UNIVERSIDADE FEDERAL DE VIÇOSA

Reinforcement Learning Applied to Robot Navigation

Kevin Braathen de Carvalho
Doctor Scientiae

VIÇOSA - MINAS GERAIS
2025

KEVIN BRAATHEN DE CARVALHO

Reinforcement Learning Applied to Robot Navigation

Thesis submitted to the Computer Science Graduate Program of the Universidade Federal de Viçosa in partial fulfillment of the requirements for the degree of *Doctor Scientiae*.

Adviser: Alexandre Santos Brandao

VIÇOSA - MINAS GERAIS
2025

Ficha catalográfica elaborada pela Biblioteca Central da Universidade Federal de Viçosa - Campus Viçosa

T
C331r
2025

Carvalho, Kevin Braathen de, 1992-
Reinforcement learning applied to robot navigation / Kevin Braathen de Carvalho. – Viçosa, MG, 2025.
1 tese eletrônica (224 f.): il. (algumas color.).

Texto em inglês.
Inclui apêndice.
Orientador: Alexandre Santos Brandão.
Tese (doutorado) - Universidade Federal de Viçosa, Departamento de Informática, 2025.
Referências bibliográficas: f. 112-121.
DOI: https://doi.org/10.47328/ufvbbt.2025.434
Modo de acesso: World Wide Web.

1. Inteligência artificial. 2. Aprendizado do computador. 3. Robótica. I. Brandão, Alexandre Santos, 1982-. II. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDD 22. ed. 006.31

Bibliotecário(a) responsável: Euzébio Luiz Pinto CRB-6/3317

KEVIN BRAATHEN DE CARVALHO

Reinforcement Learning Applied to Robot Navigation

Thesis submitted to the Computer Science Graduate Program of the Universidade Federal de Viçosa in partial fulfillment of the requirements for the degree of *Doctor Scientiae*.

APPROVED: March 28, 2025.

Assent:

Kevin Braathen de Carvalho
Author

Alexandre Santos Brandao
Adviser

Dedico esse trabalho à todos aqueles que caminham comigo. Vocês tornam as cores da vida mais vibrantes e os pesos mais leves.

ACKNOWLEDGMENTS

Acredito que somos feitos por aqueles que caminham conosco na longa jornada da vida. Ninguém passa ileso da vida de ninguém e eu sei que tive a maior sorte do mundo por ser marcado por aqueles que cruzaram suas estradas comigo. Me sinto mais forte, mais feliz e mais capaz por ter vocês por perto. Então inicialmente eu agradeço a cada um que dividiu, divide e dividirá essa jornada comigo. A vida é mais leve com vocês nela.

Nominalmente gostaria de começar agradecendo a minha mãe Beth, por nunca ter medido esforços na criação de seus filhos. Cresci num ambiente cheio de amor e cuidado e isso é fundamental pra eu ser a pessoa que sou hoje. Não há palavras que poderiam descrever a importância de todo o seu carinho na minha vida. Dentre os vários ensinamentos que tive vindo de você, um dos mais bonitos e que mais marcam sua presença em minha vida é a do amor incondicional. Lição essa que carrego comigo e faço meu melhor pra passar adiante. Amor traz a superfície aquilo que temos de melhor em nós. Talvez por isso que sempre senti que posso mais.

Às minhas irmãs Karen e Stella eu reservo um agradecimento e um orgulho muito grande. Mesmo pessoas incríveis podem ficar de saco cheio uma das outras se conviverem juntas tempo demais. E tivemos bastante disso não é ? Convivência e de saco cheio um do outro. Sempre que relembro as coisas que já vivemos eu me encho de alegria, pois vejo como que hoje temos um elo muito mais forte, bonito e leve. Mérito nosso, é claro. Essas coisas não se ajeitam por si só. Obrigado por acreditarem sempre em mim e me apoiarem, não importa o contexto.

À minha tia Frankie sou grato por sempre tentar me ensinar a ver o lado mais leve da vida. Ensinar que devemos fazer nosso melhor, acreditar em si e seguir em frente. Almejo do fundo do meu coração ter a leveza de espírito que você tem, esse jeito jovial que contagia e alegra quem está por perto.

À minha vó Honey eu agradeço por todo o apoio que foi dado ao longo da vida. Sempre disposta a escutar minhas reclamações, que não foram poucas. A senhora é a prova viva que amor traz resiliência e que resiliência pode construir coisas muito bonitas, como a família que nós

temos. Sei que nunca vai faltar abrigo quando precisar.

Ao meu vô Per agradeço todos os ensinamentos. O senhor é um exemplo de vida e a prova vida que nada resiste ao esforço. O caminho que foi percorrido pelo senhor é de muita inspiração. Não só pra mim, mas para todos aqueles a sua volta. Espero um dia ter a sabedoria e paz de espírito que o senhor tem. Muita da minha moral, ética foi aprendida contigo. Sem contar toda a bagagem de professor que tenho hoje. Certamente ela foi muito influenciada por ter um exemplo tão bom por perto.

Eu gostaria de poder agradecer nominalmente a cada um dos meus amigos, mas, por sorte minha, são muitos. Vocês que lerem isso sabem que estou falando de vocês. Sejam do rei leão, dos bastardos, da música, dos joguinhos ou 'só' da vida. Amor em suas diversas facetas mudam nossa vida. Amar traz a tona o melhor que podemos oferecer. Ser amado faz a gente sentir e acreditar que somos melhores e que sempre teremos forças para continuar. Eu tenho tudo isso em vocês, me sinto muito querido e amo cada um de vocês. O valor que isso tem pra mim e o impacto disso em quem eu sou e como minha vida é certamente não cabe nesse texto.

Agradeço também aos meus colegas de trabalho no NERo, por aguentaram minha falação, por me ajudarem nos trabalhos e por todo o apoio nesses últimos anos. Isso se estende para todos os colegas do DPI, que muitas vezes tiraram dúvidas me orientaram nos infindáveis perrengues que uma pós graduação pode trazer.

Ao meu orientador Alexandre, agradeço por todas as conversas. Sejam aquelas que direcionaram os trabalhos para rumos mais promissores, ou aquelas sobre a vida. Agradeço por acreditar em mim e fazer questão de defender isso nos momentos em que eu duvidei se merecia estar onde estava. Os ensinamentos sobre ``me sentir doutor'' sempre estarão em minha mente. É bonito ver como as coisas evoluem. Fico muito feliz de como foi meu crescimento como pesquisador e profissional. Muito disso tem sua influência.

Em especial gostaria de agradecer aos meus três orientados, Iure, Hiago e Alexandre, que toparam trilhar esse caminho novo pro laboratório. As

contribuições de vocês não só melhoraram a qualidade desse trabalho, mas também me melhoraram como pessoa. Fico muito feliz que conseguimos fazer progresso com leveza, amizade e confiança. Acredito profundamente que esse é o caminho que se deve seguir para trabalho em equipe. Sem vocês esse trabalho não estaria tão bem feito e eu não estaria tão maduro como profissional.

Agradeço à FAPEMIG pelo apoio financeiro através da bolsa concedida à mim.

Novamente é reconfortante saber que as vitórias da minha vida não são só minhas. É bonito de enxergar a contribuição de cada um de vocês nessa conquista.

This work has been sponsored by the following Brazilian research agencies: Coordination for the Improvement of Higher Education Personnel (CAPES; Financing code 001), Minas Gerais State Foundation for Research Aid (FAPEMIG) and National Council of Scientific and Technological Development (CNPq).

It's the questions we can't answer that teach us the most. They teach us how to think. If you give a man an answer, all he gains is a little fact. But give him a question and he'll look for his own answers.

-Kvothe, The Wise Man's Fear

ABSTRACT

CARVALHO, Kevin Braathen de, D.Sc., Universidade Federal de Viçosa, March, 2025. **Reinforcement Learning Applied to Robot Navigation**. Adviser: Alexandre Santos Brandao.

Path planning is a key aspect of autonomous navigation, especially for autonomous vehicles, where different priorities such as path length, safety, and energy consumption must be considered. Traditional approaches, including dynamic programming and geometric methods, have been widely used to tackle this problem. However, in recent years, artificial intelligence techniques, particularly reinforcement learning, have gained increasing attention. This thesis explores the application of reinforcement learning methods, such as Q-learning, combined with other machine learning techniques like transfer learning, to improve convergence speed and overall performance in various robotic navigation tasks. The research begins with the development of offline global path planning algorithms for both 2D and 3D environments, validated through simulations and real-world experiments. The approach is then adapted for dynamic scenarios, enabling real-time local path planning, ultimately leading to six published papers. The proposed global path planning algorithms can flexibly balance three key priorities while ensuring that the agent can reach its destination from any starting point in the map. This provides robustness against external and internal disturbances. The local path planning adaptation maintains these priorities while operating in real time. Additional contributions include applications in intelligent logistics, such as automated warehouse organization. A curriculum-based training approach was introduced, progressively increasing task difficulty to facilitate learning. This was validated through simulations, with the results currently being finalized in writing. Finally, the thesis discusses the impact of state representation using depth sensors in mapless navigation. It also examines how different hyperparameter settings and Q-table initializations affect the performance of the proposed global path planning algorithms.

Keywords: Robotic Navigation; Reinforcement Learning; Transfer Learning; Path Planning

RESUMO

CARVALHO, Kevin Braathen de, D.Sc., Universidade Federal de Viçosa, março de 2025. **Aprendizado por Reforço Aplicado à Navegação de Robôs**. Orientador: Alexandre Santos Brandao.

O planejamento de trajetória é um aspecto fundamental da navegação autônoma, especialmente para veículos autônomos, onde diferentes prioridades como comprimento do caminho, segurança e consumo de energia devem ser consideradas. Abordagens tradicionais, incluindo programação dinâmica e métodos geométricos, têm sido amplamente utilizadas para resolver este problema. No entanto, nos últimos anos, técnicas de inteligência artificial, particularmente o aprendizado por reforço, ganharam crescente atenção. Esta tese explora a aplicação de métodos de aprendizado por reforço, como o Q-learning, combinados com outras técnicas de aprendizado de máquina, como a aprendizagem por transferência, para melhorar a velocidade de convergência e o desempenho geral em várias tarefas de navegação robótica. A pesquisa começa com o desenvolvimento de algoritmos de planejamento de trajetória global offline para ambientes 2D e 3D, validados por meio de simulações e experimentos no mundo real. A abordagem é então adaptada para cenários dinâmicos, permitindo o planejamento de trajetória local em tempo real, o que resultou em seis artigos publicados. Os algoritmos de planejamento de trajetória global propostos podem equilibrar de forma flexível três prioridades principais, garantindo que o agente possa alcançar seu destino a partir de qualquer ponto de partida no mapa. Isso proporciona robustez contra perturbações externas e internas. A adaptação para o planejamento de trajetória local mantém essas prioridades enquanto opera em tempo real. Contribuições adicionais incluem aplicações em logística inteligente, como a organização automatizada de armazéns. Foi introduzida uma abordagem de treinamento baseada em currículo, aumentando progressivamente a dificuldade da tarefa para facilitar o aprendizado. Isso foi validado por meio de simulações, com os resultados atualmente sendo finalizados por escrito. Finalmente, a tese discute o impacto da representação de estado usando sensores de profundidade na navegação sem mapa. Também examina como diferentes configurações de hiperparâmetros e inicializações da tabela Q afetam o desempenho dos algoritmos de planejamento de trajetória global propostos.

Palavras-chave: Navegação de Robôs; Aprendizado por Reforço; Aprendizado por Transferência; Planejamento de Caminho

List of Figures

Figure 1 – Robotics application examples . . . 22
Figure 2 – Machine Learning Types . . . 24
Figure 3 – Supervised Learning Illustration . . . 24
Figure 4 – Unsupervised Learning illustration . . . 25
Figure 5 – Reinforcement Learning . . . 25
Figure 6 – Transfer Learning illustration. . . . 27
Figure 7 – Types of Reinforcement Learning Algorithms . . . 39
Figure 8 – Heterogeneous Transfer Learning Transformations . . . 43
Figure 9 – Transfer Learning Metrics Illustration . . . 44
Figure 10 – Curriculum Learning Examples . . . 45
Figure 11 – State Representation and Action Space . . . 48
Figure 12 – States and Actions for the proposed method with the arrows being the possible actions and the cell index indicating the state number. . . . 56
Figure 13 – Local Grid position in respect to the agent's last action . . . 56
Figure 14 – Example of local grid rotation due to no clear path between the agent and the local goal. The local goal is defined as the empty cell closest to the global goal, represented as a green G. . . . 57
Figure 15 – Example of motion behavior that could lead to collisions in real world applications and the expected correction from the fine tuning reward. . . 58
Figure 16 – Illustration of the First Stage of Training . . . 61
Figure 17 – States and Actions for the proposed method. . . . 62
Figure 18 – Illustration of the Second Stage of Training . . . 63
Figure 19 – Illustration of the Third Stage of Training . . . 63
Figure 20 – Warehouse scenario simulation . . . 64
Figure 21 – Simulation environments for training the agent and testing its performance, where starting bases are shown in blue and ending goals are indicated in red. . . . 67
Figure 22 – Depth scans are selected from -90° to 90°, centered in front of the robot, and split into 4 sectors with equal sample size. . . . 68
Figure 23 – LiDAR depths scans are split into 4 and 10 sectors respectively, resulting in sample sizes in each sector for the same measurement. . . . 68
Figure 24 – Selecting sector #2/4 and #4/10 demonstrates that the same edge on the wall (seen in Fig. 22) results in varying distributions and therefore different mean, mode and soft_min values. . . . 69
Figure 25 – A fully connected network is used to estimate Q-values for each of the 3 possible actions, given a state. . . . 73

Figure 26 – A Selector network is trained for the mapless autonomous navigation task, supported by a less-often updated Evaluator network (DDQN). . 74
Figure 27 – Map 1: Paths taking into account different sets of priorities. 77
Figure 28 – Map 2: Paths taking into account different sets of priorities 78
Figure 29 – Optimal Policy for Map 3 taking into account all the three priorities. . 79
Figure 30 – Optimal Policy for Map 4 taking into account all the three priorities. . 80
Figure 31 – Paths for different priorities on Map 1, with size of 10m×10m×1.5m. . 82
Figure 32 – Paths for different priorities on Map 2, with size of 12m×12m×1.5m. . 83
Figure 33 – Paths for different priorities on Map 3, with size of 12mx12mx1.5m. . . 84
Figure 34 – Paths starting from alternative locations map 1. 84
Figure 35 – Paths starting from alternative locations on map 2. 85
Figure 36 – Paths starting from alternative locations on map 3. 85
Figure 37 – Simulation Results. Images (a)-(c) display the paths taken by different priority sets as well as the shortest path obtained by Dijkstra. Images (d)-(f) display the policy obtained after training taking all priorities in account. 87
Figure 38 – Maps 4,5 and 6 used for real-world experiments. Red cells represents the agent's goal. 90
Figure 39 – Experimental setup. Motion capture cameras are set on the upper part of the room, some of them outside the range of the photo. 90
Figure 40 – Training Result for Average Reward. 93
Figure 41 – Training Result for Average Distance. 93
Figure 42 – Path examples in some of the environments, with the number accompanying L standing for the respective curriculum level of the image. . . . 95
Figure 43 – Examples of 3-Dimensional Trajectory of the UAV. 95
Figure 44 – Snapshot after three path re-plannings for first environment. 97
Figure 45 – Full path for each environment. The green cells represent the local goals, with darker tone to represent the passing of time. The dotted boxes represent the local grid used to plan each local path with the color coding representing the passing of time. 98
Figure 46 – Mean of the success rate of the last 10 episodes. 100
Figure 47 – Scalability test in warehouse scenario with 2 up to 7 agents. 102
Figure 48 – Reinforcement learning training results. 102
Figure 49 – Rewards results are shown for each DSRM in terms of sampling and sector number, measured during validation loops within the training stage. Initial rewards are mostly negative, arising from the collision and step reward functions, and converge to positive values as the model learns to succeed in collision-free mapless navigation. Note that mode10 DSRM fails to converge, reaching a learning plateau at -200 reward points. 105

Figure 50 – Success and collision rates are seen for each DSRM over 25k training episodes. Model training is shown to result in less collisions and more successes as desired. As in Fig. 49, mode10 failed to converge as noted as low SR and high CR values. 106
Figure 51 – Model performance in 5 tests is averaged over 1k episodes and measured on Success Rate (SR), Collision Rate (CR), Total Distance Traveled (TDT) and Mean Distance to Objects (MDTO) from left to right. Increasing sector number resulted in SR and decreasing CR for the mean sampling methods, while softmin and mode had varying results. The top individual performing model is softmin5, resulting in the highest SR, lowest CR and highest MDTO, representing respectively a successful and safe model. However, mean sampling is considered the best representation methodology following from its performance increase consistency based on sector number adjustment. 107
Figure 52 – Timeline illustrating the progression of works developed throughout this research. Blue circles correspond to single-agent path planning contributions, red circles indicate works related to intelligent logistics applications, the orange circle represents the survey on machine learning techniques for UAV navigation, and the green circle highlights the mapless navigation contribution. The last paper is still to be submitted. 110
Figure 53 – Map used for each simulation in this section. Green and blue cells represent the starting and ending point respectively. Red dashed lines represent the euclidean distance to occupied cells, orange dashed circles represent each turn. 124
Figure 54 – Parallel Coordinates Plot for Metrics in all Simulations. Axis inside green box are the hyperparameters used, and axis inside red box are the evaluated metrics. Green highlighted line constant value of 0.9 for ϵ and constant value of 0.2 for α. Red highlighted line used a variable interval of [0.3, 0.8] for ϵ and a constant value of 0.3 for α 129
Figure 55 – Parallel Coordinates Plot for Threshold in all Simulations. 130
Figure 56 – All Paths Path Length average throughout training. Sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively. Starting value is represented by a red triangle and final value by a green circle. 132

Figure 57 – All Paths Mean Obstacle Distance average throughout training. Sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively. Starting value is represented by a red triangle and final value by a green circle. 134
Figure 58 – All Paths Total Turns average throughout training. Sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively. Starting value is represented by a red triangle and final value by a green circle. 136
Figure 59 – Main Path Path Length average throughout training. Sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively. Starting value is represented by a red triangle and final value by a green circle. 139
Figure 60 – Main Path Mean Obstacle Distance average throughout training. Sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively. Starting value is represented by a red triangle and final value by a green circle. 141
Figure 61 – Main Path Total Turns average throughout training. Sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively. Starting value is represented by a red triangle and final value by a green circle. 143
Figure 62 – Box Plot for All Paths Path Length average metric. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively. 146
Figure 63 – Box Plot for All Paths Mean Obstacle Distance average metric. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively. 148
Figure 64 – Box Plot for All Paths Total Turns average metric. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively. 150
Figure 65 – Box Plot for Main Path Path Length average metric. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively. 152

Figure 66 – Box Plot for Main Path Mean Obstacle Distance average metric. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively. 154
Figure 67 – Box Plot for Main Path Total Turns average metric. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively. 155
Figure 68 – Box Plot for All Paths Path Length metric threshold average. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively. 157
Figure 69 – Box Plot for All Paths Mean Obstacle Distance metric threshold average. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively. 159
Figure 70 – Box Plot for All Paths Total Turns metric threshold average. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively. 161
Figure 71 – Box Plot for Main Path Path Length metric threshold average. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively. 163
Figure 72 – Box Plot for Main Path Mean Obstacle Distance metric threshold average. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively. 165
Figure 73 – Box Plot for Main Path Total Turns metric threshold average. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively. 167
Figure 74 – Map used for each simulation in this section. Green and blue cells represent the starting and ending point respectively. Red dashed lines represent the euclidean distance to occupied cells, orange dashed circles represent each turn. 169
Figure 75 – Parallel Coordinates Plot for Metrics in all Simulations. Axis inside green box are the hyperparameters used, and axis inside red box are the evaluated metrics. Green highlighted line constant value of 0.9 for ϵ and 1 for Exploring Starts. 173
Figure 76 – Parallel Coordinates Plot for Metrics in all Simulations. Axis inside green box are the hyperparameters used, and axis inside red box are the evaluated metrics. 174
Figure 77 – All Paths Path Length average throughout training. Sub figures (a), (b) show the results for constant and variable ϵ values, respectively. Starting value is represented by a red triangle and final value by a green circle. . 175

Figure 78 – All Paths Mean Obstacle Distance average throughout training. Sub figures (a), (b) show the results for constant and variable ϵ values, respectively. Starting value is represented by a red triangle and final value by a green circle. 176
Figure 79 – All Paths Total Turns average throughout training. Sub figures (a), (b) show the results for constant and variable ϵ values, respectively. Starting value is represented by a red triangle and final value by a green circle. . 178
Figure 80 – Main Path Path Length average throughout training. Sub figures (a), (b) show the results for constant and variable ϵ values, respectively. Starting value is represented by a red triangle and final value by a green circle. 179
Figure 81 – Main Path Mean Obstacle Distance average throughout training. Sub figures (a), (b) show the results for constant and variable ϵ values, respectively. Starting value is represented by a red triangle and final value by a green circle. 180
Figure 82 – Main Path Total Turns average throughout training. Sub figures (a), (b) show the results for constant and variable ϵ values, respectively. Starting value is represented by a red triangle and final value by a green circle. . 181
Figure 83 – Box Plot for All Paths Path Length average metric. The colors light blue and light green represent the groups A and B, respectively. 183
Figure 84 – Box Plot for All Paths Mean Obstacle Distance average metric. The colors light blue and light green represent the groups A and B, respectively. 185
Figure 85 – Box Plot for All Paths Total Turns average metric. The colors light blue and light green represent the groups A and B, respectively. 187
Figure 86 – Box Plot for Main Path Path Length average metric. The colors light blue and light green represent the groups A and B, respectively. 188
Figure 87 – Box Plot for Main Path Mean Obstacle Distance average metric. The colors light blue and light green represent the groups A and B, respectively. 190
Figure 88 – Box Plot for Main Path Total Turns average metric. The colors light blue and light green represent the groups A and B, respectively. 191
Figure 89 – Box Plot for All Paths Path Length metric threshold average. The colors light blue and light green represent the groups A and B, respectively. . 193
Figure 90 – Box Plot for All Paths Mean Obstacle Distance metric threshold average. The colors light blue and light green represent the groups A and B, respectively. 195
Figure 91 – Box Plot for All Paths Total Turns metric threshold average. The colors light blue and light green represent the groups A and B, respectively. . 197

Figure 92 – Box Plot for Main Path average Path Length metric threshold. The colors light blue and light green represent the groups A and B, respectively. . 198
Figure 93 – Box Plot for Main Path Mean Obstacle Distance metric threshold average. The colors light blue and light green represent the groups A and B, respectively. 200
Figure 94 – Box Plot for Main Path Total Turns metric threshold average. The colors light blue and light green represent the groups A and B, respectively. . 202
Figure 95 – Map used for each simulation in this section. Green and blue cells represent the starting and ending point respectively. Red dashed lines represent the euclidean distance to occupied cells, orange dashed circles represent each turn. 203
Figure 96 – Parallel Coordinates Plot for Metrics in all Simulations for Map 1. Saf. Const stands for K_s, St. Const stands for K_d and Ener. Const stands for K_t. 206
Figure 97 – Parallel Coordinates Plot for Threshold in all Simulations for Map 1. Saf. Const stands for K_s, St. Const stands for K_d and Ener. Const stands for K_t. 207
Figure 98 – Map 1 used in this section's simulations. Green and blue cells represent the starting and ending point respectively. Red dashed lines represent the euclidean distance to occupied cells, orange dashed circles represent each turn. 209
Figure 99 – Map 2 used in this section's simulations. Green and blue cells represent the starting and ending point respectively. Red dashed lines represent the euclidean distance to occupied cells, orange dashed circles represent each turn. 210
Figure 100 – All Paths Path Length average throughout training for each Map, Initialization Q-value and Method. 212
Figure 101 – All Paths Mean Obstacle Distance average throughout training for each Map, Initialization Q-value and Method. 214
Figure 102 – All Paths Total Turns average throughout training for each Map, Initialization Q-value and Method. 215
Figure 103 – Main Path Path Length average throughout training for each Map, Initialization Q-value and Method. 217
Figure 104 – Main Path Mean Obstacle Distance average throughout training for each Map, Initialization Q-value and Method. 219
Figure 105 – Main Path Total Turns average throughout training for each Map, Initialization Q-value and Method. 221

List of Tables

Table 1 – Time consumed for 10 000 iterations on each map. 80
Table 2 – Performance Metrics for Map 1 82
Table 3 – Performance Metrics for Map 2. 83
Table 4 – Performance Metrics for Map 3. 83
Table 5 – Time consumed for 50 000 iterations on each map with 5 levels of altitude. 85
Table 6 – Simulation Performance Metrics 88
Table 7 – Average time consumed for 100 000 iterations on each map. 89
Table 8 – Sequence of Experiments. 91
Table 9 – Environment Details. 92
Table 10 – Performance Metrics Comparison 94
Table 11 – Environment Details 100
Table 12 – The success rates of tasks according to training episode lengths 102
Table 13 – Median of test metrics averaged over 1k episodes for each DSRM demonstrate that mode sampling displays the most consistent adjustment, such that mode10 is the ideal DSRM with similar results to the top-performing model (*softmin5*). 107
Table 14 – ϵ and α Permutation Indexes. Group A has constant values for ϵ and α, B constant values for ϵ and variable for α, C variable values for ϵ and constant for α and D has variable values for both. 128
Table 15 – ϵ and Exploring Starts (Exp. Strt) Permutation Indexes. Group A has constant values for ϵ and B constant values for ϵ. 172
Table 16 – Combinations of K_s, K_d, and K_t with assigned indices. 205
Table 17 – Final Result Metrics by Map and Approach for Initial Q-Value of 5. . . 222
Table 18 – Final Result Metrics by Map and Approach for Initial Q-Value of 50. . . 223

Contents

1 INTRODUCTION 21
1.1 Glossary 27
1.2 The Problem and its Impact 28
1.3 Objectives 29
1.4 Related Works 30
1.4.1 Path Planning Strategies 30
1.4.2 Machine Learning applied to Intelligent Logistics Scenarios 32
1.5 Contributions 33
1.6 Work Structure 35

2 THEORETICAL BACKGROUND 37
2.1 Reinforcement Learning 37
2.2 Transfer Learning 41

3 METHODOLOGY 47
3.1 An Offline Q-learning based global path planner for 2D and 3D environments with priority shifting rewards. 47
3.1.1 Reward Shaping 48
3.1.2 Training procedure 50
3.2 An Offline Deep Q-learning based global path planner for 3D environments with priority shifting rewards. 51
3.2.1 Reward Shaping 52
3.2.2 Deep Q-Learning 53
3.2.3 Training Procedure 54
3.3 An Online Q-learning based local path planner for 2D environments with priority shifting rewards. 54
3.3.1 Reward Shaping 54
3.3.2 Local Grid 55
3.3.3 Training Procedure 57
3.3.4 Path generation 58
3.4 Curriculum-based Reinforcement Learning for an Effective Multi-Agent Path Planning Algorithm in Warehouse Scenarios 59
3.4.1 Stage One: Free Space 61
3.4.2 Stage two: Static Environment 62
3.4.3 Stage three: Dynamic Environment 62
3.4.4 Stage four: Desk Operation 63

3.4.5 Transfer Learning . . . 64
3.5 State Representation Selection for Mapless Autonomous Navigation with Deep Reinforcement Learning . . . 66
3.5.1 Simulation Environments . . . 66
3.5.2 State and Action Space . . . 69
3.5.3 Reward Shaping . . . 70
3.5.4 Double Deep Q-Learning . . . 72
3.5.5 Training Procedure . . . 74

4 RESULTS AND DISCUSSION: GLOBAL AND LOCAL PATH PLANNING FOR SINGLE AGENT . . . 76
4.1 2D Global Planner Algorithm Validated with Simulations . . . 76
4.1.1 Simulations for Path Planning with Different Priorities . . . 77
4.1.2 Simulations to Analyze Optimal Policy . . . 78
4.1.3 Simulations for Time Consumption . . . 80
4.2 3D Global Planner Algorithm Validated with Simulations . . . 81
4.2.1 Simulation Settings . . . 81
4.2.2 Simulation Results . . . 82
4.3 2D Global Planner Algorithm Validated with Simulations and Experiments . . . 86
4.3.1 Simulation Results . . . 86
4.3.2 Experiments Results . . . 89
4.4 3D Deep Q-Learning Global Planner Algorithm Validated with Simulations . . . 92
4.4.1 Quantitative Results . . . 93
4.4.2 Qualitative Results . . . 94
4.5 2D Local Planner Algorithm Validated with Simulations . . . 96
4.5.1 Simulation Settings . . . 96
4.5.2 Simulation Results . . . 96
4.6 Path Planning in Intelligent Logistics Scenarios . . . 99
4.6.1 Curriculum Learning Tests . . . 100
4.6.2 Transfer Learning Test . . . 101
4.6.3 Transfer Learning and Curriculum Learning Test . . . 101
4.6.4 Scalability Tests . . . 101
4.6.5 General Results Discussion . . . 102
4.7 State Representation Selection for Mapless Autonomous Navigation with Deep Reinforcement Learning . . . 103
4.7.1 Analyzed Metrics . . . 104
4.7.2 Training Convergence and Validation . . . 105
4.7.3 Performance Testing . . . 106

5 CONCLUDING REMARKS 109

BIBLIOGRAPHY 112

APPENDIX 122

APPENDIX A – HYPERPAREMETER AND Q-TABLE INITIALIZATION ANALYSIS 123

A.1 Comparison of Epsilon and Alpha Hyperparameters 123
A.1.1 Performance Metrics and Representations 125
A.1.2 Parallel Coordinates Plots 126
A.1.3 3D Plots for All Paths 127
A.1.4 3D Plots for Main Path 138
A.1.5 Box Plots for All Paths Metrics 145
A.1.6 Box Plots for Main Path Metrics 151
A.1.7 Box Plots for All Paths Metrics Thresholds 156
A.1.8 Box Plots for Main Path Metrics Thresholds 162
A.2 Comparison of Epsilon and Exploring Starts Hyperparameters . . 166
A.2.1 Performance Metrics and Representations 168
A.2.2 Parallel Coordinates Plots for All Paths 170
A.2.3 3D Plots for All Paths 171
A.2.4 3D Plots for Main Path 178
A.2.5 Box Plots for All Paths Metrics 182
A.2.6 Box Plots for Main Path Metrics 186
A.2.7 Box Plots for All Paths Metrics Thresholds 192
A.2.8 Box Plots for Main Path Metrics Thresholds 196
A.3 Comparison of Reward Constant Hyperparameters 201
A.3.1 Performance Metrics and Representations 203
A.3.2 Parallel Coordinates Plots 204
A.4 Q-Table Initialization 208
A.4.1 2D Plots for All Paths Metrics 211
A.4.2 2D Plots for Main Path Metrics 216
A.4.3 Final Result Metrics 220

1 Introduction

Robot agent employment for different activities is a topic of great interest for the scientific community. This is largely due to the possibility of utilizing such agents, individually or cooperatively, in weary missions, repetitive tasks or environments that are unsafe for humans (RUBIO; VALERO; LLOPIS-ALBERT, 2019). In the past years studies focused both in Unmanned Ground Vehicles (UGV) as well as Unmanned Aerial Vehicles (UAV) were done and there are many examples in the literature. Tasks such as search and rescue, industrial applications, mapping, assistive robotics as well as surveillance and inspection can have their performance and/or flexibility greatly enhanced by the usage of one or more robotic agents.

