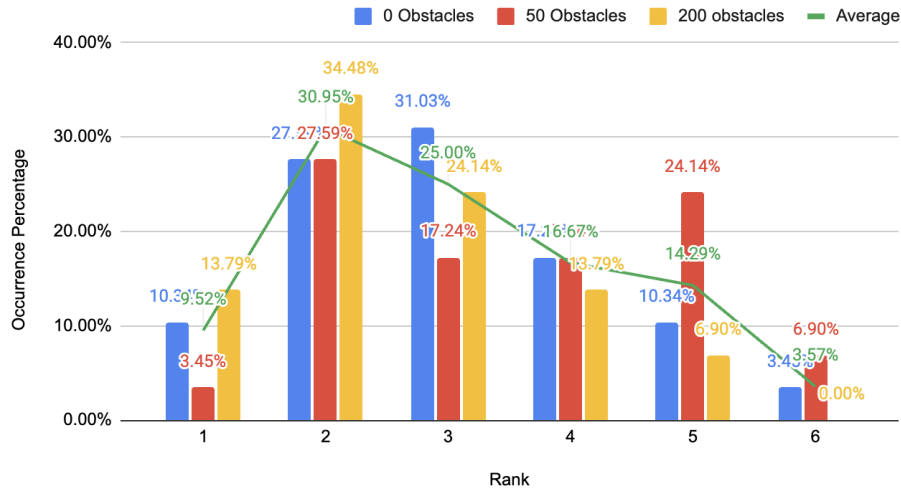


Figure 12: Percentage of Occurrences of Each Ranking per Amount of Obstacles

and 6th place is almost a third of the time. This clearly suggests that our player is not very good at navigating sparse obstacles. We suspect that it is because other players are not as good at navigating in high number of obstacles as we do, but they did very well in a sparse flee market as their look up is less frequent than ours.

Now looking at different number of stalls in Figure 13, two pieces of data that stood out was the frequency of 2nd place when running with 20 stalls and the frequency of 4th place when running with 30 stalls. More than half of the time (55.56%), when running with 20 stalls, our player got 2nd place, which does indicate the optimal amount stalls to visit for player 6 is 20 stalls. And when running with 30 stalls, our player got 4th place more than 40% of the time (42.86%). The rest of the data set seems to be as expected. Contradicting to part of what we theorized from earlier, which was "either our player didn't do very well with small amount of stalls to visit or when we do finish visiting the small amount stalls required, our frequent look up strategy caused us lower satisfaction points," with a small amount of stalls to visit, which are 2 and 3 stalls (red and blue), we fall mostly in 2nd and 3rd place.

In general, we did the best with 20 stalls to visit and worst with 30 stalls to visit. It's