Mapping can be crucial to different applications, due to the extra information a reliable map can provide to the situation. Global path planning algorithms can be employed when a map is available, providing room for optimal planning. Kähler, Prisacariu e Murray (2016) addresses one of the main concerns in mapping applications: loop closure detection. When navigating, an agent is prone to accumulating odometry errors and, thus, carrying these errors to the mapping procedure. Loop-Closure detection aims to identify closed-loops in the agents route and correct the mapping errors. Figure 1(d) shows an illustration of this application.

Search and rescue applications require careful map less navigation or a reliable SLAM (Simultaneous Location And Mapping) algorithm given the uncertainty present in a disaster situation. High adaptability is required to address this task and it can greatly benefit by the cooperation of multiple agents in a homogeneous or heterogeneous team composition. Dubé et al. (2016) apply a 3D SLAM algorithm in order to reconstruct the environment to better act in search and rescue missions. On the other hand, Rodríguez et al. (2020) propose a cooperation framework to benefit from the innate freedom of movement of Unmanned Aerial Vehicles (UAV) to better assist Unmanned Ground Vehicles (UGV) in wilderness search and rescue scenarios. Figure 1(a) illustrates this application.

Surveillance and inspection missions can be greatly aided by the use of one or more robotic agents, being UAV, UGV or a combination of both. There is a great variety of scenarios in this type of task, such as urban areas, industrial sites or even in the wilderness, which lead to an inherent high dimensionality and uncertainty to theses situations. Julian e Kochenderfer (2019) apply deep reinforcement learning techniques to achieve decentralized control strategies for a team of UAVs aid firefighters controlling wildfires. Liebisch et al. (2016) coordinates a heterogeneous robot team to provide automatic crop management. A UAV surveys the region from above, providing information about the scenario and the

UGV performs targeted interventions in the field. Figure 1(b) provides an illustration of this application.

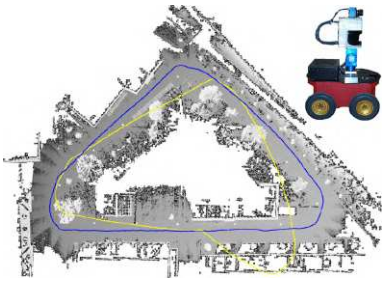
As for industrial applications, their current state and future projections have heavily shifted in the passed years with Industry 4.0 concepts, leading to new manufacturing and distribution processes. Robots are expected to carry out the work, or at least collaborate with it, while humans supervise and ensure quality in other processes (EVJEMO et al., 2020). Oliff et al. (2020) model the human-robot interaction in assistive manufacturing applications and uses reinforcement learning to improve it. In the context of logistics, Rey et al. (2019) propose the employment of multiple robotic agents to work in collaboration with humans in warehouse scenarios. Graph search methods such as Lazy Theta are used to achieve effective global planning. Figure 1(c) showcases an example of this application.

Assistive Robotics is a field of application that is related to human well being. In the past decades its usage has been investigated in applications such as physical or mental therapy. It is multi-disciplinary in its very core, having to deal with trust, identification and socialization challenges. Langer et al. (2019) discuss the issue with trust in socially interactions between humans and robots, while Clabaugh et al. (2019) propose a level of personalization in an in-house assistive robot for autistic children. Figure 1(e) exemplifies this application.

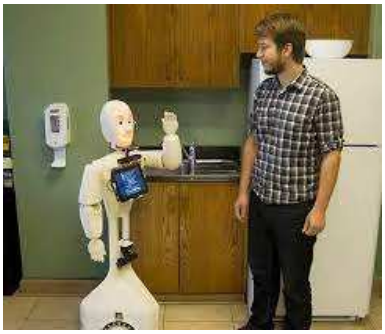
(a) Source:(ARMBRUST; CUBBER; BERNS, 2014)

(b) Source:(SURVAILLANCE, 2022)

(c) Source:(INDUSTRIAL-2022, 2022)



(d) Source:(MAPPING, 2022)



(e) Source:(ASSISTIVE-2022, 2022)

Figure 1 – Robotics application examples: (a) UGV being used for a Search and Rescue situation (b) UAV being used to inspect and surveill crops (c) Robotic arm in a industrial setting (d) UGV using external sensors to map the environment (e) Assistive robot interacting with a human

Safe and efficient navigation is one of their most crucial parts in most robotics applications and even though there is many works in the literature involving this topic, it maintains as a field of great interest in the research community (SERRANO-PÉREZ et al., 2021).

Robotic navigation can be classified in respect to the information and the type of path planning done. In regard to the information used to navigate, there are mainly two types: map-based and mapless navigation (RAHMANI et al., 2015). The first one is especially useful when it comes to structured and known environments. If applied in unknown locations, a reliable SLAM algorithm must be used to provide partial information initially and perhaps a complete environment representation after some time. Qin et al. (2019) use a heterogeneous robot team to achieve efficient environment perception in GPS-denied environments. When it comes to environments were prior information is not available, or were a SLAM based reconstruction is not feasible, mapless navigation strategies must take place and require a high degree of adaptability in order to be successful.

There are three types of algorithms when it comes to path planning for mobile robotics. Global Path Planning, also referred as deliberative navigation, Local Path Planning, also referred as reactive navigation, and Hybrid Path Planning, which combines both previous mentioned approaches. The first one required prior environment information and its focus is to provide a global solution that leads the agent from the start spot to its destination. It can be done taking in account different priorities such as path length, safety as well as energy consumption rates. Ou et al. (2022) use the famous meta-heuristic Genetic Algorithm to provide global path planning. Event though this type of algorithm can sometimes guarantee a globally optimal solution, it is prone to the caveat of lacking the adaptability required for more dynamic scenarios. The second one is designed to be able to react to the uncertainty of a given scenario. Many different algorithms have being used to solve this problem, such as the Artificial Potential Fields (APF) technique, but like many other approaches, it suffers from local minima issues. Szczepanski, Bereit e Tarczewski (2021) propose a APF modification supported by augmented reality to avoid this problem. Finally, the third one takes the best of the other two approaches. They are able to provide a global solution and react to environment modifications throughout its path. Piccinelli e Muradore (2018) combine Velocity Obstacle (VO) and Voronoi Diagrams (VD) in order to perform hybrid path planning.

Previous methods for solving the path planning problem include dynamic programming (XIE; CARRILLO; JIN, 2020) and geometric algorithms, such as A* (TSENG et al., 2014), which operate on the minimization of the computational cost. Algorithms of such nature are heavily based on cost maps that can be a time consuming task, and also can struggle when regarding multiple constraints (ZHAO; ZHENG; LIU, 2018). Similar to other fields in robotics, Computational and Artificial Intelligence methods rise as potential

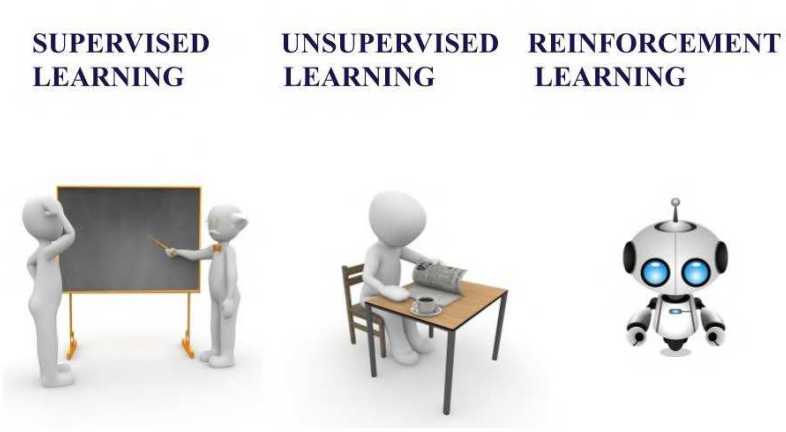


Figure 2 – Machine Learning types: Supervised learning relies on labeled data in order to learn, Unsupervised Learning algorithms aim to find similarities in unlabeled data and Reinforcement Learning works with learning through experience. Source:(MACHINE-LEARNING, 2022)

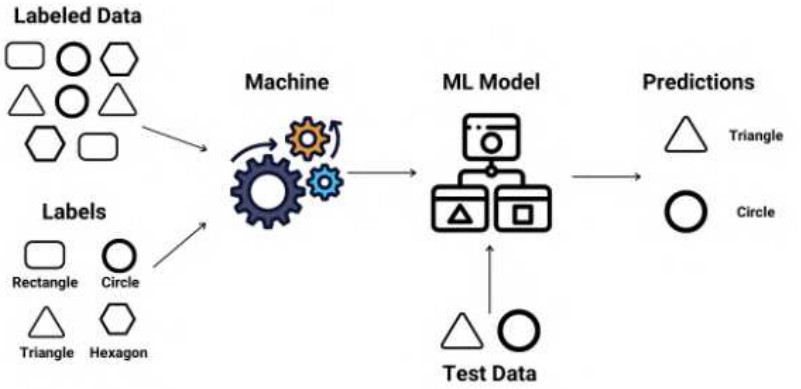


Figure 3 – Supervised Learning illustration, where labeled data is provided to the machine learning model and it learns how to classify them. Source:(SUPERVISED, 2022)

candidates for problem solving, due to its adaptability and generalization capabilities (ARINEZ et al., 2020).

When it comes to Machine Learning techniques, there are three main areas: Supervised, Unsupervised and Reinforcement Learning, depicted in Figure 2. In summary, Supervised Learning will acquired knowledge using labeled data, which is where the "supervised" part of the name comes, because it requires previous annotation of the used data. In the training process, samples in the training data are taken as input in which features are learned via the learning model (NASTESKI, 2017). This type of algorithm is specially used in regression and classification problems with famous techniques such as Artificial Neural Networks (ANN)(WALCZAK, 2019), Support Vector Machines (SVM)(MAMMONE; TURCHI; CRISTIANINI, 2009) and also Deep Learning techniques (LECUN; BENGIO; HINTON, 2015). Figure 3 illustrates the concept of this type of learning.

On the other hand, Unsupervised Learning aims to analyze and cluster unlabeled

datasets. These algorithms discover hidden patterns or data groupings with no need of human intervention, hence "unsupervised learning" (CELEBI; AYDIN, 2016). This type of learning is specially used in clustering, association and dimensionality reduction problems. Some of the most famous approaches are K-mean Clustering (SINAGA; YANG, 2020) and Principal Component Analysis (PCA)(BRO; SMILDE, 2014). Figure 4 illustrates the concept of this type of learning.

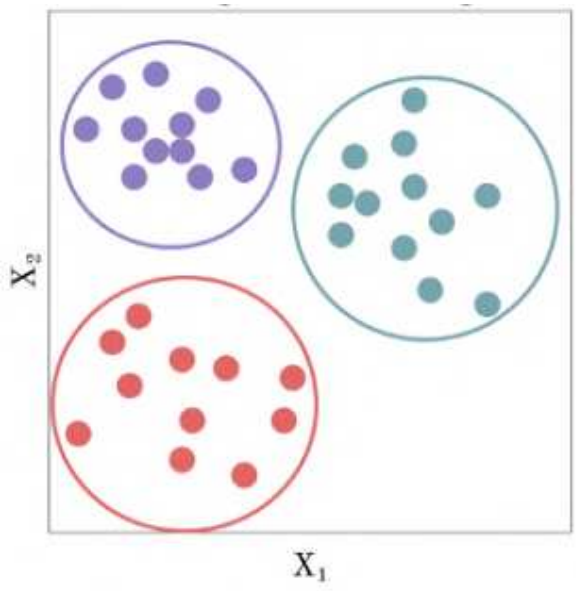


Figure 4 – Unsupervised Learning illustration, where similar data was clustered together. Source: (UNSUPERVISED, 2022).

Reinforcement Learning (RL) is a framework where the agent learns from experience through several environment interactions, in order to maximize the expected cumulative rewards (YAN; XIANG; WANG, 2020). Figure 5 illustrates the concept of this type of learning.

One of the key points of solving a problem modeled as a MDP is the Markov property, that states "The future is independent of the past given the present". This way the agent learns from experience, but takes its actions based solely in its current state. Reinforcement

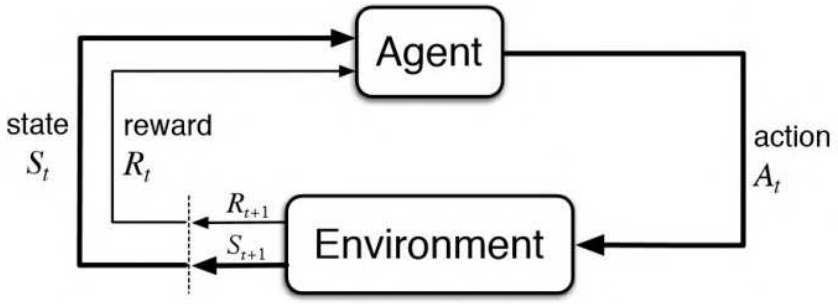


Figure 5 – Reinforcement Learning illustration. Given the agent's state, it takes an action and then it receives a reward for it, and it reaches a new state. This processes repeats itself throughout the training. Source:(**??**)

Learning algorithms aim to find the optimal policy π^*, which will lead the agent to take the best action in any given state (SUTTON; BARTO, 2018). Even though Reinforcement Learning techniques have been applied to many different problems, ranging from playing Atari games (MNIH et al., 2013), energy systems (PERERA; KAMALARUBAN, 2021) to modern robotics (POLYDOROS; NALPANTIDIS, 2017; FALLOOH et al., 2025). There is no consensus in the literature for the best way of modeling the problem as a MDP. In other words, what is defined as the key art of modeling, i.e, the state, action and reward for the systems, vary in each different approach, depending on the nature of the key part, such as being continuous (BOUHAMED et al., 2020) or discrete (IMANBERDIYEV et al., 2016), and as to the number of different states and actions, as it can be seen in (WANG et al., 2020; IMANBERDIYEV et al., 2016).

In the Machine Learning context, there are several decisions when it comes to developing and deploying an algorithm. The process can be divided in four stages: problem understanding, data handling, model building and model monitoring(NASCIMENTO et al., 2019). The first stage involves grasping the complexity of the situation and choosing metrics that will reflect the measurement of the desired outcome. The second stage is all about finding out what data is necessary to meet the problem's requirements as well as the data acquisition, structuring and feature engineering if necessary. In this stage, acquiring the correct data set can be quite challenging, also manually labeling the data can be quite expensive, so there are various scalable techniques that propose using semi-supervised learning, crowd sourcing or weak supervision(WHANG; LEE, 2020). The third stage is where the model structure, training algorithm and other nuances of the development are determined. There are several approaches to building a machine learning model from scratch, and many challenges as well. It is a stage that is highly prone to inherit problems in the previous mentioned steps. The fourth stage involves the model evaluation and improvement using metrics as guiding lines to ensure performance evolution. An approach that is gaining more interest in the research community is the usage of Transfer Learning techniques, where it possible do exploit and correlate knowledge across different domains (ZHUANG et al., 2020).

Transfer learning is a concept that comes initially from educational psychology. As proposed by the psychologist C. H. Judd, learning to transfer is the result of the generalization of experience (ZHUANG et al., 2020), which claims that is possible to realize the transfer from one situation to another, as long as the individual generalizes its experience. This assumes that there is at least a connection between the tasks to be learned. For example, someone who has learned to play the violin can learn how to play the piano faster than others, due to both been musical instruments and sharing many musical theories. Figure 6 shows examples of related tasks in which transfer learning can be applied in real life.

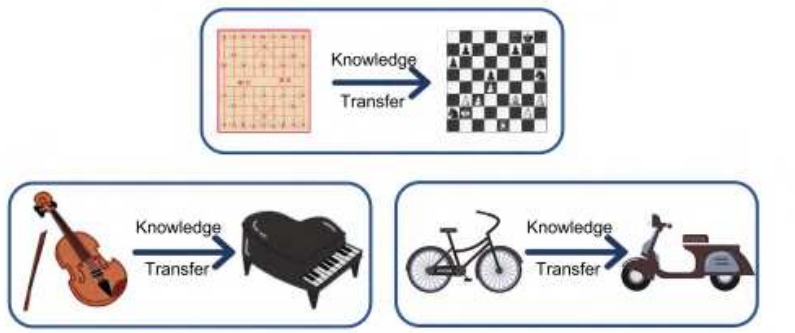


Figure 6 – Transfer Learning illustration, where similar tasks are used as source domain to provide knowledge transfer to different activities. Source:(ZHUANG et al., 2020)

Due to possible discrepancy between task's domains, transfer learning can be divided in two categories: homogeneous and heterogeneous transfer leaning (DAY; KHOSHGOFTAAR, 2017). In the first one it is possible to assume that the differences between the feature spaces of each domain are determined by only marginal distributions, thus this method thus focuses on bridging the gap in the data distributions. Domain adaptation is used for object recognition in (GOPALAN; LI; CHELLAPPA, 2011), which exemplifies one application of homogeneous transfer learning. As this similarity between tasks may not be the case in many scenarios, the second one has the addition of also having to adapt the feature space. This can be observed, for example, in sentiment classification problems (CAO et al., 2021), where some words can have different meaning tendencies for each domain.

As a special form of transfer learning, curriculum learning involves a set of tasks organized in progressively more difficult fashion (SHAO; ZHU; ZHAO, 2018). This strategy has been used in several different fields, such as natural language processing (NLP)(CAUBRIÈRE et al., 2019), image classification (GONG et al., 2016) and in reinforcement learning (LUO; KASAEI; SCHOMAKER, 2020). By exposing the machine learning model to simpler tasks at first, and only after some training that moves to more complex assignments, this strategy can provide two main benefits: an increase of the convergence speed of the training process and also a better accuracy.

1.1 Glossary

For the sake of clarity, some terms used in this work, specially those used in the reinforcement learning context, will be briefly explained here.

- Agent - Learning unit in a reinforcement learning context. The agent is what interacts with the environment and what actually learns throughout training. In many cases

it is a virtual agent, given most learning in reinforcement learning scenarios is done in simulated environments.

- Action - The possible ways the agent can interact with the environment. Can be in the form of low level instructions such as direct control signals, or high level actions such as "Move Forward" or even full instruction scripts such as making the agent follow a pre determined path.

- State - The way that the environment is represented to the agent. Can be represented in the form of direct sensoring information such as laser readings or camera flow data. Also can be designed as higher level features such as obstacle corners.

- Policy - Returns an action for a given state. In other words, is what decides what action the agent should take at a given moment.

- Model - In the context of model-based or model-free approaches to reinforcement learning, model refers to the environment, in respect of being capable of predicting the response of a given action when interacting with the environment. In order to be able to predict this response, one has to have a model of the used setting.

- Value Function - Stands for how good a state or a state-action pair is. In the first case, it is called a State Value Function (V(s)) and in the second one is called Quality Value Function (Q(s,a)), or Q-value in short. State value function represents the sum of expected future rewards for being in state s. On the other hand, Q-value represents the sum of expected future rewards for being in a given state s and taking action a.

- Reward Function - Determines the reward the agent will receive by taking a given action in a given state. This function can be dense, with many features that will yield positive or negative rewards, or sparse, where few instances will lead to a reward, most commonly only the end of the episode.

- Curriculum - Set of progressively more difficult stages to expose the learning agent to easier tasks at first, and, only after some basic knowledge of the environment, present more complex scenarios to it.

- Q-Table - Data structure where the Q-values of each state-action pair. They should be iteratively updated throughout training.

1.2 The Problem and its Impact

Navigation is a crucial part of any robotics application, being with one or multiple agents. In the literature there are several different approaches to solve this issue and,

despite that, new algorithms are still being developed. The specificity of each situation when combined with rise of new applications to suit modern days industrial and research trends, bolster the claim that there is still room and demand for more solutions in this field.

Machine Learning algorithms such as reinforcement learning has been seen as a promising approach to tackle many different problems and navigation is one of them. It can greatly benefit from its capacity to adapt and learn from experience. Given this, having a clear understanding and capacity to implement these algorithms is paramount to be able to propose innovative solutions to this well known problem.

This thesis combines machine learning techniques such as reinforcement learning, as well as other methodologies to enhance the learning experience and improve results, such as Transfer Learning and Curriculum learning, to propose algorithms for single and multi agent navigation scenarios, with the latter being focused on the scenarios of intelligent logistics, such as using robotic agents to organize or fetch and retrieve objects in warehouses, libraries or commerce. Additionally, this work examines the impact of state representation in mapless navigation and provide an initial study on how various hyperparameters influence one of the proposed global path planning algorithms.

Therefore reliable navigation tools can be obtained from machine learning algorithms, as will be demonstrated in the following chapters.

1.3 Objectives

The general purpose of this work is to investigate and propose reinforcement learning-based solutions for both single and multi-agent navigation scenarios. This research aims to address the central question that guides this work: Is it possible to use reinforcement learning to generate viable routes for the robot navigation problem in structured and semi-structured environments? The proposed algorithms will be tested and validated in different contexts, including deliberative and reactive navigation, mapless navigation for single agents, and multi-agent path planning for logistics applications.

Specifically speaking, this work proposes to:

1. Develop a Global Path Planning algorithm capable of accounting for different sets of priorities;
2. Develop a based Local Path Planning algorithm to be applied in dynamic environments;
3. Develop a Path Planning algorithm for intelligent logistic applications such as in warehouse scenarios;

4. Analyze and discuss the effects of different state representations using depth sensors in mapless navigation.

These specific objectives are order in such a way that the knowledge obtained to achieve the first one will facilitate the develop the next algorithms. All of the proposed objectives have Reinforcement Learning based approaches and may use of enhancing techniques such as Transfer and Curriculum learning to improve the algorithms performance when facing the challenges that are inherent to the situation.

1.4 Related Works

1.4.1 Path Planning Strategies

Path planning problems are gaining an increasing importance in robotics. This problem is simply the calculation of a path free of collision between a starting point and a goal in order for a robot to move in a environment (TAN; MOHD-MOKHTAR; ARSHAD, 2021). Several strategies have been proposed to autonomously guide robots in static environments. Some of them, related to this work, are described below.

In Computational Intelligence context, deterministic and heuristic algorithms to solve the path planning problem are constantly used. Lamini, Benhlima e Elbekri (2018) use the famous meta-heuristic Genetic Algorithm (GA) to solve this problem in static scenarios. The authors propose a fitness function that penalizes turns and obstacle proximity. They also contribute with a new crossover method that avoids premature convergences, offering new viable routes to following generations with increased fitness values, thus improving convergence speed.

Dijkstra's algorithm (BARBEHENN, 1998), $A*$ (VASCONCELOS; BRANDÃO; SARCINELLI-FILHO, 2020), and $D*$ (FERGUSON; STENTZ, 2005) are some of the widely used search-based algorithms for path planning in fully or partially mapped environments. In most cases, real-time constraints are not met with search-based approaches due to high computational demands and memory footprints for state space representation (KULATHUNGA, 2021).

In order to solve the path planning or trajectory problem, many strategies based on Dijkstra's method have been implemented. In (FUSIC; RAMKUMAR; HARIHARAN, 2018) the authors modified parameters in Dijkstra algorithm in order a robot could find an appropriate path to reach the destination. The reduction method was effective in terms of time and velocity in the created environments for robot path planning.

In a similar way, A* can be used as a type of path planning strategy, which is applicable in situations in which the global environmental information is already known.

In order to solve this problem in dynamic environments, Vasconcelos, Brandão e Sarcinelli-Filho (2020) proposed the $LPA*$, a replanning method that is an incremental version of $A*$ algorithm for 2D single-shot grid-based path finding. The results show that the proposed method is able to guide the mobile robot to its target in a changing environment. When obstacles are interactively included in the scene, the route is redesigned and the seeking for a new optimal solution connecting the current robot position and its target position is computed and established.

Regarding reinforcement learning, several areas aim to develop algorithms to impact its performance. State and action space sizes can severely impact the any reinforcement learning algorithm overall performance or convergence rate. Maoudj e Hentout (2020), focus on this subject and proposes a new state-action representation. To generate the state space, the authors first fit each obstacle in the grid on a rectangular shaped box. After that, these boxes are used to generate escape points near each of its vertices and the action space consists in moving in a straight line from the current spot to one of these escape points or to the destination itself. This greatly reduces to traditional state space size in this kind of problem. The authors also contribute with a new reward function to initialize the Q-table, to offer a prior knowledge about the environment, improving the convergence speed of their algorithm.

Similarly, Yan e Xiang (2018) also use prior knowledge of the environment to develop a Q-Table initialization method. The Euclidean distance from the grid cell to the destination goal is used as base for this process, which enhances the effectiveness of the path design procedure. When compared to other algorithms simulations demonstrated superior performance, taking path length and safety as priorities.

Tai e Liu (2016) propose the use of feature maps obtained by the depth sensor as the agent's state and apply Deep Q-Learning (DQN) in order to achieve safe exploration navigation. One of the downfalls of the DQN algorithm is that it often overestimates the Q-values in early stages of training. With the intention of minimizing this issue, Xue et al. (2019) propose a Double DQN structure, although it uses the positions of the agent and obstacles as state, instead of depth images.

A great challenge in reinforcement learning techniques is the trade-off between exploration and exploitation as well as the reward shaping process. Chakraborty e Banerjee (2013) coined a new approach to RL called Advanced Q-Learning, applying a fuzzyfication process to the reward function making it being represented by a Gaussian membership function. Explore/exploit ratio is dealt with by separating the reward function in two components, the first one being positive that stimulates the agent to move towards its destination and the second one being a negative, repulsing the agent from obstacles.

In order to solve the problems mentioned above and make the employment of reinforcement learning in real robotic applications viable, Akrour et al. (2018), Arai

et al. (2018) reduced the dimensionality curse impact and enhanced the RL algorithm convergence and generalization with smart approaches to discretize the state and action spaces. In a similar way, Lim, Ha e Choi (2020) proposed a new function approximation to ease the dimensionality problem, besides helping to predict the reward function.

1.4.2 Machine Learning applied to Intelligent Logistics Scenarios

When it comes to intelligent logistic solutions, they face several challenges such as coordinating multiple agents as well as guaranteeing safe and efficient navigation. Various techniques have been employed to solve them. Chen, Li e Liu (2019) use the Windowed Hierarchical Cooperative A*, which converts the environment map into a 3D space-time map, and the path of the robot is planned according to predetermined rules, limiting the search space to a fixed time window to improve convergence time. The authors also take in account a turning factor to find paths with fewer turns to reduce the agent's travel time.

Tsang et al. (2018) use a genetic algorithm based strategy to allocate the agent's tasks efficiently and modifies the Artificial Potential Field (APF) algorithm, where two operations are proposed in other to deal with the APF intrinsic local minima problem. One of the proposed interventions, is to increase the potntial in the location of the agent while reducing in its neighbors.

Machine learning algorithms such as reinforcement learning are also used in the literature in the context of path planning in intelligent logistics scenarios. Yao et al. (2020) propose another variation of the traditional APF path planning method, but employ curriculum learning in tandem with reinforcement learning to deal with the local minima problem. The algorithm rewards the agent for changing the current potential field. Also the agents are trained in increasingly more complex scenarios to increase performance. In the same context, Ren e Huang (2021), use APF to guide the search space of a deep deterministic Policy Gradient algorithm, generating more quality training data.

In the context of an intelligent logistics application for several agents, Kamoshida e Kazama (2017) combine action abstraction with deep reinforcement learning applying it to a 10 times 21 map. The shortest distance path, the shortest trip time path, or the congestion avoidance path to the destination are the three high-level operations available as the RL algorithm's action outputs. The authors do this by representing the agent's state as a 940-dimension vector, which includes the potential action courses and the nodes that other agents would use in their routes. Even though the proposed algorithm chooses between high-level actions, it is assumed that those paths are calculated by other algorithms, which can hinder processing time.

Li et al. (2019) use deep Q-Learning to coordinate up to four agents to deliver objects in a warehouse. Their performance is benchmarked against a simple shortest path

approach. Their algorithm focus on dispatching an available agent to perform the closest and safest tasks.

In a pure machine learning perspective, the paper (LEE; JEONG, 2021) applies a variation of Q-Learning to perform path planning in a warehouse scenario for a single agent. A key factor for optimization their results is the usage of a dynamic reward adjustment for based on the agent's action similar to a local search.

It is possible to see in this section that the field of machine learning techniques applied to intelligent logistics solutions still has many room for development, as there are still many unsolved challenges.

1.5 Contributions

The contributions done during this PhD program are listed bellow. There were a total of eleven written papers. One of them has just being accepted and its on the final steps for publication.

Two of the papers are in the field of Human-Robot Interaction with communication done by gestures. They are not directly related to the main objective of this work, but techniques used for the communication algorithm were insightful in respect to feature representation and dimension reduction and the experiment part of the second paper helped to improve knowledge necessary to deploy control algorithms to validate the future algorithms in real life experiments. Both papers use a neural network based gesture recognition system which required.

The first one proposes a gesture recognition algorithm using neural networks. It uses skeleton information extracted by a depth sensor and using a trigger to begin capturing the movement, guaranteeing all gestures have the same execution time, and after clever dimension reduction via eigen values decomposition, the resulting system requires a tiny dataset, with only 5 samples of each of the nine classes, to work. The system is validated using a playful experiment with educational purposes as motivation.

The second one enhances the system in order to not required a trigger to signalize the system to capture the following movement. New action classes are proposed to be used in teleoperation of a heterogeneous robot team. 14 gesture classes are proposed and real world experiments are done to validate the intended application.

The third, fourth, fifth, sixth, and seventh papers are directly tied to one of the main objectives of this work. They focus on developing a planning algorithm capable of guiding an agent to its goal from any position on the map, while considering three distinct priorities during the planning process. The journey began with the creation of an offline 2D global planning algorithm using Q-Learning (third paper). This approach

was then extended to 3D environments, still leveraging Q-Learning (fourth paper). Next, the algorithm was improved by adopting Deep Q-Learning to better handle 3D scenarios (fifth paper). Building on this, it was adapted for local planning to enable navigation in dynamic environments using Q-Learning (sixth paper). Finally, the algorithm reached its most advanced version, which was tested in both simulated and real-world scenarios (seventh paper).

The eighth paper is a survey on machine learning techniques used for UAV navigation, where approaches such as neural networks and reinforcement learning are discussed showing the benefits of each one and the most common settings that each of them are used as well as the current trends in the literature.

Finally, the last three published papers are advances on the path planning problem in a multi-agent/single-agent logistic applications in a warehouse and/or library settings. The solutions were developed using reinforcement learning techniques as well as other machine learning approaches such as transfer learning and curriculum learning to improve convergence rates and overall performance.

- de Carvalho, K. B., Thinassi, V.B, Brandão, A. S. "Action recognition for educational proposals applying concepts of Social Assistive Robotics." published in Cognitive Systems Research (2022) (CARVALHO; BASÍLIO; BRANDÃO, 2022);
- de Carvalho, K. B., Villa, D. K., Sarcinelli-Filho, M., Brandão, A. S. "Gestures-teleoperation of a heterogeneous multi-robot system." published in The International Journal of Advanced Manufacturing Technology (2022) (CARVALHO et al., 2022b);
- de Carvalho, K. B., de Oliveira, I. R. L., Villa, D. K., Caldeira, A. G., Sarcinelli-Filho, M., Brandão, A. S. "Q-learning based Path Planning Method for UAVs using Priority Shifting." published in International Conference on Unmanned Aircraft Systems (ICUAS). IEEE, 2022 (CARVALHO et al., 2022a);
- de Carvalho, K. B., de Oliveira, I. R. L., Batista, H. de O. B., Brandão, A. S. " An offline Q-Learning 3D path planning algorithm for UAV with priority shifting rewards." 2022 published in Latin America Robotics Symposium (LARS), 2022 (CARVALHO et al., 2022);
- de Carvalho, K. B., de Oliveira, I. R. L., Brandão, A. S. "AV Navigation in 3D Urban Environments with Curriculum-based Deep Reinforcement Learning" 2023 published in International Conference on Unmanned Aircraft Systems (ICUAS). IEEE, 2023 (CARVALHO; OLIVEIRA; BRANDÃO, 2023);
- de Carvalho, K. B., Batista, H. de O. B., Brandão, A. S. "Q-Learning based Local Path Planning for UAVs with Different Priorities". Published in Latin American Robotics Symposium (LARS), 2023 (CARVALHO et al., 2023);

- de Oliveira, I. R. L., Carvalho, K. B., Batista, H. de O. B., Brandão, A. S. " Curriculum-based Reinforcement Learning for an Effective Multi-Agent Path Planning Algorithm in Warehouse Scenarios" published in the 15th IEEE International Conference on Industry Applications (INDUSCON),2023 (OLIVEIRA; CARVALHO; BRANDÃO, 2023);

- Fagundes A., L., de Carvalho, K. B., Ferreira, R. S, Sarcinelli-Filho, M., Brandão, A. S." Machine Learning for Unmanned Aerial Vehicles Navigation: An Overview". Published in SN Computer Science, 2024 (FAGUNDES-JUNIOR et al., 2024);

- Batista, H. de O. B., Hudson, T. M., de Carvalho, K. B., Brandão, A. S. "Q-Learning-Based Multi-Objective Global Path Planning for UGV Navigation" 2024 published in Latin American Robotics Symposium (LARS), 2024 (BATISTA et al., 2024);

- Batista, H. de O. B., Hudson, T. M., de Carvalho, K. B., Brandão, A. S. "Multi-Goal Robot Path Planning Based on Q- Learning for Library Logistics" 2024 published in Congresso Brasileiro de Automática, 2024 (BATISTA et al.,);

- de Carvalho, K. B., Batista, H. de O. B., Fagundes A., L., de Oliveira, I. R. L., Brandão, A. S. " Q-Learning Global Path Planning for UAV Navigation with Pondered Priorities" accepted to Intelligent Systems with Applications (2025) (CARVALHO et al., 2025).

1.6 Work Structure

This work is split in chapters and an appendix with the following contents:

1. Chapter 2: Lays out the theoretical background for the approaches employed in this work, such as reinforcement learning, transfer learning, and curriculum learning, is laid out in this chapter;;

2. Chapter 3: The methodology guiding the contributions of this work is described in this chapter. For each case, the MDP modeling, system training and other specific details are given here;

3. Chapter 4 presents and discusses the results for the contributions done in some of the published and submitted papers (CARVALHO et al., 2022a; CARVALHO et al., 2022; CARVALHO; OLIVEIRA; BRANDÃO, 2023; CARVALHO et al., 2023; OLIVEIRA; CARVALHO; BRANDÃO, 2023; CARVALHO et al., 2025);

4. Chapter 5 highlights the concluding remarks;

5. Appendix A presents a hyperparameter and Q-table initialization discussion on the context of single agent global planning. This chapter can provide valuable insights for the effects of explore/exploit in different stages of training, as well as learning rate and different Q-table initialization methods. Given that the obtained results are specific to one of the contributions of this work, the same parameters can lead to different results in other contexts. However, the provided discussion can lead to valuable insights to readers that might be interest in developing or modeling different problems using Reinforcement Learning.

2 Theoretical Background

2.1 Reinforcement Learning

Autonomous and intelligent navigation is still a topic of interest in the research community. In order to address this problem, it is necessary to have sophisticated high level control methods that can learn and adapt themselves to the environment conditions.

Reinforcement Learning is a machine learning framework in which the agent learns from its experience interacting with the environment. This technique models the situation as a Markov Decision Process (MDP), which can be defined as a 4-tuple (S_t, a_t, p, r_t), where s_t is the state the agent is in at time t, a_t is the action it takes at time t, p is the state transition probability, which is the probability of reaching a state given the agent took action a_t in a given state s_t, and r_t is the reward the environment gives the agent for reaching state s_t. One of the pillars of solving problems modeling them as MDPs is the Markov property, that states "The future is independent of the past, given the present". This way the agent learns from experience, but chooses its actions solely based on its current state. The goal of a reinforcement learning algorithm is to find the optimal policy π that leads the agent to choose the optimal action for any current state (SUTTON; BARTO, 2018). In robotics, an MDP can take different forms depending on how each of its components is modeled. The state represents what the agent perceives from its environment, which can range from a grid cell index to LiDAR or other sensor readings, impacting the system's overall complexity. Actions define how the agent interacts with the world and can be either discrete or continuous. In the discrete case, movement primitives like moving forward, backward, or turning left and right are common. In the continuous case, actions may involve direct control signals or velocity and acceleration references(FAGUNDES-JUNIOR et al., 2024).

One of the main hurdles involving reinforcement learning is the explore/exploit dilemma, which involves the trade off between choosing the best action known so far and exploiting its returns, or choosing another action and explore the possibilities it can provide the agent and potentially yielding higher returns.

To resolve this issue, some approaches are seen in the literature, such as:

- **Epsilon-Greedy**: this approach deals with the issue by having a hyper parameter ϵ, which can be static or dynamic, and exploiting the best action $100 - \epsilon$ percent of the times and choosing a random action ϵ percent of time, thus exploring new possibilities (WUNDER; LITTMAN; BABES, 2010). This is the most used approach to deal with this dilemma in the literature.

- **Optimistic Initial Value**: the rewards for each action are initialized with a high value, and the agent always prioritizes the action greedily. Throughout training, these values will be updated and come closer to their real value. This will force the agent to choose different actions due to the initial optimistic values (SHOJAEE; MASHHADI, 2017).
- **Upper Confidence Bound**: an upper bound gets smaller the more times a given action is accessed, improving the accuracy of the estimated reward. By increasing the weight of estimated rewards that were rarely accessed due to this upper bound, this balances the relationship between exploring new stocks and holding on to actions with high estimated rewards (GARIVIER; MOULINES, 2011).

In the context of motion planning algorithms, in reinforcement learning will strive to find the optimal policy, which will lead it to choose actions in which it will maximize the sum of expected future rewards. This can be seen as a process of trial evaluation. Firstly, an action is chosen given the current policy π, which leads the agent to the next state. The state value function

$$V_\pi(s) = \mathbb{E}_\pi(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s), \tag{2.1}$$

where $\gamma \in [0, 1]$ is the discount factor, evaluates how good a given state s is, by returning the sum of expected future rewards for being in state s(CUNHA et al., 2018).

The value function can be split to take in account the action a, in the current state s. This evaluates the *Quality*, of a given action in a given state, leading to the Q-value for the pair defined

$$Q_\pi(s, a) = \mathbb{E}_\pi(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s, A_t = a). \tag{2.2}$$

which stands for the sum of expected future rewards for taking action a in state s (NGUYEN; LA, 2019).

The Value Function can be obtained by using the policy π and the Q- values for each state-action pair as

$$V(s) = \sum_{a \in A} \pi(a|s) q_\pi(s, a) \tag{2.3}$$

Optimal policy π^* can be extracted of the Q-Value function

$$\pi^*(a|s) = \begin{cases} 1, & a = argmax_{a \in A} Q_*(s, a) \\ 0, & otherwise \end{cases}, \tag{2.4}$$

In the reinforcement learning field, there are many different types of algorithms, a important distinction is of model-based and model-free algorithms. The first category is not commonly applied to navigation problems given that it is required a partial or full model of the environment, meaning that state transition probabilities for each action-state pair must be known. When it comes do model-free approaches, although it is usually less sample efficient than its counterpart (BRUNNBAUER et al., 2021), it does not require a model of the environment, being more popular in the type of application proposed in this work.

The model-free RL applied to the motion planning process does not need to estimate the MDP model, and the value function or policy function can be evaluated directly by sampling to approximate the solution of reinforcement learning tasks (HUANG, 2020).

When it comes to model-free algorithms, there are several different approaches that can be mainly grouped in three categories: valued-based, policy-based and actor critic approaches. They relate to each other according to Figure 7.

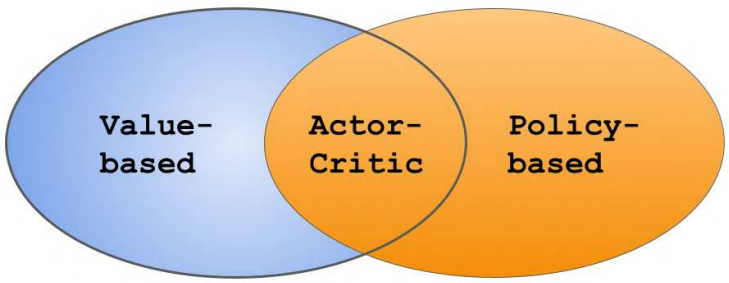


Figure 7 – Types of RL algorithms. Value-based will learn by optimizing a value function while policy-based use policy gradient methods to modify directly the policy. Actor-critic is the intersection of the other two approaches.

Value-based reinforcement learning works by maximizing value function. The value function of each state of agent can be obtained by estimation. Commonly used algorithms include Monte Carlo (MC) (FU, 2019), TD (λ) (HASSELT; MAHMOOD; SUTTON, 2014), Q-learning (DAS et al., 2012), SARSA (HARWIN; SUPRIYA, 2019). The most well known and widely used of them is Q-Learning, a model-free and off-policy algorithm. Being model-free it does not require a model of the environment to learn, and by being off-policy it does not necessarily use the best action of the current policy to explore.

In Q-learning, the state-action quality function, *Q(s,a)*, is used as evaluation tool and its value is iteratevely estimated throughout training, with update defined by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a)), \tag{2.5}$$

where r_t is the current reward, α is the learning rate, and γ is the discount factor.

Tabular methods such as Q-learning may suffer from the dimensionality curse, which means it may struggle in environments with huge number of states and/or action spaces. Therefore, many approaches have been done in the literature in order to improve learning efficiency. Jaradat, Al-Rousan e Quadan (2011) established a new state space to limit the number of state inputs, so as to greatly reduce the size of the Q-table, thus improving the speed of the planning algorithm; (SONG et al., 2012) based on the known environmental information and Q-table, a mapping is created between the initial values of the value table. The dynamic wave expansion neural network is used to specify the initial Q-value properly. To increase learning efficiency and produce better strategies, prior knowledge of the environment is integrated into the learning system.

In the context of robot motion planning, Das et al. (2010) improve learning efficiency for motion planning by filtering what Q-values need to be updated using simple rules. Based on the known topological map, Romero-Martí et al. (2016) use a clever way of reducing a map to more simple state representation by representing doorways and room centers as nodes and validate their proposal with real world experiments, even though sensor information is obtained via odometry, subject to error accumulation and inaccurate sensor information.

Reinforcement learning based on value function is also applied in the context of UAV flight control system. Khamidehi e Sousa (2020) proposed to use an updated version of Q-learning, which tackles a common issue in the classic algorithm: the over estimation of state-action Q-values. By having two Q-tables, one chooses which action to use and the other one to obtain the Q-value used to calculate the TD error.

Policy-based algorithms differ from the previously presented ones in the aspect that instead of learning by updating value-functions, it directly optimizes the policy π via policy gradient strategies (XU et al., 2020; JIA et al., 2020). Value-based approaches can be seen as actor algorithms, while policy gradient are the critic. A policy gradient method is generally obtained by parameterizing the policy π in a parameter vector. Assuming that the parameterization is differentiable in respect to this vector, a gradient can be calculated and used to optimize the policy (GRONDMAN et al., 2012). Given that the gradient is obtainable, standard optimization techniques can be used to find a locally optimal solution to the cost function. Graf e Platzner (2015) apply a policy gradient approach in the context of Computer Go and obtained better results than classical Monte-Carlo-Tree-Search approaches.

On the other hand actor-critic methods aim leverage the advantages of both value-based approach policy gradient ones. Actor-only methods provide the ability to provide continuous actions, but also produce a large variance which can be countered by adding a critic to the system. The critic's role is to evaluate the current policy defined by the

actor. The critic approximates and updates the value function using samples, which is then used to update the actor's policy towards improving performance. This usually preserve convergence properties of policy gradient algorithms (GRONDMAN et al., 2012). As described in (BAIRD; MOORE, 1998), in actor-critic methods, the policy is not directly obtained from the value function, instead the policy is updated in the policy gradient direction using a small step, meaning that a change in the value function will not result in a big difference in the policy at once, diminishing the oscillatory behavior that may be seen in actor-only approaches.

2.2 Transfer Learning

Transfer learning aims at improving the performance of target learners on target domains by transferring the knowledge obtained in different, but related domains. This technique has been demonstrated to be effective in many real-world applications exploiting this labeled knowledge of a source domain and enhancing the model's performance in a new related task even if there is little to no labeled data on it (DAY; KHOSHGOFTAAR, 2017). This technique can be applied in many fields of Machine Learning overall.

A domain $\mathcal{D}$ is defined by two items, a feature space $\mathcal{X}$ and a marginal probability distribution P(X), where $X = \{x_1, \ldots, x_n\} \in \mathcal{X}$. In other words, $\mathcal{D} = \{\mathcal{X}, P(X)\}$. A task $\mathcal{T}$ consists of a label space $\mathcal{Y}$ and a decision function f, that is, $\mathcal{T} = \{\mathcal{Y}, f\}$, the decision function is a implicit one and is expected to be learned from sampled data (PAN; YANG, 2009).

$\mathcal{D}_S$ is now defined as the source domain data where it has the form $\mathcal{D}_S = \{(x_{S1}, y_{S1})..., (x_{Sn}, y_{Sn})\}$, where x_{Si} is the ith data instance of $\mathcal{D}_S$ and y_{Si} is the associated class label for x_{Si}. Similarly, $\mathcal{D}_T$ is defined as the target domain data where $D_T = \{(x_{T1}, y_{T1})..., (x_{Tn}, y_{Tn})\}$ and where $x_{T1} \in \mathcal{X}_T$, is the ith data instance of $\mathcal{D}_T$ and $y_{Ti} \in \mathcal{Y}_T$ is the associated class label for x_{Ti}. The source predictive function is denoted as $f_S(.)$, the target predictive function as $f_T(.)$, and the target task is denoted as $\mathcal{T}_T$ and the source task as $\mathcal{T}_S$ (ZHUANG et al., 2020).

Transfer learning aims to improve the target predictive function $f_T(.)$ given a source domain $\mathcal{D}_S$ with a corresponding task $\mathcal{T}_S$, and a target domain $\mathcal{D}_T$ with a corresponding task $\mathcal{T}_T$. It is worth noting that the source domain here defined may be multiple source domains. Domains can be different for two distinct reasons. First, their feature space may differ, hence $\mathcal{X}_T \neq \mathcal{X}_S$, or that marginal distributions differ, hence $P(X_S) \neq P(X_T)$ (WEISS; KHOSHGOFTAAR; WANG, 2016). Shimodaira (2000) proved that when the latter case happens, the learner using transfer learning from a source domain will not perform optimally. For clarity sake, take a document classification problem as an example. If one has two different sets of document classification as source and task domains, and

they are written in different languages, this is a example of the source and target domain's feature vectors being different from each other. If the documents focus on different topics, then their marginal probability distribution is different.

This leads to two different types of Transfer Learning, homogeneous and heterogeneous. In homogeneous transfer learning, the feature spaces of the data in the source and target domains are represented by the same attributes ($\mathcal{X}_S = \mathcal{X}_T$) and labels as well ($Y_s = Y_t$), while space itself has the same dimension. This approach focuses in diminishing the difference in data distributions between the domains via cross-domain transfer (WEISS; KHOSHGOFTAAR; WANG, 2016). Generally, as described in (WEISS; KHOSHGOFTAAR; WANG, 2016; PAN, 2020), homogeneous transfer learning solutions can be separated in five categories: Instance-based, feature-based (symmetric or asymmetric), model-parameter-based, relational-informational-based and hybrid-based approaches, which are discussed in depth in (WEISS; KHOSHGOFTAAR; WANG, 2016). It is worth highlighting that there might be marginal differences in domains in homogeneous transfer learning. In these cases, there is some work in domain adaptation, with the goal of reducing the accuracy drop due to distribution shift.

As for heterogeneous transfer learning, the source and task domain have different feature spaces and labels ($\mathcal{X}_S \neq \mathcal{X}_T$) and ($Y_s \neq Y_t$). This method requires further work before knowledge transfer properly occurs, such as feature and/or label space transformations as well as dealing with cross-domain data distribution differences. In other words, one can transfer knowledge between domains, but data is represented differently from source to target domain. Most heterogeneous transformation techniques fall in two categories: symmetric and asymmetric transformation. The first case takes both source and target feature space and learns proper feature transformation to project them into a common subspace $\mathcal{X}_C$ for adaptation purposes. This allows the individual feature spaces to share a common feature representation and then traditional Machine Learning technique may be applied. Asymmetric transformation on the other hand, transforms the source feature space to align with the target domain reducing the problem to a homogeneous transformation (DAY; KHOSHGOFTAAR, 2017). Both methods are illustrated on Figure 8.

In a context more directly related this work, in a path planning problem, if an agent has its state space defined as the possible cells in a grid and it learns how to navigate in a environment, changing the obstacles position and using the first task as source domain, it would be a homogeneous transfer case, given that the feature space is similar in both cases and the marginal probability distribution for each state is different. If, for the second map, the state representation changed to contain the agent's cell number as well as the obstacles', this would be a heterogeneous transfer learning scenario, given that not only the feature vector differs, but its very dimensionality also is different.

A very popular application of Transfer Learning is to use previously trained deep

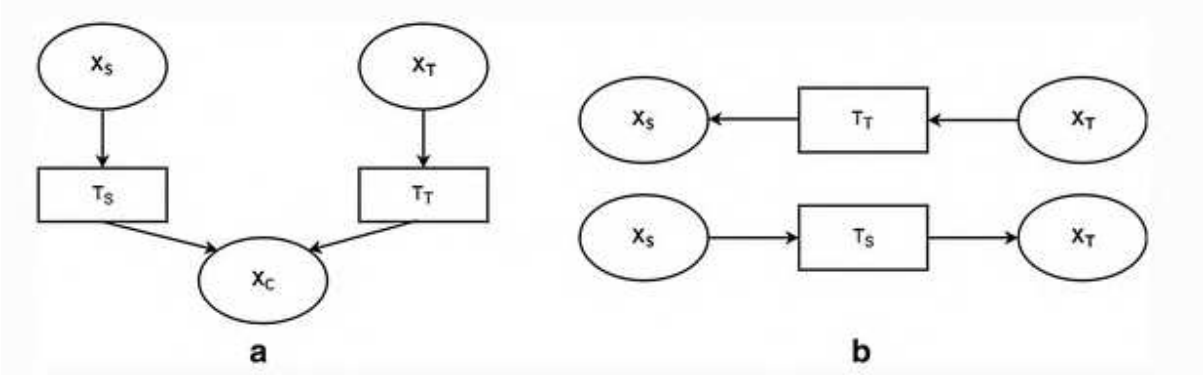


Figure 8 – Symmetric (a) and Assymetric (b) Transformations in Heterogeneous Transfer Learning. The first one applies a transformation to both source and target feature spaces to find a common representation. The second applies different transformations on the source domain to align it with the target domain representation. Source:(DAY; KHOSHGOFTAAR, 2017).

learning models for image classification such as AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2017), Resnet (HE et al., 2016) and many others. These networks are trained using data sets with millions of images, which leads to a very computational costly process. In convolutional neural networks, the layers are capable of progressively more sophisticated abstraction levels, identifying edges, shapes and such. So, instead of training a model from scratch, many papers use a pretrained network and replace their final layer(s) and retrain them, allowing to leverage the feature extraction and abstraction learned in the earlier layers and while tailoring the network to the current application. This has been used in several different field such as ear recognition (ALMISREB; JAMIL; DIN, 2018), orientation detection (NAGATA et al., 2020), pathological brain detection (LU; LU; ZHANG, 2019), facial detection (SEKARAN; LEE; LIM, 2021) as well as robotics (ABBAS et al., 2020).

When it comes to transfer learning applied directly in reinforcement learning, the knowledge transfer can be seen as low-level knowledge such as a tuple instance with precious stage, action, reward an next stage, (s, a, r, s'), a action-value function, Q, a policy, π, a full task model, or as higher level knowledge such as what action to use in some situation (providing as subset of the full action list as options), partial policies, rules, important features for learning or shaping rewards (TAYLOR; STONE, 2009).