Percentage of Occurrences of Each Ranking per Stalls to Visit

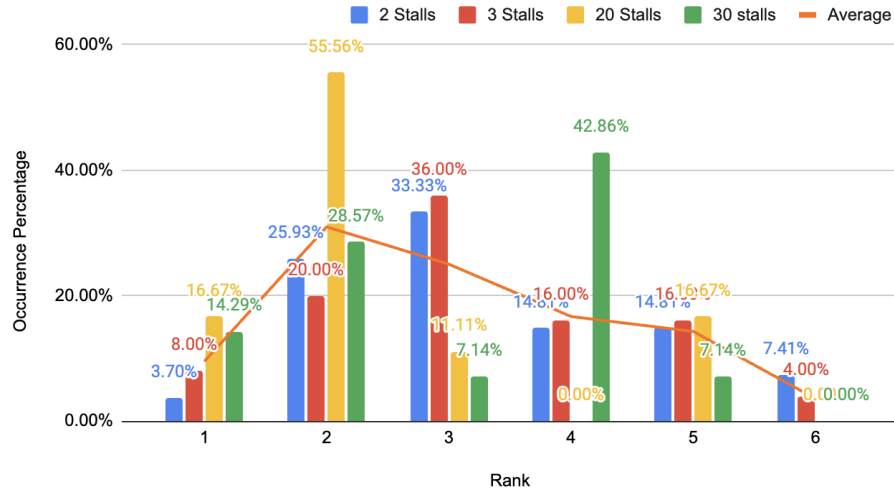


Figure 13: Percentage of Occurrences of Each Ranking per Stalls to Visit

possible that the dynamic strategies employed by Player 6 align well with the characteristics of scenarios featuring 20 stalls. This stall count might strike a balance between task complexity and navigation efficiency, enabling the player to excel. The observation that player 6 tends to rank lower with 30 stalls may be linked to the player's interaction strategy with the cellphone. In scenarios with a higher number of stalls, the balance between satisfaction points from continuous interactions and the need for efficient navigation might be disrupted, leading to lower overall performance. External factors, such as the distribution of stalls, the placement of obstacles, and the arrangement of other players, can significantly impact Player 6's performance. These factors may interact differently with the player's strategies in scenarios with 30 stalls, leading to the observed results.

The unexpected observations suggest that further refinement and fine-tuning of Player 6's strategies may be necessary to optimize its performance across a wider range of scenarios. The player's adaptability and interaction with changing environmental factors could be a key focus for future improvements. In summary, these observations are indeed intriguing and suggest a complex interplay between stall count, player strategies, and performance outcomes. Further analysis and experimentation may be required to uncover the specific

factors that contribute to these observations and to refine the player’s strategies for improved consistency across varying scenarios.

4 Conclusion

In the pursuit of simulating real-world cellphone-induced distractions within a bustling flea market scenario, our team aimed to deliver an effective player for this simulation. We addressed the complexities of multitasking and navigation, ultimately creating a player capable of efficiently interacting with stalls while ”distracted” on their cellphone.

The initial stage saw us exploring trigonometric-based obstacle avoidance, aiming to deviate the player’s course when obstacles were detected. However, this approach encountered challenges, particularly in working with our checkpoint system. While it showed promise, we evolved our strategy in the intermediate stage. Here, we introduced a dynamic obstacle avoidance mechanism, leveraging a set of known obstacles. This enabled the player to adapt more effectively to varying obstacle scenarios, minimizing potential collisions. However, an occasional circling behavior emerged that highlighted how this approach to avoiding obstacles was still too simple. In this stage, our navigation strategy also evolved. The Fast-TSP algorithm guided the player’s path, ensuring optimal item collection. We began to recalculate the TSP throughout the players game, in an effort to be more efficient but in reality this approach was not as promising.

In the final stage, we implemented a significant overhaul, adopting the A* search algorithm for obstacle avoidance and player avoidance. This algorithm effectively computed the best path to the next stall, ensuring the player could navigate around obstacles efficiently. Adjusting the A* algorithm’s parameters allowed for adaptability to various obstacle scenarios, leading to substantial improvements in performance.

Our journey through these stages involved iterative testing, fine-tuning, and trade-offs to strike a balance between obstacle avoidance, navigation efficiency, and computational

resources. Ultimately, our final player demonstrated exceptional capabilities in navigating the flea market while busy in cellphone interactions.

Through our efforts, we successfully developed a player capable of navigating a bustling flea market while remaining distracted on a cellphone. This project serves as a testament to our team’s dedication, adaptability, and problem-solving skills with regard to simulated environments.

5 Future Work

In the future, we can improve in various ways, including but not limited to:

- To improve player avoidance with rigorous testing
- To utilize other algorithms possibly in conjunction with TSP (which we also consider using Google’s OR Tools instead of fast-tsp for better performance) for finding the best possible stall order
- To improve obstacle avoidance when there is a very large amount of obstacles

6 Summary of Contributions

We divided this project into three main sections: obstacle avoidance, player avoidance, and navigation between stalls. Each of us contributed to different sections at different player stages.

- Christian worked on the first implementation of the TSP algorithm and the first implementation for the A* search with regard to obstacle avoidance.
- Fenglei designed the initial player/obstacle avoidance, and implemented the vector tool and the initial player avoidance; designed initial and final A* search and TSP, and implemented their final version.

- Chloe worked on the first implementation of naive obstacle avoidance, update TSP dynamically, tournament analysis, testing, and debugging.