In order to evaluate the effectiveness of the transferred knowledge, there are several reinforcement learning specific metrics such as seen in (ZHU; LIN; ZHOU, 2020) and described bellow:

- **Jumpstart performance**: the initial performance of the agent;
- **Asymptotic performance**: the ultimate performance of the agent;
- **Accumulated rewards**: the area under the learning curve of the agent

- **Transfer ratio**: the ratio between the asymptotic performance of the agent using transfer learning compared to the agent not using it;
- **Time to threshold**: number of iterations until needed to reach a preset performance threshold
- **Performance with fixed training epoch**s: the performance achieved after a preset number of episodes
- **Performance sensitivity**: variance in performance using different hyper-parameter settings such as learning rate, $\epsilon - greedy$ values an such.

Figure 9, illustrates some of the metrics.

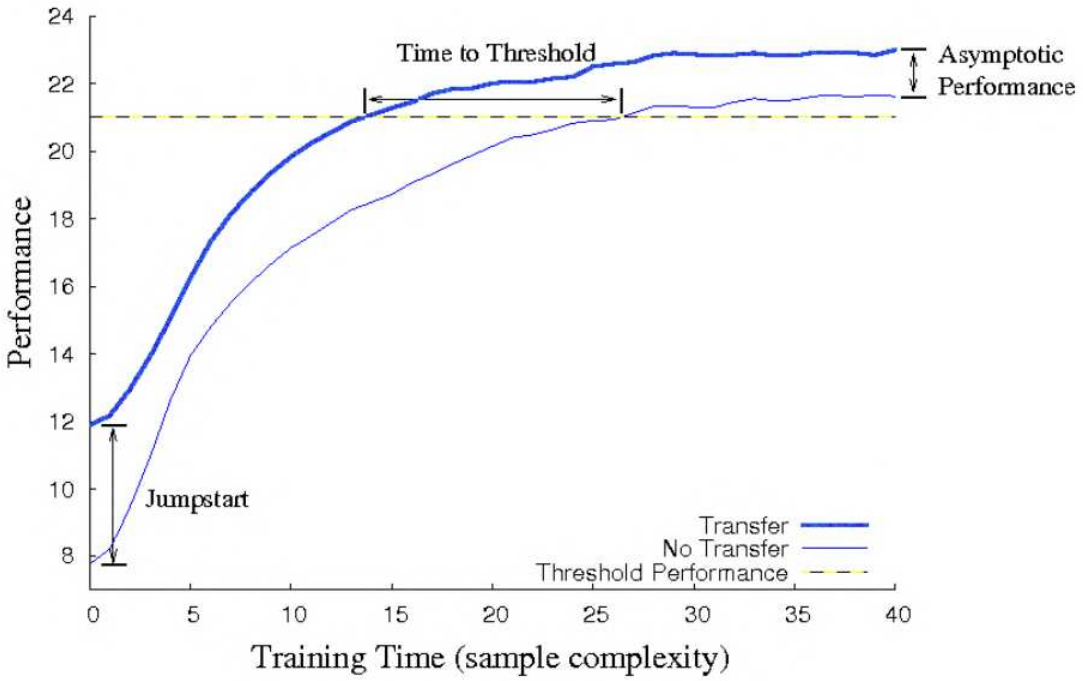


Figure 9 – Transfer Learning Metrics Illustration (TAYLOR; STONE, 2009).

A special type of transfer learning that has gaining interest in the scientific community is Curriculum Learning. This is inspired in human learning, which is frequently supervised, but also is often accompanied by a curriculum (WEINSHALL; COHEN; AMIR, 2018). When humans teach each other, the order of the contents is not randomly chosen. A task may be divided by the teacher in smaller sub-tasks, often called shaping (KRUEGER; DAYAN, 2009). A well proposed curriculum can improve convergence speed and overall performance of a model.

Specially in the context of reinforcement learning and Markov Decision Process, Narvekar et al. (2020) highlight 3 key factors:

- **Task Generation**: The quality of the curriculum is directly related to the quality of the tasks available to choose from. This key factor is the process of creating a good set of sub tasks from which to obtain experience samples. These tasks can be pre determined or dynamically generated during curriculum constructing by observing the learning agent.

- **Sequencing**: Deciding how to partially order the set of intermediate tasks is as a key challenge as choosing the set itself. Usually this is manually defined by a human supervisor,but automated curriculum sequencing have been more explored.

- **Transfer Learning**: The tasks contained in the curriculum may differ in state/action space, reward function or even transition function from each other and specially from the final task. Therefore the way that knowledge is transferred from one stage to the other may not be trivial and must be examined.

Figure 10 shows examples of curricula structures.

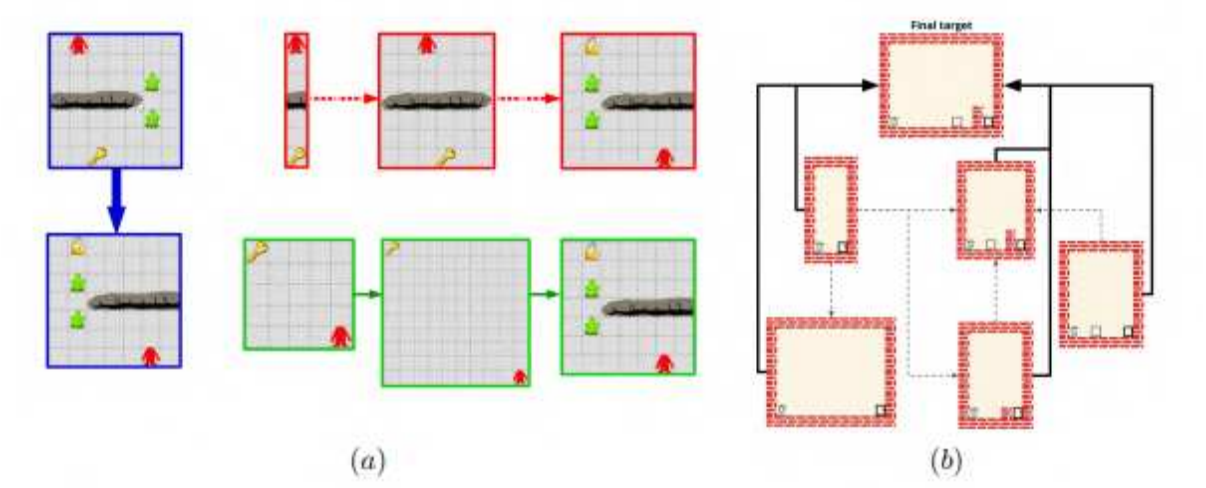


Figure 10 – (a) Three examples of linear sequences in gridworld scenarios (b) Directed acyclic graphs in block dude, where there is more than one stage option for each stage in the curriculum. Source: (NARVEKAR et al., 2020).

In attempt to automatically generate the curriculum structure, Florensa et al. (2017) progressively initiate the learning agent further away from the predetermined destination, leading to a steady flow of more difficult tasks (longer path to goal) throughout training. In the same context, Luo, Kasaei e Schomaker (2020) use automatic generation of the curriculum in a robotic manipulator agent by employing a more loose accuracy requirement to consider a success in earlier stages, leading the agent to obtain more rewards for simple skills at first before being more severe with accuracy standards.

There is still room for improvement in the Transfer Learning field, such as better curriculum design, metrics or even automated episode generation. Also there are still

challenges when it comes to integrating these techniques in reinforcement learning scenarios. This work proposes to contribute in this field as well.

3 Methodology

This chapter is divided into five sections. The first covers three research papers on global path planning. The first paper models the problem as a 2D navigation scenario, the second extends the approach to 3D grids, and the third validates the 2D path planner with real-world experiments. The second section explores more advanced techniques, such as Deep Q-Learning and Curriculum Learning, to address global path planning. The third section focuses on adapting the algorithm for local path planning. The fourth details the implementation of a path planning algorithm for multiple robots in a logistics warehouse. Finally, the last section examines the impact of state representation on mapless navigation using Double Deep Q-Learning.

3.1 An Offline Q-learning based global path planner for 2D and 3D environments with priority shifting rewards.

Autonomous robotic navigation is a field of great importance due to its vast array of applications such as exploration, transportation, industry or defense. When it comes to theses scenarios, robotic agents can enable different approaches that can increase the task's efficiency and/or flexibility. In this section it is proposed an offline path planning for static 2D and 3D environments using Q-Learning. The reward shaping is done in such a fashion that is able to account for three different priorities, namely path length, energy consumption and safety, that can be tuned freely by the user to suit the desired application. Due to a well balanced exploring/exploiting ratio, the proposed method can lead the agent to the desired destination starting from anywhere in the map, which can be helpful in scenarios where internal or external disturbances that can lead the agent stray from its main path may be expected. Scalability tests where also done to benchmark the proposed method's performance for larger maps. Real world experiments were also done to validate the 2D path planning algorithm.

An offline path planning algorithm that accounts for different factors such as distance, safety and/or energy consumption during its process is proposed. It leads to a policy π, capable of guiding the agent to its goal from anywhere in the grid pondering the predetermined priorities using a reinforcement learning technique.

The final policy returned by the proposed method is influenced by the weights the user determines for each priority. In other words, the user can freely adjust the relevance of each factor to its current application prior to the training process.

Reinforcement Learning techniques required that the state, action and reward

representation are carefully tailored for each application. For this work, the states were considered the cell index in previously known discrete map, where the top left cell of the ground level is state 0, the one on the right is state 1, and so on, and for the 3D scenario, the state count is restarted count on the top left cell of the next higher level, as show in Figure 17. For the 2D environment, the action space is defined by eight high level actions, up, down, left, right and the four diagonal movements, as for the 3D scenario, two extra actions were added, allowing the virtual agent to move upwards or downwards between map levels.

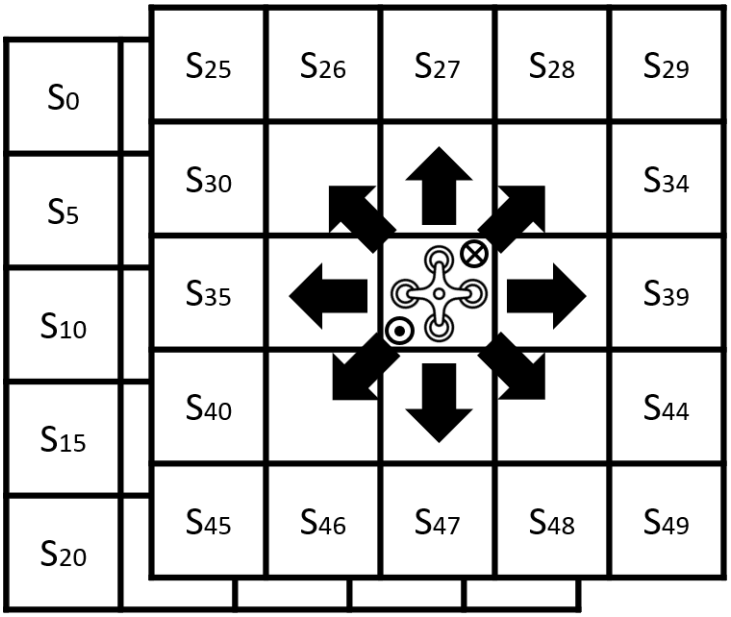


Figure 11 – State Representation and Action Space. The state is represented by the cell number of the agent. While the action space contains the eight horizontal actions in the case of 2D scenarios and adds the upwards and downwards actions in 3D scenarios.

3.1.1 Reward Shaping

The main feature that allows the user to choose the set of priorities prior to the planning process is the reward shaping. It is divided into four categories, three of them related to the priorities and one related to reaching the destination. The first available priority is path length. This reward encourages the virtual agent to reach the goal using the shortest path possible, and it is calculated by

$$r_d = \begin{cases} -K_d, & \text{if action is horizontal or vertical} \\ -\sqrt{2}K_d, & \text{if action is diagonal} \\ -0.5K_d, & \text{if action is upwards or downwards} \end{cases} \tag{3.1}$$

where K_d is a positive constant that determines how much this reward should be prioritized. The constants in each case are proportional to the size of the step in that action. The

offset between two levels of the grid is considered as being 50 centimeters and the cell size being 1x1 meter in the 2D horizontal plane. When the planning is done in a 2D scenario, since there is not possibility of moving upwards or downwards, the last part of Eq. 3.1 is ruled out.

The safety priority takes into account the distance of the agent to close obstacles and penalizes it proportionally to how close the agent is, leading to a safer route. It is calculated according to

$$r_{saf} = \sum_{i=1}^{N} \min(0, -K_s(MinDist - ObsDist_i)) \tag{3.2}$$

where N is the number of obstacles that is taken into account, $MinDist$ is the minimum Euclidean distance that an obstacle must be from the virtual agent to be accounted for (thus it does not penalize the agent when it is far from obstacles), $ObsDist_i$ is the distance from the i-th closest obstacle, and K_s is a positive constant that determines how much this reward should be prioritized.

The energy consumption priority works penalizing heavy turns, so that encourages the agent to maintain a smoother speed pattern, which is directly reflected to a lower energy consumption rate as detailed in (SHIVGAN; DONG, 2020). In addition to penalizing broader turns, this priority also weights in the energy consumption of moving upwards or downwards, with a heavier penalty for moving upwards, given it requires more motor power when applied to 3D scenarios. This is represented by

$$r_t = \begin{cases} -\frac{K_t}{\pi} arccos(\theta) & \text{if horizontal movement} \\ -0.25K_t, & \text{for downward action} \\ -0.5K_t, & \text{for upward action} \end{cases} \tag{3.3}$$

where K_t is a positive constant that determines how much this reward should be prioritized, θ is the turn angle. The first case is divided by the number π in order set the penalty interval to $r_t \in [-K_t, 0]$.

As for rewards related to ending an episode, it is defined as follows:

$$r_g = \begin{cases} 100, & \text{if reaches goal,} \\ -100, & \text{if hits an obstacle,} \end{cases} \tag{3.4}$$

and if the agent hits an obstacle, the episode is considered finished, the reward is applied and then the environment is reset, just as it happens when the agent reaches the goal.

The final reward for reaching state s_t is determine by

$$r_{s_t} = r_d + r_{saf} + r_t + r_g, \tag{3.5}$$

which sums the rewards for each priority for the current state, as well as for reaching the goal or hitting an obstacle.

3.1.2 Training procedure

To ponder between exploration and exploitation of the agent actions, $\epsilon - greedy$ approach was employed using a linear decay ratio, with the ϵ value starting at 1, reducing to 0.1 throughout the training to encourage more exploration in the early episodes in both 2D and 3D scenarios. In order to improve convergence speed and exploration levels, for the 3D path planner and the real world experiments using the 2D path planner, the exploring-starts strategy was also used, which consists on defining a random starting point for the agent every k episodes, where k is a predetermined positive integer. These two approaches combined lead to a well balanced exploration/exploitation ratio, allowing the agent to learn how to reach its goal effectively and from different points of the map.

As for the training algorithm itself, every episode the Q-Value for the current state-action pair is updated using the Bellman Equation as follows:

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha(r_{s_t} + \gamma \max_a Q(s_{t+1}, a)), \tag{3.6}$$

in case the episode has not ended and

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha r_{s_t}, \tag{3.7}$$

if the episode has ended. Here α is the learning rate, and γ is a discount factor. The current best policy for each state is given by the action with the highest Q-value for that state.

The training takes as inputs the constants for each priority provided by the user, the map grid and the destination point. Then, the training iterates through Ite_{Max} episodes, with the starting point being defined by the exploring starts strategy when applied, and in each one of them the virtual agent chooses its actions for the current state via $\epsilon - greedy$, finds the next state, updates the Q-value table and then proceeds to the next state. The episode ends if the agent reaches the predetermined goal, or if it hits an obstacle. This process is detailed in the algorithm 1.

Algorithm 1 Global Path Planning Algorithm

Inputs: K_d, K_{saf}, K_t, Map, Goal
Output: Policy π
for i from 1 to Ite_{Max} **do**
    Every k episodes change starting point, if applied
    ϵ for current episode
    while Episode not over **do**
        Choose action via $\epsilon - greedy$
        Calculates r_{s_t} via (3.5)
        $s_{t+1} = Step(a_t)$ - uses action to find next state
        Q(s_t,a_t)=Update via (3.6) or (3.7)
        $s_t = st + 1$ - updates current state
    end while
end for

3.2 An Offline Deep Q-learning based global path planner for 3D environments with priority shifting rewards.

After successfully developing robust 2D and 3D global planning algorithms using Q-Learning, this section extends the approach by incorporating more advanced techniques. While tabular methods like Q-Learning perform well in many scenarios, they can become challenging to scale in environments with high-dimensional state or action spaces. Despite the strong results achieved in previous work, this section presents the methodology for integrating Deep Q-Learning with Curriculum Learning, along with different strategies for state representation and reward shaping. The goal is to enhance the environment's resolution, enabling the development of a robust policy π that reliably guides the agent to its destination from any point on the map while optimizing path length, safety, and energy consumption. Also, to increase training speed, multiple agents were used, without the possibility of collision between them. Finally the agent is subjected to flight instabilities during training to improve the final policy's robustness.

In order to properly used Reinforcement Learning techniques, one must carefully define the state, action and reward representation. For section, the state space is defined by a combination of factors. First, the 3D coordinates of the agent, then the Euclidean distance to each obstacle in the environment, represented by buildings in a urban area, and finally the Euclidean distance to the destination. With this the state representation is more detailed than just the cell index of the agent, providing more features to better reflect the environment.

As for the action space, six primitive movements were used, namely North, South, West, East, Up and Down. Allowing the agent to move in three dimensions,

3.2.1 Reward Shaping

The proposed reward function is divided in different steps to encourage desired behaviors and inhibit the ones that can be hazardous to the mission.

A fixed penalty is given the agent for each step it takes. This encourages it to reach the destination in the shortest path it can. Every step is also penalized by the distance the agent is from its goal, which stimulates the agent to search for paths that leads it closer to the goal, as represented in

$$R_1 = \frac{K_s + \sqrt{3}K_s}{\sqrt{(x_D - x_G)^2 + (y_D - y_G)^2}}, \tag{3.8}$$

where $K_s = 1$, x_D and y_D are the agent coordinates, and x_G and y_G are the goal coordinates.

In order to assure a safe navigation, the agent receives a negative reward inversely proportional to the distance of obstacles that are closer than a predetermined safety threshold, as suggested in

$$R_2 = \frac{K_{obs}}{\sqrt{(x_D - x_{obs})^2 + (y_D - y_{obs})^2}}, \tag{3.9}$$

where $K_{obs} = \pi$, x_D and y_D are the agent coordinates, and x_{obs} and y_{obs} are the closest building coordinates.

As flight autonomy is a relevant constraint when dealing with UAVs, a negative reward is given to the agent representing the energy cost of each action. Moving in a horizontal plane demands more energy than moving downwards and less energy than moving upwards. This reward is given by

$$R3 = \begin{cases} -20, & if\ action\ is\ Up \\ -10, & if\ action\ is\ Horizontal \\ -5, & if\ action\ is\ Down \end{cases} \tag{3.10}$$

Finally, there are rewards associated with the end of an episode. Each episode ends when the agent either crashes into an obstacle, reaches exhaustion due to low battery levels or reaches its destination. Each of this events have a value associated to it, given by

$$R4 = \begin{cases} -50, & if\ crash \\ 50, & if\ reaches\ goal \end{cases} \tag{3.11}$$

As for the parallel strategy of using multipĺe UAVs in the simulation, the whole episode is considered finished when 80% of the agents ends their navigation course by one of the above mentioned events.

The final reward function then is determined by

$$R_f = R_1 + R_2 + R_3 + R_4. \tag{3.12}$$

To emulate the agent's battery levels, it was determined arbitrarily that each agent has a 5,000 unit battery and that the horizontal actions consume 10 units of energy, the upwards action consumes 20 units and the downwards action consumes 5, enabling to have a comparison of final battery levels in each episode.

3.2.2 Deep Q-Learning

Instead of using a table to store the state-action pairs' Q-Value, Deep Q-Learning (DQL) uses the function approximation capability of neural networks to express Q-functions as a function of the weight parameter *theta* as $Q(s, a; \theta)$ in order to describe a function with a high number of such state-action pairs. Here, s and a stand for the state and action, respectively, and s' is the next state (achieved by carrying out a at s). At each iteration i, the state and action information is fed to the network, which generates the estimation of the respective Q-values. The target value is

$$y_i^{DQN} = r + \gamma \underset{a'}{max} Q(s', a'; \theta^-), \tag{3.13}$$

by optimising the loss function represented in the form of an expectation over s, a, r, s' given by

$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{(s,a,r,s')} \left[\frac{1}{2} \left(y_i^{DQN} - Q(s, a; \theta_i) \right)^2 \right], \tag{3.14}$$

where θ^- denotes the target network parameters. The two key aspects that improve the algorithm performance are:

- Target Network - In order to reduce the moving target effect due to boostraping the estimation of the Q-values of the next state and, therefore reducing training stability, two networks are used simultaneously, a prediction network and a target network. The later has its weights updated in a lower frequency than the first. After a fixed amount of iterations the target network weights are copied from the current values of the prediction network.

- Experience Replay - During training, experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ are collected over many episodes to form a dataset of experiences $\mathcal{D} = \{e_1, e_2, e_3 \cdot |\mathcal{D}|\}$. Instead of using samples in the standard sequence, mini-batches are randomly sampled from this dataset for training in order to reduce correlation between samples.

  Including these in (3.14) results

$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{(s,a,r,s')}\ \mathcal{D} \left[\frac{1}{2} \left(y_i^{DQN} - Q(s, a; \theta_i) \right)^2 \right], \tag{3.15}$$

whose gradient is given by

$$\Delta_{\theta_i}\mathcal{L}(\theta_i) = \mathbb{E}_{(s,a,r,s')}\ \mathcal{D}\left[\left(y_i^{DQN} + \right.\right.$$
$$\left.\left. - Q(s,a;\theta_i)\right)\Delta_{\theta_i}Q(s,a;\theta_i)\right]. \tag{3.16}$$

The gradient is utilized to update the model's parameters θ through a gradient descent algorithm in order to obtain the optimal set of parameters. To ensure convergence to the optimal parameters, the learning rate for the weight update is gradually decreased over epochs. The neural network architecture used in this work consists of 5 fully connected layers with 30, 5, 8, 5, and 2 neurons, respectively, with all layers employing the ReLU activation function. The Adam optimizer was utilized for training.

3.2.3 Training Procedure

The training process follows an episodic approach where multiple agents navigate a 3D environment while learning an optimal policy through Deep Q-Learning. At the beginning of each episode, agents are randomly placed within the environment with a fixed battery capacity. The episode continues until 80% of the agents have either reached the goal, exhausted their battery, or collided with obstacles.

During training, the agents interact with the environment by selecting actions based on the ϵ-greedy policy. The reward function, as defined in previous sections, guides the agent's behavior by balancing path efficiency, safety, and energy consumption. The experience replay buffer stores past experiences, and mini-batches are randomly sampled to update the Q-network with the target network being used to stabilize training by reducing the moving target effect. This process is represented in Pseudoalgorithm 2

3.3 An Online Q-learning based local path planner for 2D environments with priority shifting rewards.

3.3.1 Reward Shaping

The user still has the flexibility to prioritize the different constraints—path length, safety, and energy consumption—by adjusting the constants associated with each, as seen in Equations 3.1, 3.2, and 3.3 on Section 3.1.1, considering their application in 2D scenarios. The agent continues to receive a reward for reaching its goal, which is now a local goal based on the current grid (as shown in Equation 3.4). Additionally, a fine-tuning reward was introduced to discourage potentially dangerous behaviors, as demonstrated in Figure 15. Since a cell is considered occupied if any obstacle is present, regardless of whether the obstacle occupies most or just a small portion of it, the agent could cause a collision if it

Algorithm 2 Deep Q-Learning Global Path Planning Algorithm

Inputs: Learning rate α, discount factor γ, replay buffer $\mathcal{D}$, ϵ-greedy parameters
Output: Policy π
Initialize Q-network $Q(s, a; \theta)$ with random weights
Initialize target network $Q(s, a; \theta^-)$ with $\theta^- \leftarrow \theta$
Initialize replay buffer $\mathcal{D}$
for i from 1 to Ite_{Max} **do**
    Every k episodes update starting conditions, if applied
    Set ϵ for current episode
    while Episode not over **do**
        for each agent **do**
            Observe current state s_t
            Select action a_t using ϵ-greedy policy
            Execute a_t, observe next state s_{t+1}, and receive reward r_{s_t}
            Store transition $(s_t, a_t, r_{s_t}, s_{t+1})$ in $\mathcal{D}$
        end for
        if buffer $\mathcal{D}$ is sufficiently full **then**
            Sample mini-batch (s, a, r, s') from $\mathcal{D}$
            Compute target $y = r + \gamma \max_{a'} Q(s', a'; \theta^-)$
            Update Q-Network via (3.15)
        end if
        if update step condition met **then**
            Update target network: $\theta^- \leftarrow \theta$
        end if
    end while
end for

moves as shown in Figure 15(a). To prevent this, the fine-tuning reward is proportional to K_d for each step taken with this type of movement. This results in the agent preferring the movement pattern shown in Figure 15(b) over the one in Figure 15(a). The reward is defined as:

$$r_{ft} = -10.K_d \tag{3.17}$$

where K_d is a positive constant that determines how much this reward should be prioritized.

making the new total reward defined by

$$r_{s_t} = r_d + r_{saf} + r_t + r_{ft} + r_g, \tag{3.18}$$

3.3.2 Local Grid

The path planning process is done using a local grid in the surroundings of the agent. It is predefined and fixed at 5 by 5 cells, but it faces a direction based on the last action the agent took as seen in Figure 13, where if the last action was diagonal, the grid

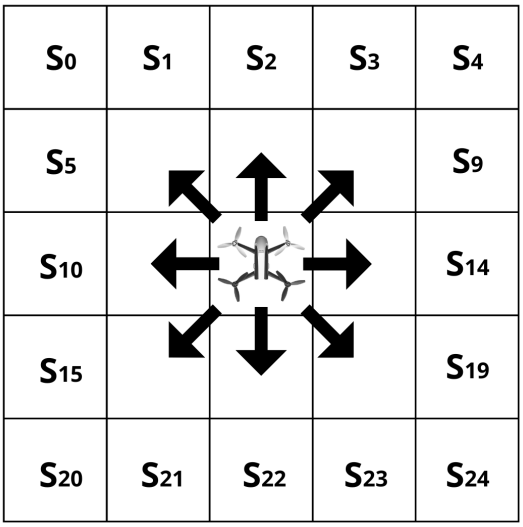


Figure 12 – States and Actions for the proposed method with the arrows being the possible actions and the cell index indicating the state number.

will have the agent in one of its corners, and if its one of horizontal or vertical actions, the grid will have the agent in the middle one of the grid sides. It is worth highlighting that the last action is inferred by analyzing the agent's current grid cell and the previous one, which can lead to a diagonal movement being perceived as a combination of a horizontal and a vertical action or vice versa, given that sometimes the agent won't dislocate from one cell to another going exactly over the vertex that connects both. The local goal is defined as the empty local grid cell closest to the global destination.

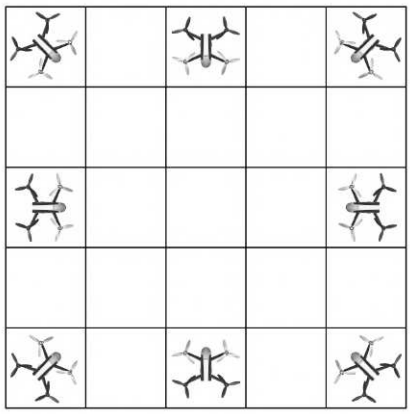
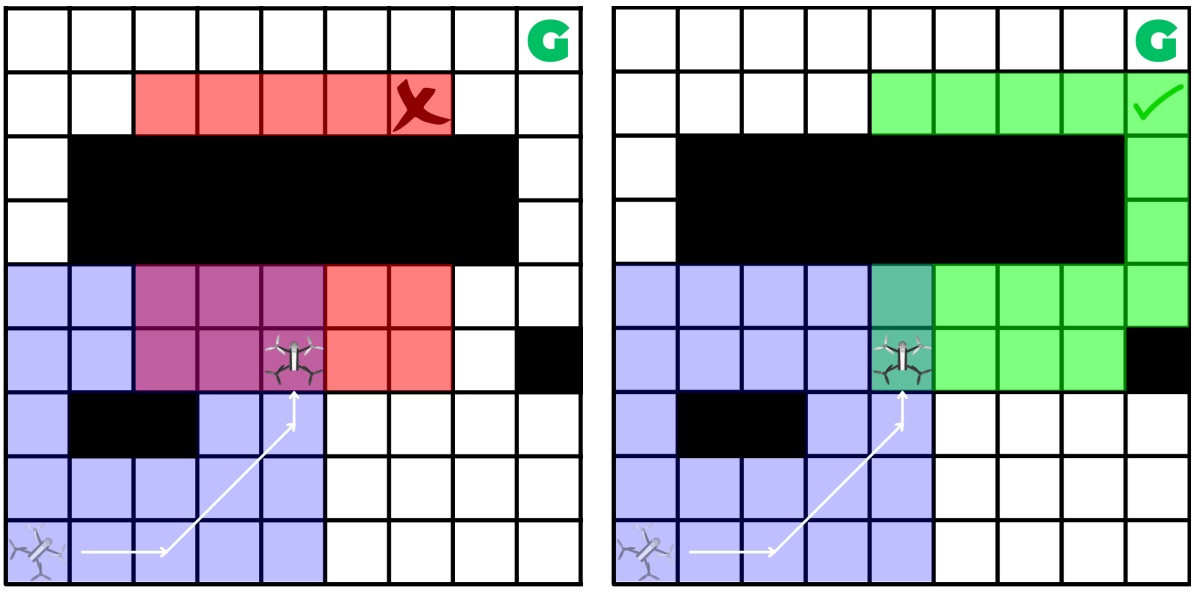


Figure 13 – Local Grid position in respect to the agent's last action

After the path planning algorithm is processed, a sequence of waypoints is generated and a smooth path is defined. There are two different triggers to request a new path. The first one is when the agent reaches the cell just before the current local goal. This issues a request for a new local path and it's not done when reaching the local goal to avoid the possibility of having an obstacle near the destination, but that was outside of the local grid when the destination was defined. With this, the agent will have maneuvering room to avoid this situation. In the rare occasion where the local goal is defined as the cell right

next to the agent, a recalculation trigger will be issued, then the last action used to define the local grid's limits is the action defined by the local policy. The second trigger is when dynamic obstacles' movement is detected inside the current local grid. This is done to allow the agent to re-plan its path taking into account the obstacles' new position.

When calculating a new path, the local grid used is defined by the agent's last action as seen in Figure 13, but in some scenarios, there might not be a feasible path between the agent's path to its goal. In order to deal with this situation, the local grid is treated as a monochromatic image, where empty cells are white and obstacles are black, then a quick segmentation technique is employed to separate the 5 by 5 image in regions, if the agent is not in the same region as the goal, it means that there is no feasible path, then the local grid is rotated as if the agent had done a different action. This action is chosen in clockwise order, i.e. if the agent's last action was up, the next local grid candidate will be in respect to the upward right diagonal action, as seen in Figure 14.

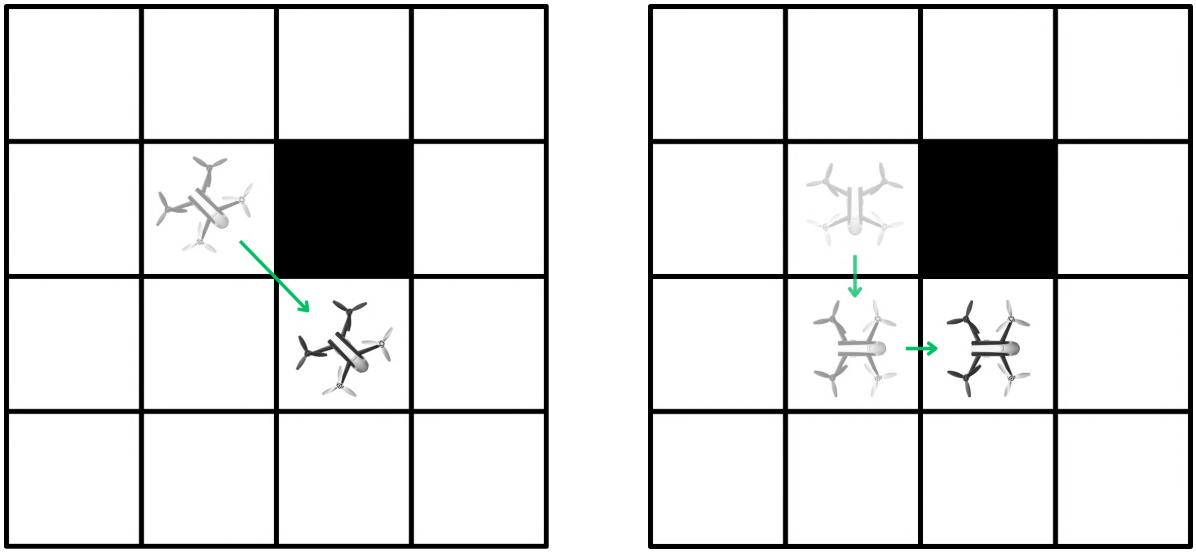


(a) No feasible path (b) Feasible path

Figure 14 – Example of local grid rotation due to no clear path between the agent and the local goal. The local goal is defined as the empty cell closest to the global goal, represented as a green G.

3.3.3 Training Procedure

Given the local grid size, the algorithm is fast enough to train the agent in real time, to provide the new path when needed. The training works similarly as seen in the previous section where it iterates until Max_{Ite}, with the starting point for every episode being defined by the exploring starts strategy. The actions chosen by the agent are done following the ϵ-greedy approach, then the state-action pair is updated using Bellman's



(a) Undesired Behavior (b) Desired Behavior

Figure 15 – Example of motion behavior that could lead to collisions in real world applications and the expected correction from the fine tuning reward.

Equation. The episode ends if the agent reaches its goal, or if it hits an obstacle. This process is detailed in Pseudocode 3.

Algorithm 3 Local Path Planning Q-Learning Algorithm

```
Inputs: Kd, K_saf, K_t,Grid, Local and Global Goal
Output: Q-Table (Q)
for i from 1 to Max_Ite do
    Define starting point according to exploring starts
    while Episode not over do
        Select Action - Using ϵ-greedy
        Calculates r_{s_t} via (3.18)
        s_{t+1} = Step(a_t) - uses action to find next state
        Q(s_t,a_t) = Update via (3.6)
        s_t = s_{t+1} - updates current state
    end while
end for
```

3.3.4 Path generation

In order to provide a smooth path, the waypoints outputted by the optimal policy found after training, a Bèzier curve based approach is employed to generate a fully connected, continuous, and smooth path considering the current position of the robot and the arrangement of the way-points whenever they are updated by the local planning algorithm. Consider the set of Cartesian points, or grid cell indices, returned by the Algorithm 1, represented as $\mathbf{q}_{d,i} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_k \end{bmatrix}^\top$, where $\mathbf{x}_k = \begin{bmatrix} x_k & y_k & z_k \end{bmatrix}^\top$, $k > 1$, and i is the iterator for the way-points sets received over time. The approach adopted here is

divided into two forms of path creation: for the first set of way-points ($i = 1$), the current position of the robot must be connected to the first k-th point in $\mathbf{q}_{d,1}$; and for subsequent local planning iterations ($i > 1$), the newly calculated path is connected to the closest point on the current path, considering the robot's position. A special case that needs to be addressed is when $k = 1$, which typically occurs when the robot is one grid cell away from the final destination and an obstacle moves. In this case, the adopted strategy is to connect this point to the existing path, while preserving the characteristics of the globally planned path, such as smoothness and continuity.

In order to do so, we leveraged the mathematical formulation that describes Bèzier curves, which always connect an initial and final point - as mathematically demonstrated in (FAROUKI, 2008) - with a smooth and continuous path, delimited by the convex polyhedron described by its control points. Instead of creating a single curve connecting the first and last nodes smoothly, using the other way-points as control points, Bèzier curves are used here to generate curves between each of the way-points, connecting the point closest to the global path and the second $\mathbf{q}_{d,i}$ way-point using the first way-point as control point of the curve, and so on.

The Bèzier curve is defined by

$$\mathbf{B}[n] = \sum_{i-0}^{k} \binom{k}{i} (1-n)^{(k-i)}\, n^i \cdot \mathbf{P}_i, \quad \forall\, n \in [0, 1, 2, \ldots, N], \tag{3.19}$$

in which $\binom{k}{i}$ are the binomial coefficients. The points $\mathbf{P}_i \in \mathbb{R}^{3\times k}$ are called control points for the Bézier curve.

3.4 Curriculum-based Reinforcement Learning for an Effective Multi-Agent Path Planning Algorithm in Warehouse Scenarios

In many situations, multi-robot systems can significantly increase flexibility and/or efficiency. The literature has papers for applications in a variety of contexts, including search and rescue, disaster management, and load transportation. The logistics in a warehouse scenario is one instance where the utilization of numerous agents can be quite advantageous. In order to conduct the path planning process, this work suggests a multi-agent Q-learning based algorithm with curriculum learning and transfer learning. The algorithm can achieve high success rates by training in progressively more sophisticated phases and transferring knowledge from one stage to the next. Simulated comparisons were made between the suggested method and different combinations of the employed techniques as well as to Q-learning alone in order to validate the method. Scalability tests were also performed. The proposed method achieved up to 94% success rate after training.

In the literature it can be found works employing more sophisticated reinforcement learning algorithms such as Deep Q-Learning, or even as simple as Q-learning, but only few are capable of working with multiple robots. This is clearly still an open issue. Our current work is a initial step towards proposing a reinforcement learning algorithm for multi-agents to work in logistic applications in warehouse like scenarios.

In this section we propose a multi-agent Q-Learning based algorithm is proposed to deliver objects in a warehouse scenario using Curriculum Learning and Transfer Learning to improve the baseline reinforcement learning algorithm. This section lays out the details of each step of the development of this work.

In this work, the environment map is used as a real-world example of warehouse logistics situation. The initial justification for this map's applicability is that the warehouse's blueprint has been rasterized, the locations are largely fixed, and the map size is constrained, all of which contribute to make modeling the problem as a MDP. The second reason is that by sharing the acquired information in the database, path calculation during system operation uses fewer resources. The $Q-$learning algorithm in reinforcement learning is used as the main learning technique because the method does not require an accurate environment model. In addition, the method is an off-policy algorithm, which means that it does not necessarily uses the current best policy to guide it actions in the learning stage, facilitating full search without affecting the generation of optimal policies and utilizing experience to a greater extent than on-policy algorithms. These two points clarify the suitability of the $Q-$learning algorithm for learning in the path-planning problem.

Since the environment's state information changes in response to the actions of various robots, it is inevitable that the complexity of the learning strategy rises as the state and action dimensions do as well. This leads to two issues: the first is that computational complexity rises, and the second is that learning effectiveness may decline. Agent path-planning must take into account avoiding collisions with both static and dynamic obstacles, such as stationary objects like shelves and workbenches as well as other robots, and it must also aim to get as close to the target point as possible by traveling the shortest distance. In order to ensure that the agents of the entire system arrive at their respective target points in a coordinated manner, this study combines the current motion information of all robots into an action vector. The state data of every robot is also included in a state vector to ensure that the environmental information is constantly changing. In this way, in the entire system, the Q-value tables of each robot are all merged into combined action vectors and combined state vectors mapped to Q-values so that the actions and states of other robots in the system can be taken into account when the current robot makes a decision.

The warehouse scenario in this work is a structured environment that is assumed to be known, so the agents have access to the locations of both the static and dynamic

obstacles and are informed in advance of their locations. Operation desks and shelves are examples of static obstacles, whereas people and other environmental agents are examples of dynamic obstacles.

A curriculum learning approach was proposed to divide the training process into stages, allowing the agent to gain knowledge in progressively more challenging contexts. The final objective is to have multiple agents taking objects simultaneously from predetermined points to their goals to perform it, this curriculum approach is split in four stages, described as follows.

3.4.1 Stage One: Free Space

The starting stage requires one agent to reach from one point to another in a map with no obstacles. The agent state is defined as the cell number the agent occupies and the action space of the agent comprehends the four possible actions as seen in Figure 17. The starting and ending point of the task was chosen by random in each episode of training and the reward function is defined by Eq. (3.20), where K is a positive reward. Figure 16 shows a sample representation for this stage.

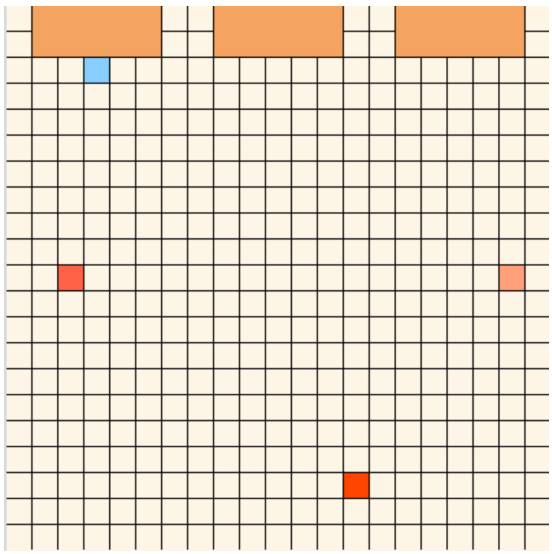


Figure 16 – Illustration of the First Stage of Training, where the upper orange rectangles are the workstations where the agent will receive objects and the orange squares are possible destinations.

$$reward_{s1} = \begin{cases} K, & \text{if reaches goal} \\ 0, & \text{otherwise} \end{cases} \tag{3.20}$$

The number of steps a robot would take in this warehouse scenario is equal to the Manhattan distance. Use of Manhattan distance as the comparing criterion is therefore more logical. Chebyshev distance ought to be more appropriate to use if the robot could move in eight directions.

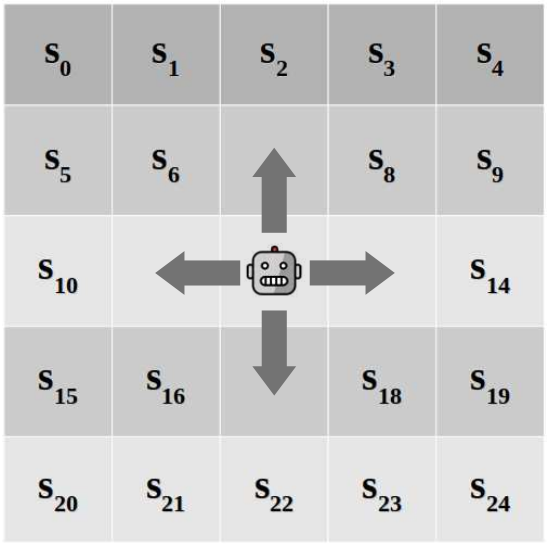


Figure 17 – States and Actions for the proposed method. The state is represented by the cell number. The action space comprehends the four actions represented by the arrows.

3.4.2 Stage two: Static Environment

The second stage increases the difficulty by adding the warehouse's static obstacles to the grid map, while the agent's goal remains the same: to go from the starting point to a random destination. The agent state is defined as the agent's cell number as well as the cell number of the static obstacles and the action space maintains the same as previous stage. The reward function now has a penalty for hitting an obstacle, which also triggers the end of the episode. The new reward function is given by

$$reward_{s2} = \begin{cases} K, & \text{if reaches goal,} \\ -K, & \text{if hits an obstacle,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.21}$$

Figure 18 shows a sample representation for this stage.

3.4.3 Stage three: Dynamic Environment

For this stage, dynamic obstacles are randomly added to the grid, while the agent still tries to reach a random destination starting in a random cell each episode. The state is now defined as the cell number that the agent occupies as well as the cell number of each obstacle, dynamic or static. The action space and reward function remains the same as the previous stage Eq. (3.21).

Figure 19 shows a sample representation for this stage.

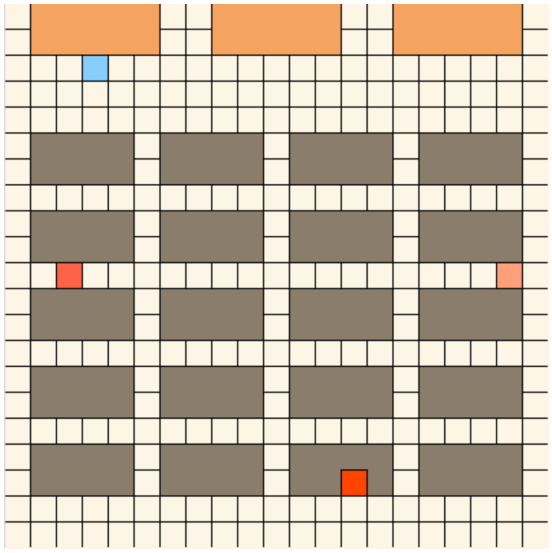


Figure 18 – Illustration of the Second Stage of Training, where the brown rectangles are added static obstacles, which represents the warehouse's shelves.

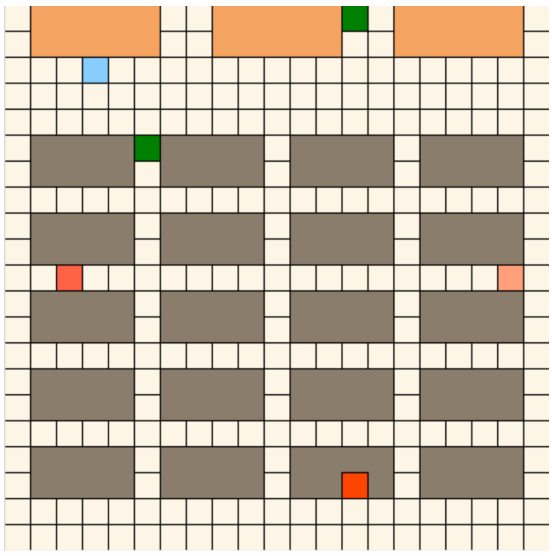


Figure 19 – Illustration of the Third Stage of Training, where the green squares are added dynamic obstacles, i.e, human workers.

3.4.4 Stage four: Desk Operation

The final stage represents the actual task. Therefore more agents are added to the system and each of them must reach an individual goal and come back to their predetermined operation desk. The agent state is now defined by the cell number that the agent occupies, as well as the cell number of the static and dynamic obstacles (the other agents are considered dynamic obstacles for this purpose). The action space remains the same as on previous stages and the new reward function is defined by

$$reward_{s4} = \begin{cases} K, & \text{if reaches goal,} \\ K, & \text{if reaches operation desk,} \\ -K, & \text{if hits an obstacle,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.22}$$

where the reward for reaching the operation desk is only credited if the agent has already reached the goal in the first place.

Figure 20 displays an example of the final stage of training, with blue color squares being the agents, green human workers, orange the operation desks and dark brown the warehouse shelves.

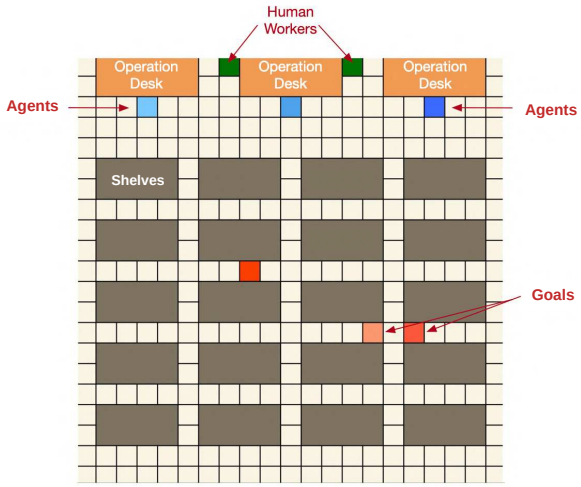


Figure 20 – Warehouse scenario simulation. Green shades blocks represent dynamic obstacles, i.e. human workers. Blue shaded blocks represent the robotic agents. Orange shaded blocks in the middle of the grid are the delivery points and the ones on top are operation desks. Finally the brown shaded blocks are the warehouse's shelves.

3.4.5 Transfer Learning

In order to use the knowledge of previous stages, on the start of a new one the agent's Q-Table is initialized via Transfer Learning using the previous task as source domain. By recognizing similar states between the current stage and the prior one, such as the agent being in the same grid map cell in both, Inter Task Mapping is performed. After that, a threshold is used to determine which Q-values should be added to the subsequent stage's Q-Table, giving the agent its first shred of information. The imported Q-values have an attenuation factor applied to them because even though the new stage may be similar to the past one, is not exactly the same. The threshold and attenuation applied to each Q-value in this work were both set at 60%. In other words, the Q-values that are

moved to the next $Q-$table for each state of the previous stage are those that are higher than 60% of the highest Q-value for that state, and these values are initialized at 60% of their prior value.

The overall learning algorithm works as follows: The first stage initializes its Q-table with zeros and each subsequent stage has its Q-Table initialized using the transfer learning approach described in this work. In each stage, for each episode until a predefined maximum number of iterations Ite_{Max}, while the episode has not ended, an action is chosen using the $\epsilon - greedy$ strategy, then the next state, s_{t+1}, is observed and the reward r_t is obtained. Finally the $Q-$value for the state-action pair is updated using Eq. (3.23) in case the episode has not ended and Eq. (3.24) in case it has, and then the current state is updated. This process is detailed in the algorithm 4.

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_{s_t} + \gamma \max_a Q(s_{t+1}, a)), \tag{3.23}$$

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha r_{s_t}, \tag{3.24}$$

The proposed curriculum and the use of other Transfer Learning techniques to initialize the Q-table for the new stages substantially increases the convergence speed and overall performance of the algorithm, as it will be shown in the next Chapter.

Algorithm 4 Warehouse Scenario Training Algorithm

```
Inputs: Environment for each Stage
Output: Q−Table(Q)
for stage ST in Curriculum do
    if not first ST then
        Initialize Q−Table via Transfer Learning
    end if
    for episode from 1 to Ite_Max do
        while episode not done do
            Chose action using ϵ − greedy
            Observe a new state s_{t+1}
            Calculate reward r_t
            Update Q(s_t, a_t) via (3.23) or (3.24)
            s_t ← s_{t+1}
            Update current state
        end while
    end for
end for
```

3.5 State Representation Selection for Mapless Autonomous Navigation with Deep Reinforcement Learning

As discussed in previous chapters, defining the state and action spaces is crucial when modeling decision problems as MDPs and applying them in reinforcement learning. However, there is no clear consensus in the literature on the best approach to determining an agent's state (FAGUNDES-JUNIOR et al., 2024; CHEN et al., 2022), nor do existing works extensively explore how different state representations impact the agent's learning. This section introduces the concepts and methodology for investigating various state representations using depth sensors for mapless autonomous navigation. The study employs Double Deep Q-Learning to assess the effects of different state representations based on depth sensor readings and quantization methods.

3.5.1 Simulation Environments

Three simulation environments were designed for mapless navigation simulations to train a model and measure its performance in new territory. Figure 21 illustrates these environments, containing obstacles of varying shapes, in different positions, orientations and distances from each other. Environment 1 is shown in Figure 21, at the top-left corner, featuring one base point location where the agent starts each simulation (B1), as well as two goal locations (G1) and (G2). The goal (G2) is designed for verifying that the agent did not overfit and memorize the path to (G1), since only (G1) will be used in training.

In the bottom of Figure 21, Environment 2 similarly shows a starting point (B2) and two different goal points (G3) and (G4), that will be used to determine whether the agent is able to navigate in a larger, previously unseen map after being trained. Finally, at the top-right corner of Figure 21 displays a final base point (B3) and goal point (G5), in a room identical in size to Environment 1, however containing no obstacles. The purpose of Environment 3 is to validate that the agent is able to navigate environment without reference points from obstacle shapes.

Pioneer 3-DX is a ground robot designed by Adept Mobile Robots, capable of achieving 1.2 m/s in linear velocity and 300°/s angular velocity. It has two front wheels with differential drive motors, and a third stabilization non-controllable back wheel. The model for this robot is available in Gazebo and a ROS-based communication handler class was developed for measuring the position and orientation of the agent in the environment, as well as for sending linear and angular velocity commands.

As for the sensing unit, a 2D-LiDAR depth measurement instrument available publicly for use in Gazebo was chosen. Its measurements are returned within ROS communication with a 270° field of view, which is reduced in this work to 727 measurements in a

0 to 5-meter range, from 0 to 180°, centered at the front of the agent.

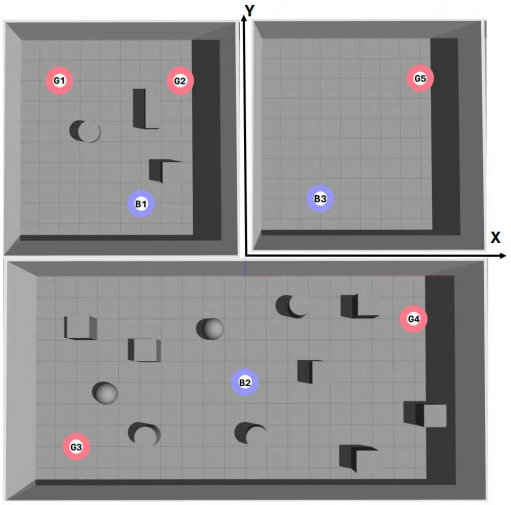


Figure 21 – Simulation environments for training the agent and testing its performance, where starting bases are shown in blue and ending goals are indicated in red.

In the training environment, as shown in Figure 22, the initial position and orientation are defined for the agent training, seen also as (B1) in Figure 21. Here, a depth sensor measures the distance between the agent and objects within the environment, which must be modeled to - ideally - ensure autonomous navigation with safety guarantees. Figure 23 displays the complete readings of the depth sensor in this state, covering a range of 180 degrees centered in front of the agent and collecting 727 measurements within a 0 to 5-meter depth range.

Furthermore, Figure 23. separates the readings into 4 sectors, each spanning 46 degrees on the topmost image. Below, Figure 23 similarly presents the same readings divided into 10 intervals of 18 degrees each. A comparative analysis of the content within each sector in Figure 23 reveals that different details captured within the same initial measurement. Specifically, Figure 24 showcases sector #2 from the topmost image in Figure 23 and the 4th sector on the lower image, capturing the edge of the wall through the collected measurements and the histogram of this sector.

In this work, it is defined the concept of Depth State Representation Methods (DSRM), which are quantizations of the sensor sample distribution of a number of sectors by a statistical measure. Three approaches were used: the mean, mode, and the minimum value of the twentieth percentile, henceforth referred to as ”soft_min”, as DSRMs to be used for 4, 5, 6 and 10 sectors.

Notably, when comparing the readings in Figure 24 it becomes evident that the same edge, when quantified through different sampling, results in distinct measurements of the same environmental phenomenon. Consequently, this study proposes defining the agent’s state based on the separation of LiDAR measurements into 4, 5, 6, and 10 sectors, using the mean, mode, and ”soft_min”. Ensuing sections will then describe how, by varying

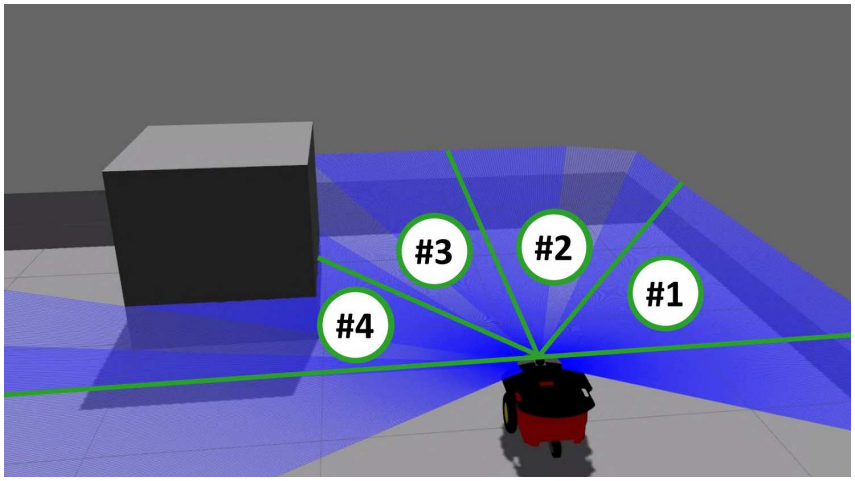


Figure 22 – Depth scans are selected from -90° to 90°, centered in front of the robot, and split into 4 sectors with equal sample size.

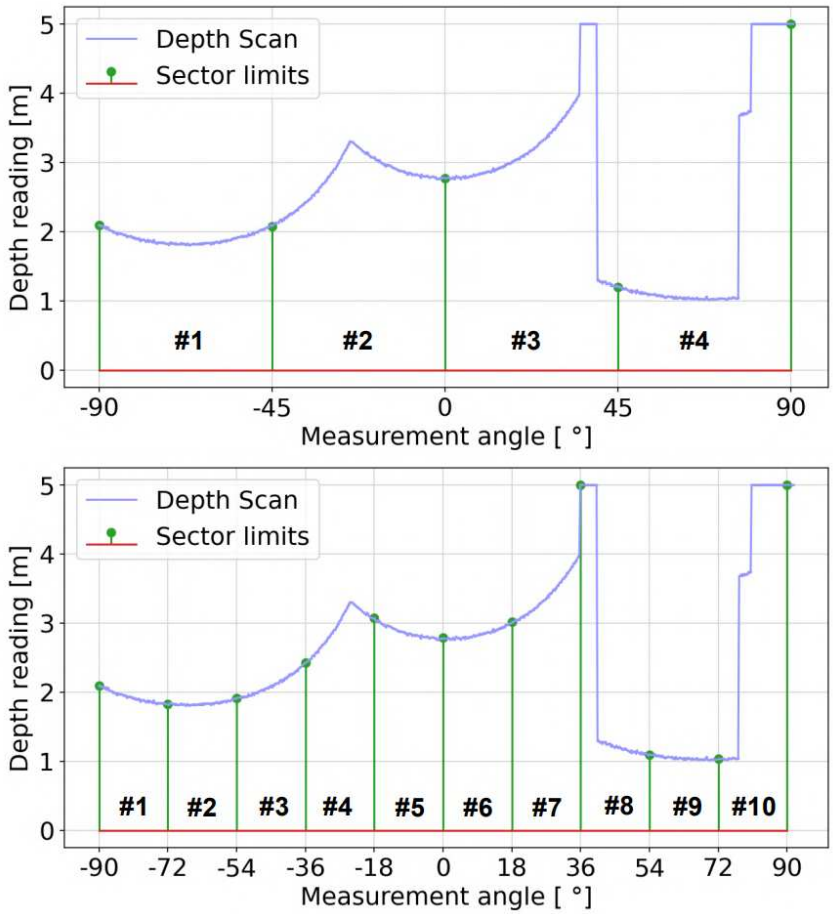


Figure 23 – LiDAR depths scans are split into 4 and 10 sectors respectively, resulting in sample sizes in each sector for the same measurement.

the DSRM, the number of states and their rendition of the environment result in differently shaped DDQN models to be trained and compared in tests.

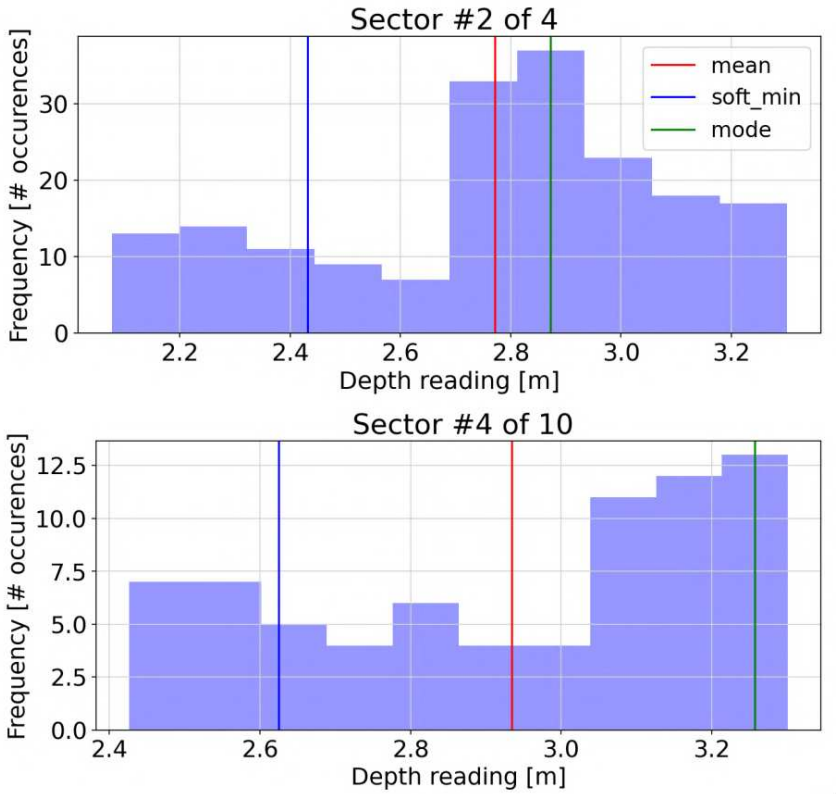


Figure 24 – Selecting sector #2/4 and #4/10 demonstrates that the same edge on the wall (seen in Fig. 22) results in varying distributions and therefore different mean, mode and soft_min values.

3.5.2 State and Action Space

Building upon the DSRM definitions, the states that the agent uses to perceive the environment are defined as a discrete set of numeric values. Moreover, in order to encourage the agent to be oriented towards the goal, an extra goal-orientation directional state $\mathcal{S}_o$ is defined by splitting the depth agent orientation into 6 states. Namely, an integer is used to represent whether the goal is behind, or in one of five directions in front of the agent.

The goal-orientation directional state, $\mathcal{S}_o$, is designed as:

$$\begin{aligned}\mathcal{S}_o = \Big\{ & S_{back}, S_{[-90,-53°]}, S_{[-54,-17°]}, \\ & S_{[-18,17°]}, S_{[18,53°]}, S_{[54,90°]} \Big\} \\ = & \{0,1,2,3,4,5\} \in \mathbb{N}^1\end{aligned}$$

such that, $\mathcal{S}_o = 0$ if the goal is behind the agent, $\mathcal{S}_o = 1$ if the goal is at the LiDAR angle interval between $[-90, -53°]$, and so on for the other intervals.

The set of depth states $\mathcal{S}_d$ is modeled as a discrete set of sectors with 4, 5, 6 or 10 elements, each of which are real numbers computed with a given statistical measure, composing a DSRM: $\mathcal{S}_d \in \mathbb{R}^{\{4,5,6,10\}}$.

Finally, the complete state space for the agent is defined as the union of the set of DSRM samples and goal-orientation directional state: $\mathcal{S} = \mathcal{S}_d \cup \mathcal{S}_o$.

Three actions are selected for the agent to navigate the environments. Namely, the agent can either move straight forward with linear velocity, or make a soft curve to its left or right, composed by a linear and angular velocity. Thus, we define the action space as a discrete set of three actions, mathematically given by:

$$\mathcal{A} = \left\{ \begin{array}{ll} a_{right} & = (\dot{\xi} = 0.2 \text{ m/s}, \dot{\psi} = -0.4 \text{ rad/s}), \\ a_{forward} & = (\dot{\xi} = 0.4 \text{ m/s}, \dot{\psi} = \quad 0 \text{ rad/s}), \\ a_{left} & = (\dot{\xi} = 0.2 \text{ m/s}, \dot{\psi} = +0.4 \text{ rad/s}) \end{array} \right\} \tag{3.25}$$

where

$$\text{Agent Pose: } \left\{ \begin{array}{l} \xi = [x, y] \\ \psi = \arctan(y/x) \end{array} \right. \tag{3.26}$$

Thus, the agent action and state spaces are formally defined, where the action set is $\mathcal{A} \subset \mathbb{R}^{3\times 2}$ and the state set is $\mathcal{S} \subset \{\mathbb{R}^{\{4,5,6,10\}+1} \cup \mathbb{N}^1\}$.

3.5.3 Reward Shaping

The reward function was modeled based on reliability and efficiency requirements for the task of autonomous navigation. In detail, five rewards are presented below, composing the total reward for any given new state after acting.

First, the distance reward, R_{dist}, which is a reward given based on how close the agent is to the goal at each iteration of an episode. The reward fraction R_{dist} is calculated as:

$$R_{dist}(i) = k_{R_{dist}} \cdot \frac{|D_{start} - D_i|}{D_{start}} \tag{3.27}$$

where the initial and current distance to the goal are given by:

$$D_{start} = \|\xi_{start} - \xi_{goal}\|$$
$$D_i = \|\xi_i - \xi_{goal}\|$$

such that the reward for staying at the starting position is null and increases as the agent reaches the destination in every iteration step i within an episode. The constant $k_{R_{dist}}$ is a free parameter that was set to 80 analytically, to yield null rewards after a certain number of iterations, related to the negative reward R_{steps}. Note that, because of the modulus function, this reward does not inhibit straying further from the goal, which is addressed later in R_{steps}.

A reward to related to collisions is proposed, R_{col}, which is a negative reward built for inhibiting the agent from navigating close to objects, other environment components or

crashing. For each sector, the depth sample distance in each iteration ($d_{s,i}$) is discretized as close (≤ 0.5 m), acceptable ($\leq 1,3$ m) or safe, and negative rewards are attributed to each case:

$$R_{col}(i) = \sum_{s=1}^{N_{sectors}} \left\{ \begin{array}{ll} -100, \text{ if} & d_{s,i} \leq 0.5 \text{ m} \\ -2, \text{ if} & 0.5 < d_{s,i} \leq 1.3 \text{ m} \\ -1, \text{ if} & d_{s,i} > 1.3 \text{ m} \end{array} \right\} \tag{3.28}$$

A success reward, $R_{success}$, was designed to encourage the agent to reach the vicinity of its goal. Numerically:

$$R_{success}(i) = \left\{ \begin{array}{ll} k_{R_{success}} & \text{, if } D_i \leq d_{reach} \\ 0 & \text{, if } D_i > d_{reach} \end{array} \right. \tag{3.29}$$

where $k_{success}$ is a free constant parameter, set to 20. The minimal distance from the agent to the goal - used to consider the episode a success - is defined as d_{reach}. It is also a free constant parameter that was set to 0.5 meters in our simulations.

In order to encourage the agent to minimize the number of steps taken to finish its navigation, a negative steps reward, R_{steps}, was designed and is computed as:

$$R_{steps}(i) = -k_{steps} \cdot i \tag{3.30}$$

were k_{steps} is a constant positive free parameter, set analytically to 2, such that a reward identical to a collision would be given for every step after iteration number 50.

To further improve the agent's navigation towards the goal, a direction reward, R_{dir}, is proposed to align its direction to the goal by using the goal orientation state. If the orientation state $\mathcal{S}_o$ is aligned to the goal a positive reward is given, and a negative reward is received otherwise:

$$R_{dir}(i) = \left\{ \begin{array}{ll} -k_{dir,1} & \text{, if } \mathcal{S}_o \neq 3 \\ k_{dir,2} & \text{, if } \mathcal{S}_o = 3 \end{array} \right. \tag{3.31}$$

where $\mathcal{S}_o$ is the goal orientation state present in all methods, $k_{R_{dir}1}$ is a positive free parameter set to 3, and $k_{R_{dir}2}$ is a positive free parameter set to 10 to match other reward amounts.

Finally, the total reward for each iteration of episode is calculated as the sum of the five previous reward. Therefore, a total reward:

$$R_T(i) = R_{dist} + R_{col} + R_{success} + R_{steps} + R_{dir}$$

is given after reaching a new state, at each iteration sequence of action selection after a state observation.

3.5.4 Double Deep Q-Learning

Deep Q-Leaning was the first step in the Deep Reinforcement Learning field. Even though it pushed the boundaries of what was possible to do in terms of action and state spaces as well as results, it has its draw backs such as a tendency to overestimate Q-values due to its method for computing the loss function for gradient descent, which is computed as follows:

$$\text{Loss}(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a', \theta) - Q(s, a; \theta^-)\right)^2\right]$$

r : is the reward for the current state

γ : is a discount factor to future rewards

a, a' : are the initial and next actions

s, s' : are the initial and next states

where θ is the network for estimating Q, while θ^- is a copy of θ made before the current training. This difference between the *a priori* ($Q(s, a; \theta^-)$) and *a posteriori* ($Q(s', a', \theta)$) reward estimation is known as Temporal Difference (TD) error.

The DQN method for computing the TD error results in systematic overestimation of rewards, and an approach to solve this issue was provided by Hado van Hasselt (HASSELT; GUEZ; SILVER, 2016), proposing that two networks are used in the learning process, entitled the Double Deep Q-Network (DDQN) method.

In DDQN, one network is trained and used to estimate Q-values in order to select actions, and will be regarded in this section as the "Selector θ" network. Another network, named here as the "Evaluator θ^-", is used for estimating more accurate Q-values *a posteriori*, such that the DDQN loss equation is given by:

$$\text{Loss}(\theta) = \mathbb{E}\left[\left(r + \gamma Q(s', a'(\theta); \theta^-) - Q(s, a; \theta)\right)^2\right]$$

$$a'(\theta) = \arg\max_{a'} Q(s', a'; \theta)$$

such that the Selector θ chooses both the current and next actions (a, $a'(\theta)$) but the Q-value of next action is computed by Evaluator θ^-. This Q-value evaluated by θ^- is then subtracted from the Q-value originally estimated by Selector θ, computing a more stable TD error, thus also a more stable loss and convergence. Similarly to DQN, in DDQN the Evaluator θ^- is never trained, only copied from Selector θ after a chosen amount of training episodes.

In practical terms, this means that errors by Selector θ are computed comparing its *a priori* Q-value with the *a posteriori* Q-value estimated by an older copy using the originally chosen actions, reducing overestimation. DDQN has shown better results than

DQN in terms of training length in time as well as resulting performance, and was therefore selected.

In terms of neural network architecture, a shallow multi-layered perceptron has been shown to yield sufficient performance, and was thus selected as minimal architecture in this work. Namely, it is defined as the sequential application of activation functions of inputs multiplied by weights and biases matrices, representing the oriented graph for the network, shown in Figure 25.

In this architecture, the input state s is processed through three fully connected layers with the 'Leaky' ReLU activation function. The first layer (θ_1) maps the input to a 64-dimensional hidden representation, the second layer (θ_2) further transforms this representation in a 64-dimensional space, and the third layer (θ_3) maps it to a 3-dimensional output representing the Q-values for each of the 3 available actions. Mathematically, outputs from the network seen in Figure 25 are given by:

$$Q(s,a) = f_{\text{L-ReLU}}\left(\theta_3 \cdot f_{\text{L-ReLU}}\left(\theta_2 \cdot f_{\text{L-ReLU}}\left(\theta_1 \cdot s\right)\right)\right) \tag{3.32}$$

where:

$$\begin{aligned}
s &: \text{Input state vector of size } \{4,5,6,10\}+1 \\
a &: \text{Action vector of size 3} \\
\theta_1 &: \text{Input weights, shaped } (\{4,5,6,10\}+1)\times 64) \\
\theta_2 &: \text{Hidden layer weight matrix, shaped } (64\times 64) \\
\theta_3 &: \text{Output layer weight matrix, shaped } (64\times 3) \\
f_{\text{L-ReLU}} &: \text{'Leaky' Rectified Linear Unit activation}
\end{aligned}$$

This architecture learns to estimate Q-values for state-action pairs, therefore computing the expected sum of future rewards for selecting actions in mapless navigation.

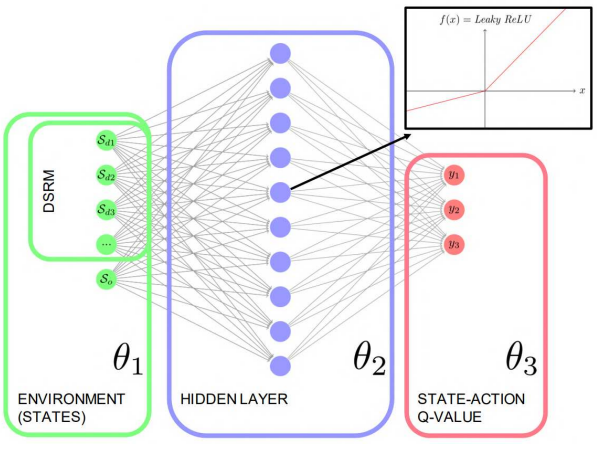


Figure 25 – A fully connected network is used to estimate Q-values for each of the 3 possible actions, given a state.

3.5.5 Training Procedure

Following the definition of the training and testing environments, using the agent states and possible actions, an algorithm was defined for training the DDQN model. In detail, the learning phase is split into training and validation where the convergence of the model for each given DSRM is analysed.

In DRL, mapless autonomous navigation is achieved by means of action selection based maximizing expected rewards. That is, in practical terms, selecting the action that has the highest Q-value for a state-action pair and therefore the highest expected rewards.

In Figure 26, the complete loop for autonomous navigation is shown, where the agent perceives the environment through its DSRM, generating a state s. Inputting this state into the Selector network computes Q(s,a) for each action as outputs, and selecting the output with the highest Q-value results in choosing one of the three possible actions. At each decision, the agent has an $\epsilon \in [80, 0.15]$ probability of choosing a random action instead of the network output, a method known as ϵ-greedy for balancing exploration and exploitation, which, in this case, starts at 80% chance of a random action and linearly decays until 15%.

This incurs in interaction with the environment by moving, which leads the agent to its next state s'. This set of state, action, new state and the Q-values (s,a,s',Q) is then saved to a memory buffer at every iteration, storing up to 50k iterations in our particular algorithm.

An episode is defined by an environment navigation sequence of at most 60 iterations of action selection based on Q-value estimation from the current state. Episodes may be shorter than 60 iterations - also named steps - in case of success or collision.

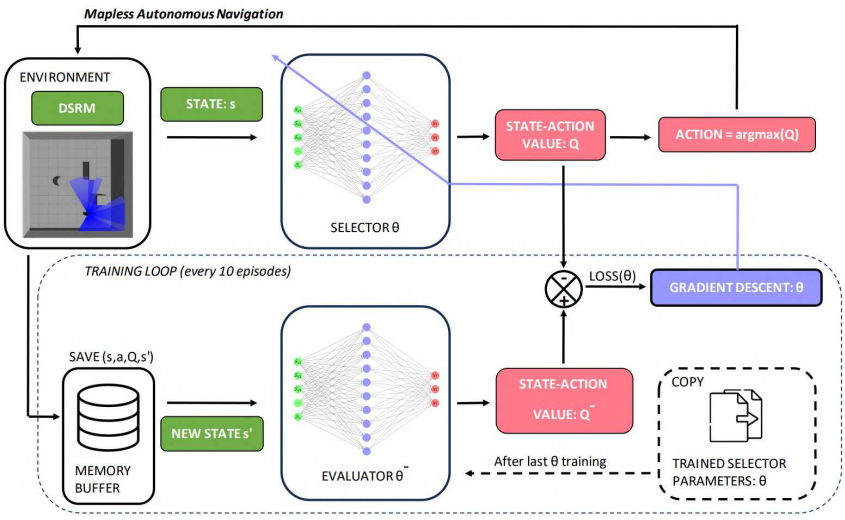


Figure 26 – A Selector network is trained for the mapless autonomous navigation task, supported by a less-often updated Evaluator network (DDQN).

After each 10-episode interval, the simulation is paused and the agent enters the Training loop, that is prescribed within the dashed area in Figure 26. In this stage, each recording from the memory buffer has its s' (next state) inputted into the Evaluator θ^-, thus estimating future rewards from its outputs Q^-.

Then, these estimated future rewards from the Evaluator θ^- are subtracted from the rewards estimated originally by the Selector θ, resulting in a squared error that is used as loss function for applying gradient descent to the Selector θ. Effectively, this loss represents the TD error, and is used for correcting the reward expectations of the Selector θ. Next, the parameters for the Evaluator θ^- are updated by copying the newly-trained Selector θ as shown on the right side of Figure 26. Finally, the simulation resumes to navigation, using only the Selector θ for decisions, and applying the Evaluator θ^- only for the training loop.

This process of autonomous navigation lasts for 25k episodes - iteratively training at each 10 episodes - at which point the training process is interrupted regardless of its performance. Note that this training process is used for 25k episodes for each specific DSRM, resulting in 12 different models: one for each pair of number of sectors (4, 5, 6, 10) and sampling method (mean, mode, "soft_min").

4 Results and Discussion: Global and Local Path Planning for Single Agent

This chapter presents the results of four studies on global path planning, along with findings from a paper on local path planning. The first study models the problem as a 2D navigation scenario, while the second extends the algorithm to 3D grids. The third validates the 2D path planner through real-world experiments, and the fourth explores global path planning using Deep Reinforcement Learning and Curriculum Learning. The fifth section discusses the algorithm's adaptation for local path planning. The sixth covers results from multi-agent systems in intelligent logistics scenarios, and the final section examines the impact of state representation in mapless navigation.

4.1 2D Global Planner Algorithm Validated with Simulations

The first iteration of the algorithm was applied for 2D path planning problems. It was validated by simulations with UAV navigation applications in mind, even though the algorithm can be generalized and applied to ground agents as well. This subsection displays the obtained results.

These simulations were done to showcase the two main features of this algorithm. The first one is that the it returns feasible paths for a static environment, taking into account different priorities such as path length, safety and energy consumption that can be freely adjusted by the user to suit the application's needs. The second one is that the policy found can lead the agent to the desired goal starting from any point in the map. To do that, the agent was trained in four maps, with different obstacle disposition. The agent starts in the top left cell and the destination is the bottom right cell. Each grid in the map is 1×1 meter. The results can be seen in Figures 27, 28, 29 and 30.

The proposed algorithm acts on known scenarios. When there is an obstacle inside a cell, it was considered to compromise it whole, for safety measures. Therefore, having a blacked out cell does not mean that there is a square shaped obstacle in it, but only that there is a relevant obstacle inside the cell.

Training for all maps was done using the learning rate $\alpha = 0.1$, the discount factor $\gamma = 0.9$, $MinDist = 3, N - 4, Ite_{Max} = 10000$ and considering the start point as the top left cell and the destination point as the bottom right one.

4.1.1 Simulations for Path Planning with Different Priorities

These simulations were run to allow analyzing the performance of the algorithm in finding feasible paths accounting for different combinations of priorities. It was tested for path length with energy consumption as well as with all the three priorities. In the first case, the constants used for the rewards were $K_d = 2, K_s = 0, K_t = 1.5$, and in the second case they were $K_d = 2, K_s = 1, K_t = 2$.

Map 1 represents a scenario with a large obstacle on the bottom left and a corridor in the middle of the map. Going through the corridor would reduce the path length, but is also a riskier strategy. The simulations show how the agent ponders the priorities in this scenario.

In Figure 27 it is possible to see a clear behavior difference when changing priorities in the path planning process. In the case of prioritizing shorter path length and energy consumption the policy of the agent will guide it in a straight diagonal line from the starting point to the goal, which is represented by the red line. When safety measures are also taken into account, the green path is taken, where the agent starts in a straight line to the right, then takes a small diagonal turn, and finally goes straightly downwards to the goal. In this simulation, the minimum distance, $MinDist$, to start penalizing the agent for being close to an obstacle, was set as 3. This reinforces the reason why the agent moved seven grids right on the green path before turning down, even though moving diagonally would reduce the path length, if the turn was taken earlier it would put the agent close

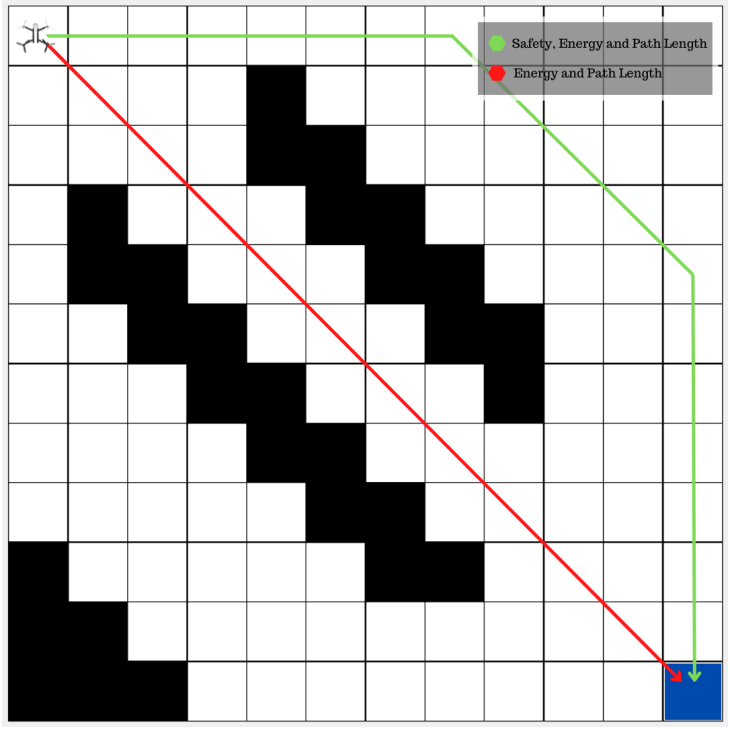


Figure 27 – Map 1: Paths taking into account different sets of priorities.

enough to the obstacles to be penalized due to safety rewards. The consideration of future rewards in (3.6) assures that the agent will not go under the obstacle corridor, due to the obstacle in the bottom left region of the map.

Map 2 represents a scenario with two large obstacles. It counterpoints with Map 1 due to not having the option to move through the obstacles, only around them, while leveraging the determined priorities. The simulations show how the agent ponders the priorities in this scenario.

The second map shown in Figure 28 also showcases the behavior of the path planning algorithm when only path length and energy consumption are prioritized, and taking safety as an important constraint. The red path shows the case in which the agent minimizes path length and energy consumption. This is why it gets close to the obstacle. But it is possible to notice that only soft turns are taken. When safety is as concern as well, the agent takes a path to the top right region of the grid in order to stay away from the obstacle in the center and not taking the risk of getting close to the obstacle in the left bottom part of the map, even though it would result in a shorter path. The agent takes only soft turns as well on the green path.

4.1.2 Simulations to Analyze Optimal Policy

This simulation was run in order to showcase how the optimal policy leads the agent to the destination starting from any point in the map. The constants used for this

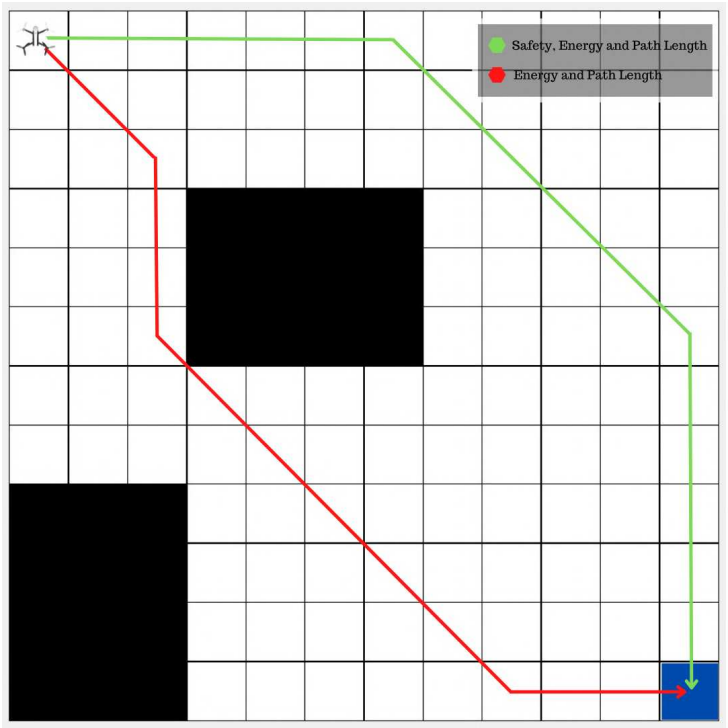


Figure 28 – Map 2: Paths taking into account different sets of priorities

simulation were $K_d = 2,\ K_s = 1,$ and $K_t = 2$.

Maps 3 and 4 are cluttered environments to test how the algorithm performs in more complex scenarios with common path planning pitfalls, such as U-Shaped obstacles.

Figures 29 and 30 show that the algorithm can find a path even in more complex and cluttered scenarios. The arrows represent the best action for each state according to the policy π^*. It is possible to notice that from anywhere on the map, the agent would reach the goal. So in case of any disturbances that could lead it astray from the main path, it would be able to still reach the goal with the appropriate priorities. It is worth highlighting as well that even on dead-end regions such as the top right part of Map 3 and the U-shaped obstacles in both maps can be overcame with the optimal policy.

In Map 4 the optimal path avoids the last U-shaped obstacle going upwards instead of moving around it because this would lead the agent towards the goal faster and also would move less cells close to obstacles as all the cells around this obstacles, highlighted as red arrows, consist on a longer path and are closer than the $MinDist$ used for the simulations.

Aerial agents are more prone to suffer from external disturbances than ground agents. Therefore, having a main path to follow, but also a policy that can lead it to its destination leveraging the determined priorities in case of such disturbances is important.

The above results show a clear preference for diagonal paths, which makes sense since it is more efficient path, in terms of length. A vast amount of path planning algorithms

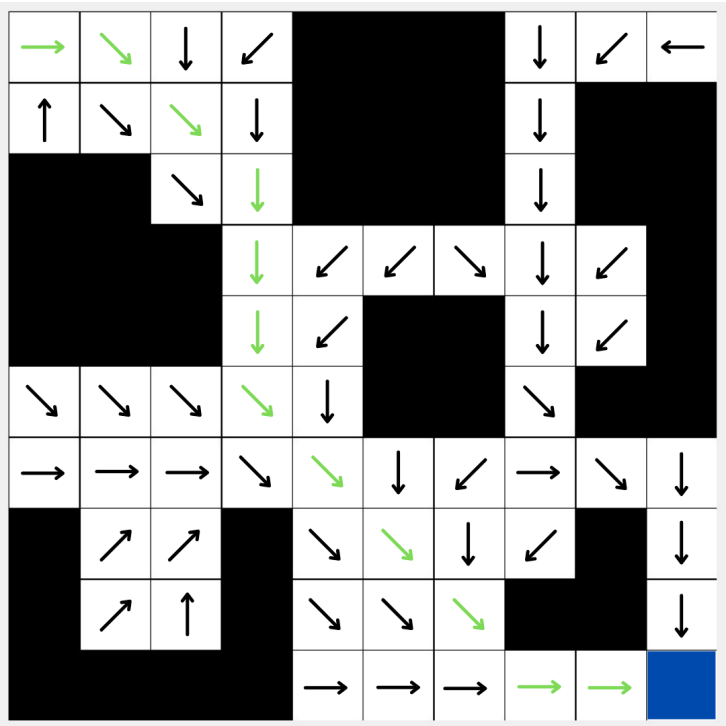


Figure 29 – Optimal Policy for Map 3 taking into account all the three priorities.

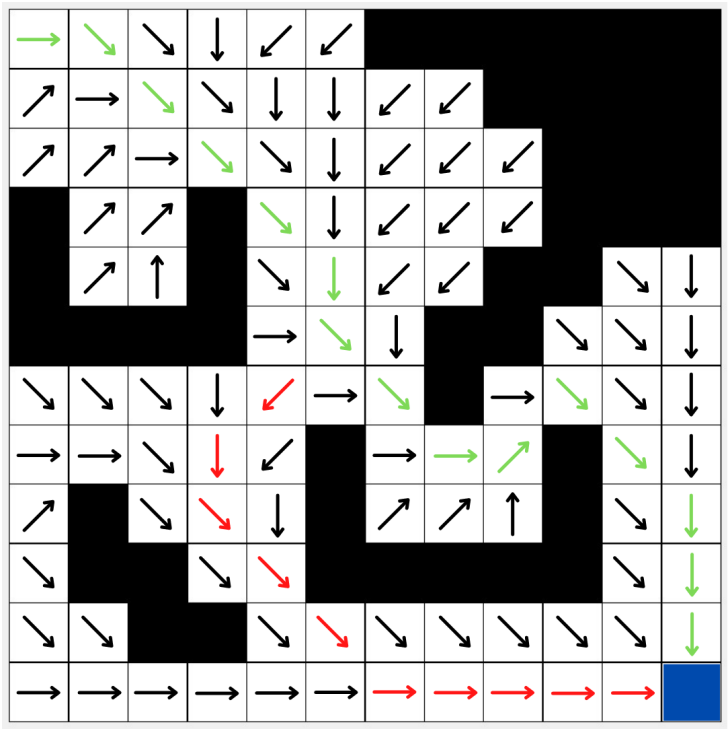


Figure 30 – Optimal Policy for Map 4 taking into account all the three priorities.

Table 1 – Time consumed for 10 000 iterations on each map.

Size	4x4	5x5	7x7	9x9	11x11	13x13	15x15	20x20
Time(s)	1,6	2,2	5,1	6,7	10,15	13,17	14,5	22,7

in the literature do not allow for diagonal movements, which can lead to non efficient paths, in terms of energy consumption.

4.1.3 Simulations for Time Consumption

Maps from sizes ranging from 40×4 to 20×20 with similar blocked cell coverage were used in order to showcase the time consumption for 10 000 episodes of the learning algorithm, taking in account all three priorities. The specifications of the computer used for simulations are the following: a AMD Ryzen 5 3600 CPU, with 16GB of RAM. No GPU resources were used in order to speed the training up. Table 7 shows the average time consumption for maps of different sizes over ten repetitions.

Given that the proposed algorithm is meant for offline path planning and returns not just a main path, but a policy that can lead the agent to its goal from anywhere in the map to account for internal uncertainties and external disturbances, a planning time from 1.6 to 22.7 seconds proves feasible for the proposed problem.

It was proposed an offline path planning method for static environments using Q-learning. The reward shaping was designed to be able to account for, in different intensities, path length, safety and energy consumption. The exploration-exploitation ratio used for

training also allows the agent to learn how to reach the goal starting from anywhere in the map, which can be useful in scenarios where internal or external instabilities can be expected. The algorithm was tested in different environments and could find feasible paths towards the goal, successfully accounting for the designed priorities. Furthermore, the planning time was found feasible for real world applications.

4.2 3D Global Planner Algorithm Validated with Simulations

In the publication process of the first manuscript, the reviewers questioned if this algorithm could be scalable for 3D scenarios. Even though most UAV applications can be done by simplifying the 3D scenario to a 2D, or even 2.5D representation, some will required a more present and careful 3D planning. Therefore the extended version of the original algorithm and also compared its performance to the graph search algorithm dijkstra. This subsection displays the obtained results.

4.2.1 Simulation Settings

Simulations were done to showcase the two main features of the proposed algorithm. The first one is that the proposed algorithm is able to provide feasible paths in a static 3D environment, while it takes in account different priorities such as path length, safety and energy consumption. The other feature is that the policy obtained after training is capable of leading the agent to the desired goal from anywhere in the map. In order to showcase theses features, the agent was trained in three maps with different obstacle disposition and compared them to the baseline shortest path obtained by the Dijkstra algorithm. Time consumption simulations were also done with increasing map size, to test scalability.

The proposed algorithm acts on known scenarios, when there is an obstacle inside a cell, it was considered to compromise it whole, for safety measures. Therefore, having a blacked out cell does not mean that there is a square shaped obstacle in it, but only that there is a relevant obstacle inside the cell.

Each grid in the map is considered to be $1\times1\times0.5$ meters. Training for all maps was done using the learning rate $\alpha = 0.1$, the discount factor $\gamma = 0.99$, $MinDist = 3m, N = 10, Ite_{Max} = 50000$. Two priority sets were used to obtain different paths. The first weighting energy consumption and path length, using the reward function constants as $K_d = 0.4, K_t = 0.4$ and $K_s = 0$, the second set takes into account all three priorities, using the reward function constants as $K_d = 0.4, K_t = 0.4$ and $K_s = 0.4$. Exploring starts changed the starting point every five episodes.

4.2.2 Simulation Results

The results of the path planning algorithm with different priorities are displayed in Figures 31-33 as well as on Tables 2-4, where the performance metrics such as path length, sum of average distance to the ten closest obstacles and number of turns. The origin point and destination for each path is represented by a circle and triangle marker, respectively. The destination used for training was fixed at the triangle marker's position.

Table 2 – Performance Metrics for Map 1

Path	Length(m)	Avg. Dist(m)	Turns
Dijkstra	10.65	22.35	5
Priority Set 1	12.24	21.55	3
Priority Set 2	14.24	24.04	6

Analyzing Figure 31, it is possible to see that the green path took less turns than the other ones, which is expected since it prioritizes only path length and energy consumption, leading to longer straight lines. The blue path takes in account all three priorities and took as many straight lines it could while keeping a safe distance from the obstacles, leading to a longer, but safer path as compared to the others. The green path took in account energy consumption and path length priorities, which lead to a longer path, comparing to the Dijkstra's, but only three turns were done, compared to five turns on the yellow path. Finally the blue path took more turns and has a longer path length, but is the safest of them all, having the sum of the average distance to the ten closest obstacles of 24.04 m.

Figure 32 shows a bigger map then the previous one, comparing three different paths, Table 3 shows the performance metrics for them. The green path takes in account only energy and distance optimization and is slightly longer than the Dijkstra's path, but takes one less turn, having longer straight lines to improve energy consumption rates. The blue path takes in account all three priorities, providing a longer but safer path than

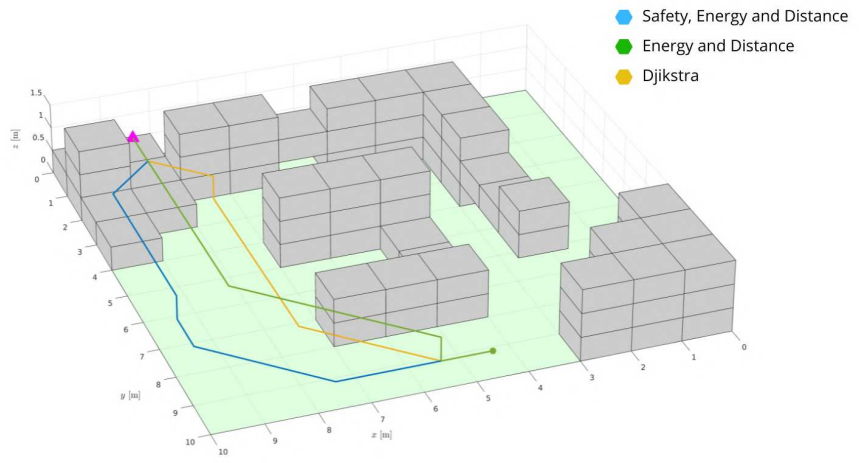


Figure 31 – Paths for different priorities on Map 1, with size of 10m×10m×1.5m.

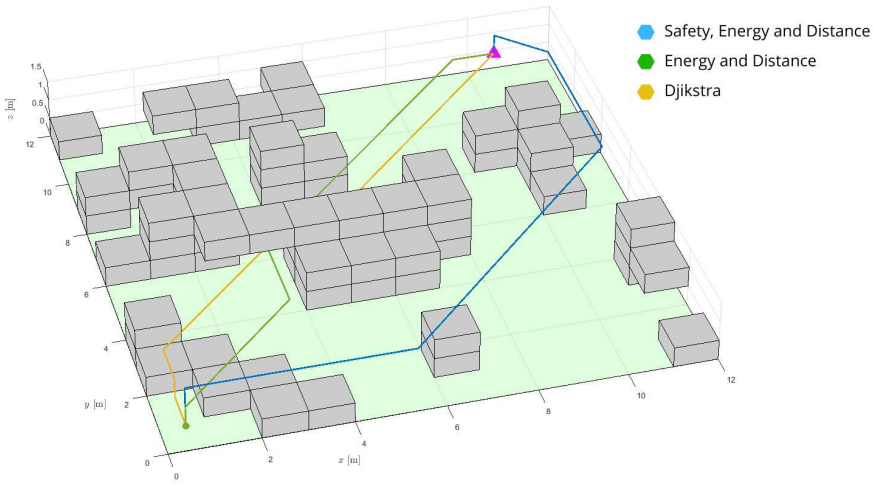


Figure 32 – Paths for different priorities on Map 2, with size of 12m×12m×1.5m.

the others, having the sum of the average distance to the ten closest obstacles of 28.03m compared to 24.66m of the second safest path. It is also possible to notice that the blue path takes long straight lines to optimize the energy consumption.

Table 3 – Performance Metrics for Map 2.

Path	Length(m)	Avg. Dist(m)	Turns
Dijkstra	16.22	23.47	5
Priority Set 1	16.72	24.66	4
Priority Set 2	21.89	28.03	5

Figure 33 shows a map with larger obstacles having two options to transpose the middle of the environment, a broader but further opening, or a closer but narrow passage. The yellow path is, by a small margin, the shortest path, but as seen on Table 4, it takes one extra turn when compared to the green path, which accounts for the first set of priorities. The second set of priorities leads to the blue path, which, even though is the longest one, maintains a higher average distance from the obstacles, being the safest path. This is made clear, by observing that this path goes through the wide arch instead of the narrow passage the other two paths take.

Table 4 – Performance Metrics for Map 3.

Path	Length(m)	Avg. Dist(m)	Turns
Dijkstra	17.81	18.19	7
Priority Set 1	17.89	19.2	6
Priority Set 2	23.55	24.51	9

To showcase the ability of reaching the goal from different points in the map, Figures 34-36 exhibit many paths leading to the goal starting in alternative locations in every tested map. These paths were obtained taking in account only energy consumption and path distance. All the paths can lead the agent to its destination, from several different

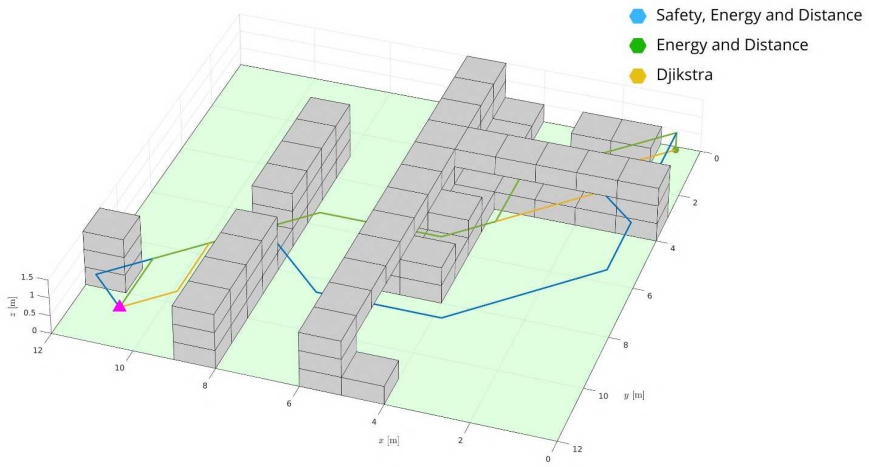


Figure 33 – Paths for different priorities on Map 3, with size of 12mx12mx1.5m.

starting points with different obstacle profiles in their way. This provides robustness to the path planning algorithm, as it can still lead the agent to its goal if internal or external disturbances lead it stray. Even though just some paths are displayed in the image, all the states were tested for each map, and all of them are able to reach the destination.

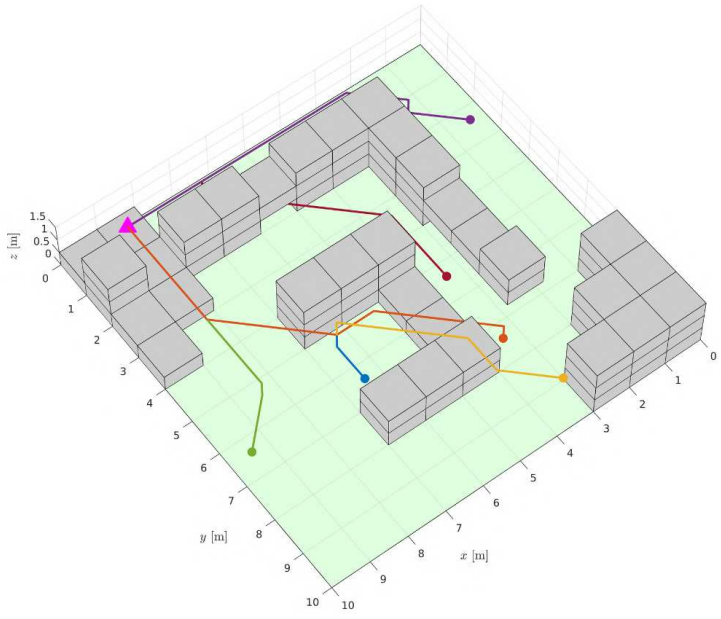


Figure 34 – Paths starting from alternative locations map 1.

Finally, it was proposed a scalability test, which involved using the proposed algorithm on maps with increasing size and 30% of its cells randomly filled with obstacles. With a fixed height of five cells, the other two dimensions started as 3×3 and were increased until 20×20. The computer used for this test has 16 GB of RAM, an Intel Core i7-7700. The results can be found in Table 5.

Given that the proposed algorithm is meant to be used for offline planning on static environments, and is robust due to its ability to reach the goal even if it is lead stray from

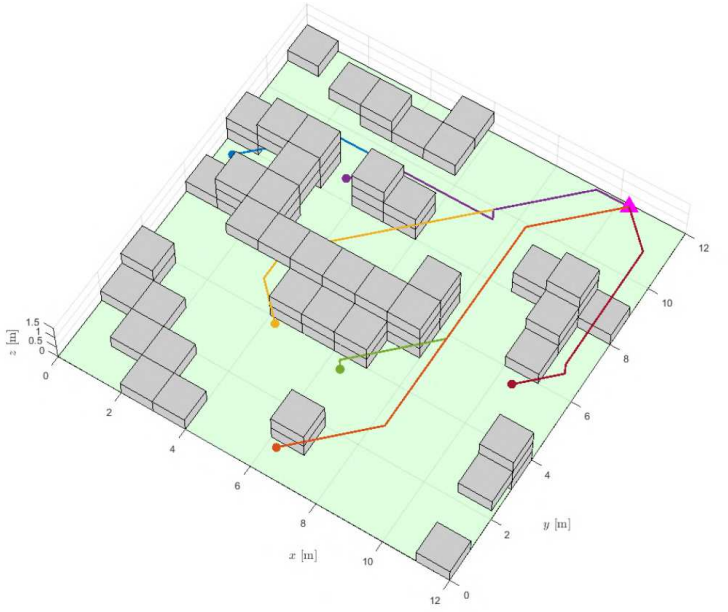


Figure 35 – Paths starting from alternative locations on map 2.

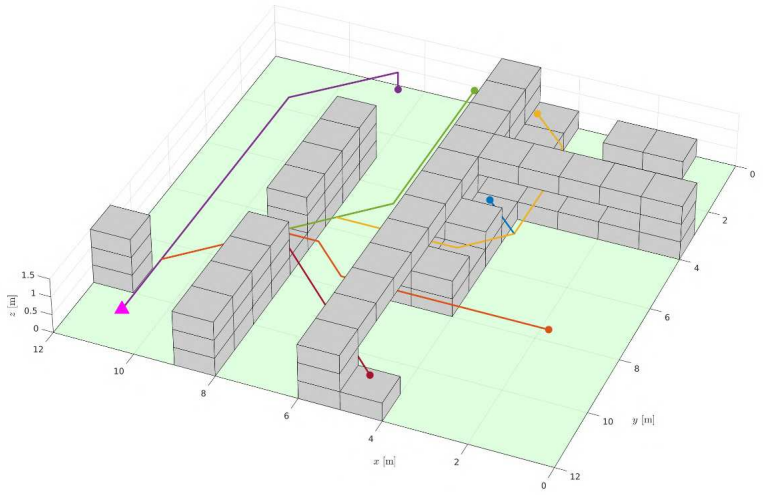


Figure 36 – Paths starting from alternative locations on map 3.

Table 5 – Time consumed for 50 000 iterations on each map with 5 levels of altitude.

Size	3x3	4x4	5x5	6x6	7x7	8x8	9x9	10x10	15x15	20x20
Time(s)	5.2	5.6	14.5	16.1	15.3	22.2	21.5	40.2	61,3	100,7

its main path, a processing time of under two minutes for a map as big as 20×20×5 cells is feasible for practical applications.

4.3 2D Global Planner Algorithm Validated with Simulations and Experiments

This is the final iteration of the algorithm, using exploring starts to accelerate convergence speeds and is also compared with Dijkstra's algorithm as baseline. The proposed path planner was validated through simulations as well as real world experiments. This subsection displays the obtained results.

Simulations and actual tests were conducted in order to highlight the two key aspects of the proposed algorithm. The first is that, while taking into account several priorities including path length, safety, and energy consumption, the algorithm generates feasible paths for static environments. The second is that the agent can start from any location on the map and follow the policy found to the intended destination. In order to assess the algorithm's scalability, time consumption tests with maps of increasing size are also performed. This Section is divided to deal separately with the simulations and real world experiments.

Our proposed algorithm acts on known scenarios. When there is an obstacle inside a cell, it was considered to compromise it whole, for safety measures. Therefore, having a blacked out cell does not mean that there is a square shaped obstacle in it, but only that there is a relevant obstacle inside the cell.

Training for all maps was done using the learning rate $\alpha = 0.1$, the discount factor $\gamma = 0.99$, $MinDist = 3, N = 10$,$k = 5, Ite_{Max} = 100000$ and considering the start point as the bottom left cell and the destination point as the top right one.

4.3.1 Simulation Results

Simulations were done using a gridworld-like environment similar to what is seen in (LEIKE et al., 2017) and in platforms such as OpenAI Gym.

Two types of simulations were done. Firstly three different maps were used to train the agent in order to demonstrate the different paths obtained by shifting the desired priorities and compare them to the Dijkstra algorithm. The results also display a policy map, where it is possible to see that the agent is able to reach the destination from anywhere in the grid. For this end, all maps used are 13x13 in size and each cell is $1x1$ meter. All starting points were set as the bottom left cell and the destination as the top right one. In order to test the scalability of our algorithm, it was tested in environments with 20% of their cells as randomly positioned obstacles and with size ranging from 4×4 to 20×20 maps.

4.3.1.1 Simulations for Path Planning

These simulations were run to allow analyzing the performance of the algorithm in finding feasible paths accounting for different combinations of priorities. Two sets were used in training, the first one prioritizing path length and energy consumption only, using the reward constants with the values $K_d = 0.6, K_s = 0, K_t = 0.6$, the second set prioritized all three constraints, with reward constants of $K_d = 0.1, K_s = 0.6, K_t = 0.6$. In order to compare the obtained paths more analytically, performance metrics were also provided, such as overall path length, sum of the average distance to the ten closest obstacles as well as the total number of turns.

Map 1 represents a scenario with a large obstacle in the middle of it as well as a long, curved one. Maps 2 and 3 were designed with increasing complexity, being more cluttered than the first one. In Figure 37 it is possible to see on images (a)-(c) the paths found for each priority set as well as the Dijkstra path and images (d)-(f) shows the obtained policy after training using the second priority set, where each arrow indicates the best action to take in each state. Furthermore, Table 6 shows the performance metrics for each map.

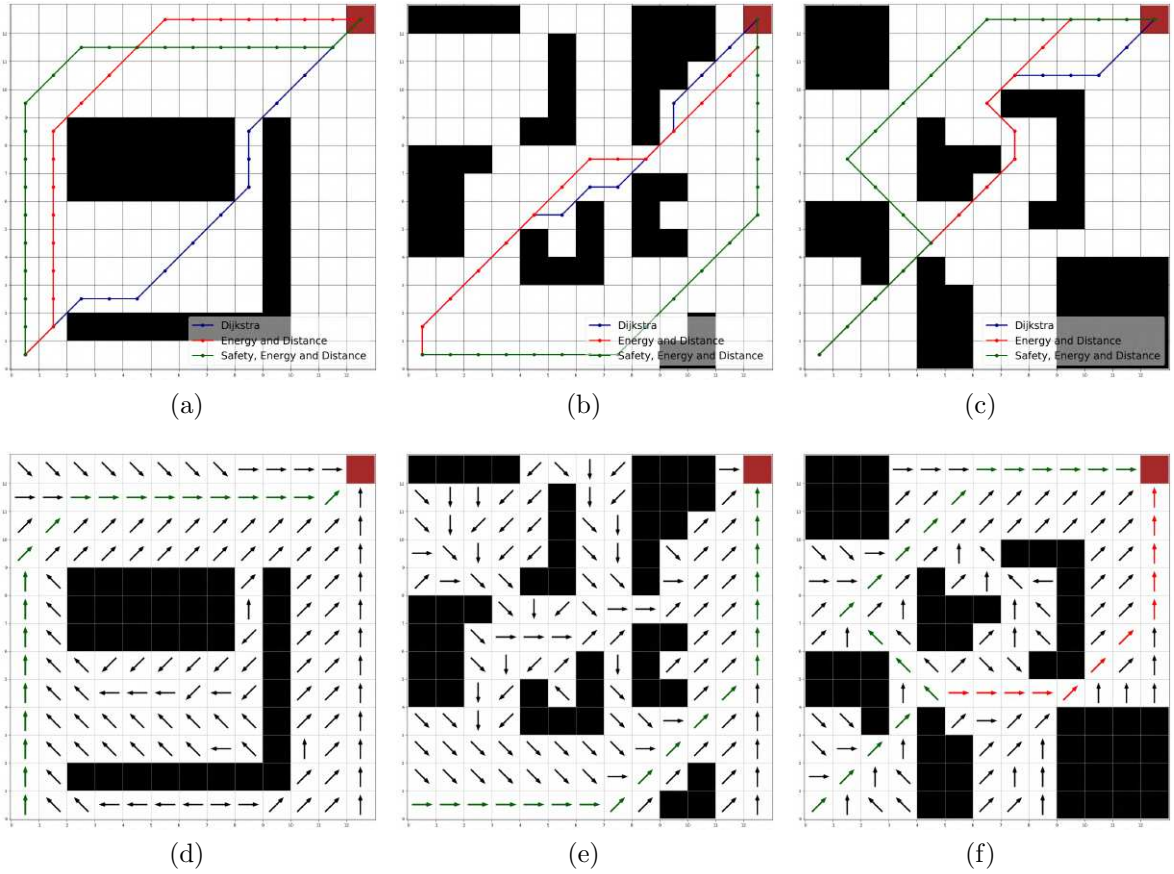


Figure 37 – Simulation Results. Images (a)-(c) display the paths taken by different priority sets as well as the shortest path obtained by Dijkstra. Images (d)-(f) display the policy obtained after training taking all priorities in account.

Table 6 – Simulation Performance Metrics

	Map 1			Map 2			Map 3		
Path	Length(m)	Avg. Dist(m)	Turns	Length(m)	Avg. Dist(m)	Turns	Length(m)	Avg. Dist(m)	Turns
Dijkstra	18.14	34.58	4	18.14	28.36	7	19.56	31.31	5
Priority Set 1	21.07	42.37	3	18.14	30.07	4	19.56	34.43	4
Priority Set 2	22.24	43.17	3	21.07	40.12	2	22.97	36.26	3

In Figure 37(a), it is possible to see that the blue path takes a more risky route than the other two. Red path represents the first set of priorities, which leverages safety and energy consumption constraints, while the green path takes in account the second set of priorities. As seen in Table 6, Dijkstra's path is the shortest one, but at the expense of safety and energy consumption, having a lower average distance to the obstacles in the map, as well as more turns than the other paths. When taking in account the first set of priorities, to be more energy efficient and take longer straight paths, the red path instead of going between the obstacles, it moves around them, accounting for a longer path, but more energy efficient. Finally, the green path adds the safety concern to its planning, taking a similar path than the first priority set, but with a added offset distance from the obstacles, accounting for a longer, but safer path, with the same amount of turns, longer length and bigger average distance of the red path. As for the resulting policy for the first map, Figure 37(d) shows that the agent can reach its destination from anywhere in the map. It is worth highlighting that when starting from between the obstacles the agent will prioritize moving far away form them maintaining a safe distance.

Map 2 is more complex than the previous and simulation for different priorities is displayed on Figure 37(b). Here Dijkstra's path has the same length as the path found with the first priority set, as seen in Table 6, as both of them do not account for safety measures, but the red path has three less turns than the blue one, leading to more energy efficient path. When safety is also taken into account on the path planning process, the resulting path is the green one, which does so while providing the best turn profile of all three paths. While it has a longer path, approximately three meters longer than the shortest ones, it has over extra ten meters in the average obstacle distance. As for the resulting policy for this map, seen in Figure 37(e), the agent can successfully reach the destination from anywhere in the grid, but it is worth pointing out that the policy shows itself quite reluctant to transverse the map going through the four obstacles in the center region, due to high risk of collisions, only committing to it after reaching the seventh row of the grid.

The third map has a higher obstacle concentration in the starting states of the map and also obstacles in the center. Figure 37(c) shows the obtained paths. To achieve the shortest route, the blue path needs to go between the two center obstacles, increasing the navigation risk. As the red path does not account for safety, it also transverses the map in a similar fashion, but presents a clear improvement in respect to its turn profile, being

able to achieve a path with the same length, but with fewer turns as seen in Table 6, but still maintains itself closer to obstacles than the green path. The second set of priorities delivers a path that hinders its path length, but is able to avoid the cluttered area in the middle of the map while achieving the best turn profile of all paths. As for the policy seen in Figure 37(f), it is possible to see that agent reaches its destination from anywhere in the map. Two paths are highlighted in this image, the green one representing the path chosen by the second set of priorities and the red one highlights the adaptability that the algorithm provides by showing that if the agent moved stray from the main path, it would choose an efficient way to leverage the chosen priorities by moving around the cluttered area in the middle of the map.

4.3.1.2 Scalability Simulations

In order to test the scalability of the proposed algorithm, the agent as trained using the second set of priorities for 100000 episodes in maps with size from $4x4$ up to $40x40$ cells with 20% of its area randomly covered by obstacles. The computer used for simulations has the following specifications: a AMD ryzen 5 5600G CPU, with 16 GB of RAM. No GPU acceleration was used in order to speed up the training process. Table 7 shows the average time consumption for different sized maps over ten repetitions.

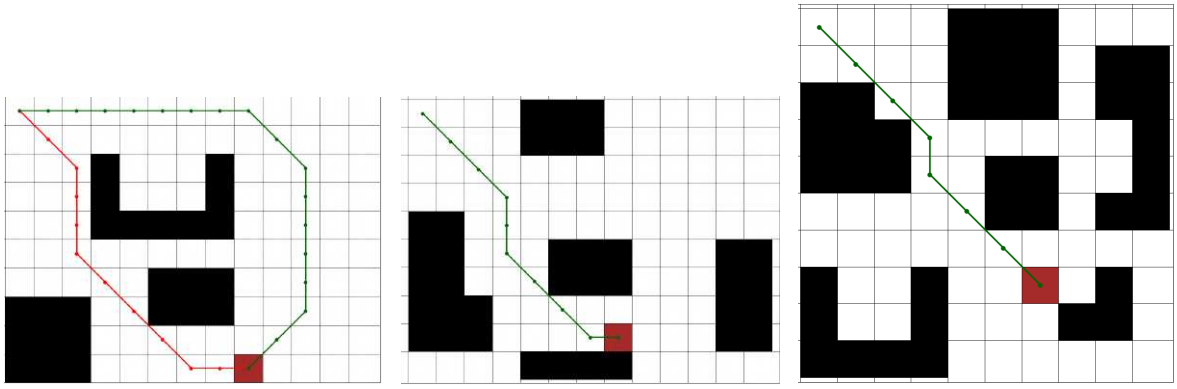
Table 7 – Average time consumed for 100 000 iterations on each map.

Size	4x4	5x5	6x6	7x7	8x8	9x9	10x10	15x15	20x20	25x25	30x30	35x35	40x40
Time(s)	4.76	7.52	7.75	6.1	9.2	12.21	8.8	17.5	16.43	16.89	24.5	28.34	33.67

4.3.2 Experiments Results

One concerning issue in some applications of robotics and artificial intelligence is the gap between execution in simulated environments with real-world scenarios. In order to validate the path planning capabilities of the proposed algorithm, three different maps were proposed to test its performance, with their design displayed in Figure 38, along with the planned paths for each of them. Due to size limitations, maps 4 and 5 are 10×13 in grid size and map 6 is 9×9, with each cell's size being 50×50 centimeters. A total of six experiments were done over the three maps with different priority sets and disturbances, as seen in Table 8.

The proposed algorithm was validated experimentally using the Bebop 2, from ParrotInc, UAV. The algorithms run in an off-board station, at a rate of 30 Hz, acquiring the poses of the vehicles through an *OptiTrack* motion capture system configured with eight cameras, and computing the reference control signals that are sent to the robots via Wi-Fi connection. Figure 39 exhibits our experimental setup.



(a) (b) (c)

Figure 38 – Maps 4,5 and 6 used for real-world experiments. Red cells represents the agent's goal.

Figure 39 – Experimental setup. Motion capture cameras are set on the upper part of the room, some of them outside the range of the photo.

For each experiment, the policy was obtained with previous training of the agent, which is indicated in Table 8. The results can be seen in full in the video <https://youtu.be/vhtvj3NSbJ0k>. In order to facilitate visualization, the experiments were recorded using two different camera angles and also a virtual version of the map in a simulated environment is provided to make it easier to associate the agent's position in the recording with respect to the grid map. For all experiments, the starting state is considered the top left cell.

In experiments 1 and 2, it is possible to see the agent successfully reaching its goal through different paths to adapt to each priority set. When taking into account

Table 8 – Sequence of Experiments.

Exp.	Time (mm:ss)	Description
1	00:17	Map 4, Energy and Distance Priorities.
2	00:42	Map 4, Safety, Energy and Distance Priorities.
3	01:14	Map 4, Safety, Energy and Distance Priorities w/ disturbance.
4	02:02	Map 5, Safety, Energy and Distance Priorities.
5	02:22	Map 5, Safety, Energy and Distance Priorities w/ disturbance.
6	03:00	Map 6, Energy and Distance Priorities.

only distance and energy consumption limitations, the agent takes the path between the obstacles while when accounting for safety as well, it goes around all obstacles, taking a longer but safer path, as seen in Figure 38(a). In this map, there is also a U-shaped obstacle, which is a common pitfall for path-planning algorithms. In experiment 3, to showcase that the policy is robust to unexpected instabilities, we simulate a mid-flight disturbance by pushing the UAV towards the internal part of the U-shaped obstacle using a joystick input disturbance, and the agent clearly is able to overcome this by following its policy and reaching its goal.

Map 5 shows a different obstacle configuration and in Experiment 4, the agent follows a path taking into account the second set of priorities. Given the constants used for training, the agent learned that moving around the obstacle in the center of the map was not worth it, due to the extra length and turns it would have to make to reach its destination. In Experiment 5, to showcase the adaptability capabilities of the trained policy, we simulate a mid-flight disturbance by pushing the UAV stray from its main path using a joystick input. Returning to the main path would be too costly in terms of total path length, safety, and energy consumption, therefore the agent's policy determines that it should move around the obstacle, successfully reaching the goal.

Ultimately, Experiment 6 was conducted on Map 6, distinguished by its smaller dimensions compared to the other maps, yet characterized by a higher density of obstacles. For this experiment, training was executed utilizing the second set of priorities. This particular priority configuration guided the agent along a path that strategically navigated through the wider opening amidst the obstacles, ultimately guiding it toward its intended objective.

The above-mentioned simulations and experiments provide robust demonstrations that the proposed path planning algorithm is capable of achieving its goal using different priority sets that can be freely tuned by the user to suit the application's needs. Also, the trained policy is capable to provide directions to the agent so that it can reach the goal from anywhere in the map, which confers robustness when it comes to possible internal or external disturbances. Furthermore, given that the proposed algorithm is

meant to be used for offline path planning scenarios, having an execution time of under 40 seconds for maps up to $40x40$ guarantees that the proposed method is feasible for practical applications. Finally, the experiments showed that the algorithm can be applied to real-world environments being successful in its performance.

4.4 3D Deep Q-Learning Global Planner Algorithm Validated with Simulations

In this section the simulation results for the proposed methods are displayed and discussed. The learning agent represents a UAV, with model and control detailed in (CARVALHO et al., 2022b).

In order to showcase the proposed algorithm's performance, training and testing in simulated environments where done. During the training, the agent was required to reach a specific destination leveraging the reward function and employed neural networks to guide its decision making. The virtual environments were discretized in square cells with a high resolution. The maps were 100m×100m×22m, with each cell having 10cm×10cm×10cm, totaling 22×10^6 cells.

The training employed a curriculum-based approach, consisting of ten levels with increasing complexity, where additional obstacles were introduced in each level. Table 9 shows the range of obstacles and the number of training episodes for each level. The distribution of training episodes for each level is heterogeneous because it is expected that an easier task, such as navigating in a simpler environment, will require fewer episodes to learn compared to a more complex one.

During training, we used a discount factor of $\gamma = 0.99$, a learning rate of $lr = 0.0004$ as the starting value for the neural network optimization, and a mini-batch of 128 samples.

Table 9 – Environment Details.

Level	**Number of Buildings**	**Number of Episodes**
1	$1 \rightarrow 3$	1000
2	$2 \rightarrow 5$	1500
3	$4 \rightarrow 6$	2000
4	$4 \rightarrow 8$	2500
5	$5 \rightarrow 9$	3000
6	$7 \rightarrow 10$	3500
7	$10 \rightarrow 13$	4000
8	$10 \rightarrow 15$	5000
9	$13 \rightarrow 18$	7500
10	$15 \rightarrow 20$	9000
Max. Number of Episodes		**40000**

With the objective of assuring the proposed algorithm's performance, quantitative results were analyzed such as the evolution of the path length, total rewards, battery levels as well as average distance to the closest obstacle. Also we use the Dijkstra algorithm as a baseline for comparison.

4.4.1 Quantitative Results

After training the agent throughout its curriculum, the metrics obtained during training are displayed in Figures 40 and 41 and in Table 10. The average reward shown in Figure 40 keep increasing during training. there is a sharp increase in the first episodes indicating basic knowledge been obtained quickly. Then, the rate in which the average reward increases is slower, but with a clear positive trend. The oscillations in the values come from changing levels, and with that a new exploration heavy phase.

Analysing Figure 41, it is possible to see a downward trend in the average final path distance. The oscillations in the image can come from the varying accumulation

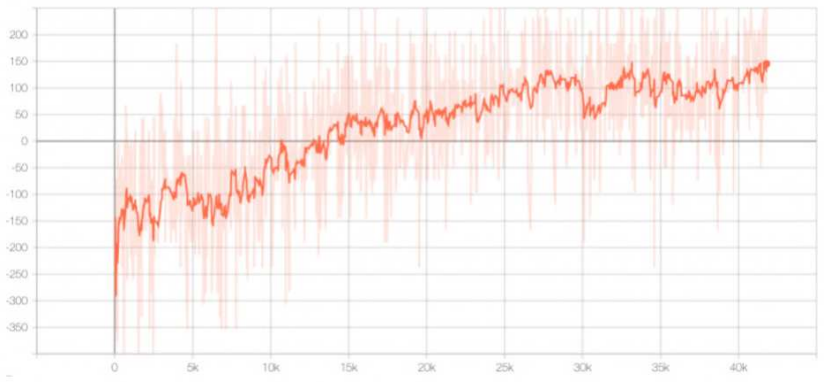


Figure 40 – Training Result for Average Reward.

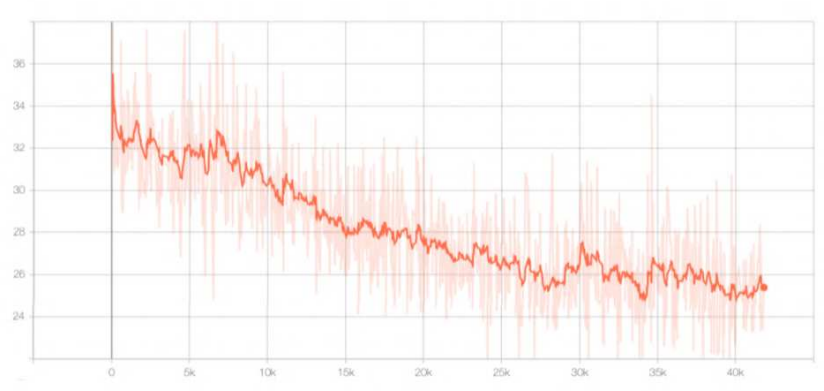


Figure 41 – Training Result for Average Distance.

Table 10 – Performance Metrics Comparison

	Dijkstra			DRL Method		
Level	**Avg. Dist. (m)**	**Energy (units)**	**Obst. Prox. (m)**	**Avg. Dist. (m)**	**Energy (units)**	**Obst. Prox. (m)**
10	64.2	2780	1.7	66.5	2895	2.3
Avg.	56.7	2355	2.2	57.3	2400	3.7

and understanding of the environment related knowledge and also for different reasons to end the episode. If the agent ends the episode due to battery levels, it will have a longer path than if it had crashed, therefore leading to these variations in the average distance measurements. But, in the end, the obtained path becomes more efficient.

In Table 10 there is a comparison on the performance metrics between the proposed and baseline algorithms on level 10 and the average among all levels. The baseline algorithm outperforms the proposed one on energy efficiency and path length, being worse on a average margin of 1.9% and a 1.05%, respectively. It is expected that Dijkstra algorithm returns the shortest path, however, as it does not take into account the path's safety. It is significantly outperformed by the proposed algorithm in the obstacle proximity metric. On average, the proposed algorithm has over 50% more distance to the closest obstacle compared to the baseline algorithm. It is worth noting that the baseline algorithm may have an advantage in this metric as it was not subject to disturbances during testing, unlike the proposed algorithm.

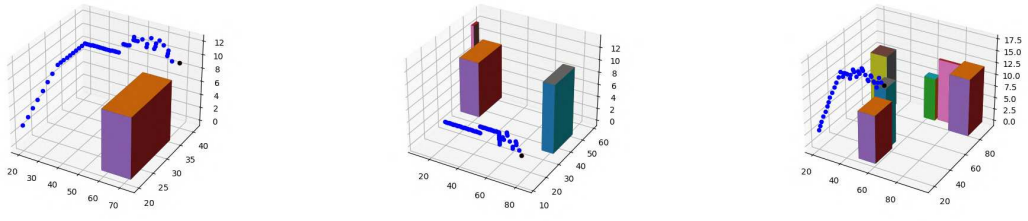
4.4.2 Qualitative Results

For a more visual demonstration of the algorithm's performance, Figures 42 and 43 show examples of a fully trained agent's paths. In the link: https://youtu.be/iczYyG5-MLw it is possible to see video footage of the virtual environment with a fully trained agent navigating as well.

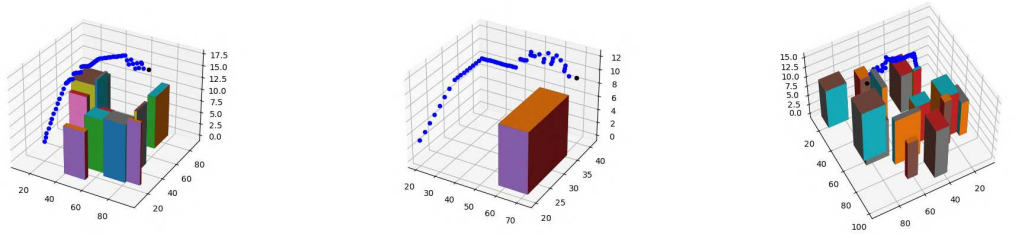
In the aforementioned Figures it is possible to see examples of paths taken by the agent in different scenarios. Some of them, such as Figure 42(a), a chaotic path is observed in the final steps of the path. This is originated by the random disturbances inserted in the proposed algorithm's simulations and it was able to react to them, reaching its destination.

Some works in the literature reduce the UAV path planning problem to a 2D or 2.5D environment, which can work in many scenarios but denies one of the major advantages of employing UAVs in a task, that is their 3D freedom of movement. Figure 43 demonstrates some examples with no disturbances with path leading the agent above some obstacles, accounting for flexibility in the planning process. The video footage provides examples for all curriculum levels, showing efficient and safe path planning.

The results demonstrate the agent can learn how to navigate safely and efficiently in different environments. The curriculum structure facilitates learning by exposing the



(a) Environment L1 (b) Environment L2 (c) Environment L4



(d) Environment L6 (e) Environment L8 (f) Environment L10

Figure 42 – Path examples in some of the environments, with the number accompanying L standing for the respective curriculum level of the image.

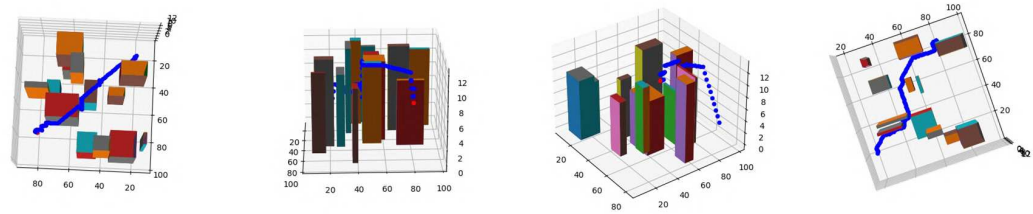


Figure 43 – Examples of 3-Dimensional Trajectory of the UAV.

agent to simple environments before it attempts to navigate in complex ones. The reward structure provided encouraging to the desired behaviors, while inhibiting the negative ones. This lead to efficient paths in the context of energy consumption, safety as well as overall distance.

The employment of Deep Reinforcement Learning techniques allowed the proposed method to deal with high dimension state spaces, with 22×10^6 possible states, where regular reinforcement learning approaches would struggle to deliver the similar results. With such a big state space, the proposed algorithm is capable of working in big areas or applied to smaller areas but with high resolution (smaller cells in the map), allowing for a high precision path planning. This can be particularly beneficial in environments with complex obstacle layouts, where high precision is necessary.

4.5 2D Local Planner Algorithm Validated with Simulations

This is an adaptation of the original global path planning algorithm so is better suited for dynamic scenarios. The proposed path planner is validated via simulations comparing to using the Dijkstra's algorithm. This subsection displays the obtained results.

4.5.1 Simulation Settings

In order to showcase the performance of the proposed algorithm, simulations were run in two different scenarios and compared to Dijkstra's algorithm as a baseline. The first scenario displays a static environment with common pitfalls of local planning algorithms such as U-shaped obstacles as well as door-like passages where the agent could be stuck. The second environment has a variety of dynamic obstacles to showcase the ability of online re-planning and navigation adaptability. The specifications of the computer used are desktop Intel® Core™ i5-4430, CPU 3.00 GHz, RAM with 16.0 GB, running Windows 10 and Ubuntu 20.04 LTS.

The proposed algorithm acts over a discrete grid representation of the agent's surroundings. When there is an obstacle inside a cell, it was considered to compromise it whole, for safety measures. Therefore, having a blacked-out cell does not mean that there is a square-shaped obstacle in it, but only that there is a relevant obstacle inside it.

The path planning algorithm used learning rate $\alpha = 0.1$, the discount factor $\gamma = 0.99$, $MinDist = 3, N = 10$. Over all simulations displayed in this paper, the average computation time was 450 ms. Allowing for online planning in the used speed profiles. With better hardware, this could be increased.

The results for the full path in both environments are shown in Figure 45. A Partial snapshot of the path obtained in the first environment is found in Figure 44. In each image, the dashed line connecting triangle way-points represents the reference path provided by the local path planning algorithm, the blue dashed line connecting circle way-points represents the local path provided by the baseline algorithm and the full black line is the actual path taken by the agent. The video of the full simulation for both environments can be seen in <https://youtu.be/8J9Mr5TblH0>

4.5.2 Simulation Results

Figure 44 showcases the local grid rotation as well as the ability to deal with common local planning pitfalls. By default, the agent starts with the biggest local grid it can fit, represented by the darkest dashed box. Given the agent could not see the U-shaped obstacle as a whole, the clear cell closest to the global goal was inside it. When the agent reaches the cell next to the local goal, it recalculates the local path. The current action

was moving upwards, but the local grid associated with this action would not yield a local map with a feasible path to the cell closest to the global goal, therefore it rotates the local grid until it finds one, which, in this case, was the local grid associated with the *right* action, guiding the agent out of the U-shaped obstacle. When it reaches the cell next to the current local goal, it recalculates the local path.

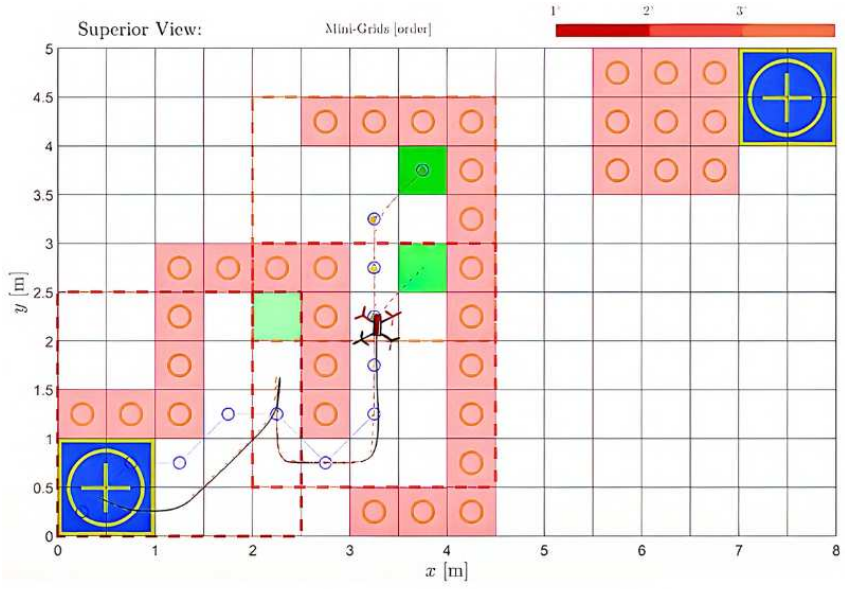
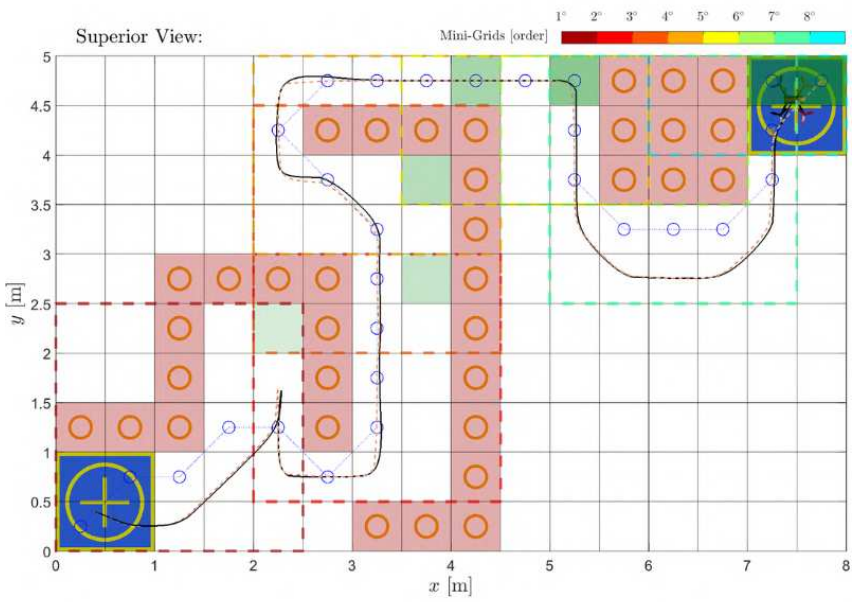


Figure 44 – Snapshot after three path re-plannings for first environment.

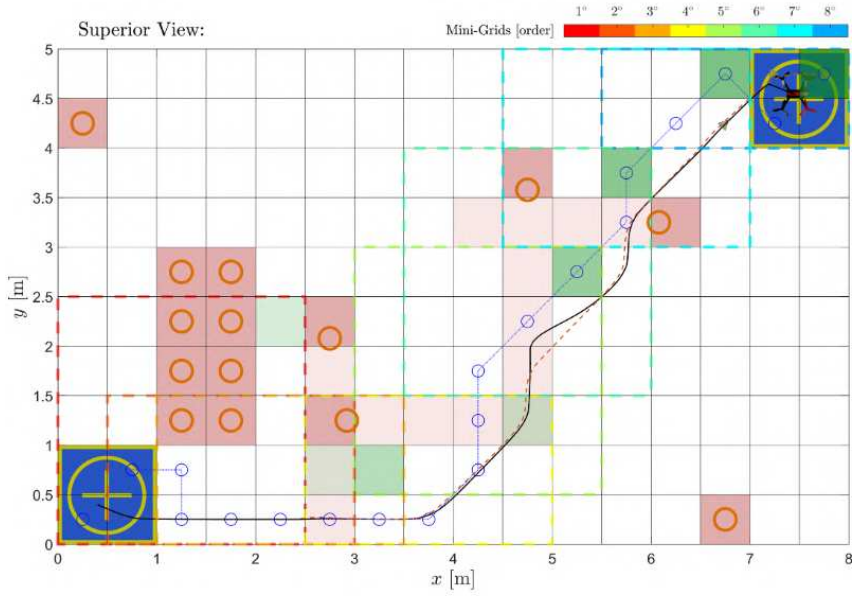
Figure 45 provides the agent's full trajectory in each of the environments. In the first environment, seen in Figure 45(a), it is possible to see when compared to the baseline algorithm, that the path outputted by our algorithm yielded a safer and smoother path. This becomes more evident in the final obstacle, where Dijkstra's algorithm uses undesired diagonal movements near the obstacle edges and also moves closer to the obstacle overall. The video https://youtu.be/8J9Mr5TblH0 provides clearer view of this process unrolling as the navigation flows with comments on the actions taken by the agent.

The second environment, as shown in Figure 45(b), it is showcased the ability to reach the global goal in dynamic environments. Most of the local goals in this image were triggered due to obstacle movement. It is worth highlighting that even though there are frequent path recalculations, the proposed algorithm manages to provide soft turns and safer routes when compared to the baseline algorithm. Given that this snapshot was taken at the end of the simulation, there are some parts of the path taken which appear near obstacles, but this only implies that the obstacle was close to the taken path at the end of the agent's navigation. In the second half of the video seen in https://youtu.be/8J9Mr5TblH0, it is possible to see more clearly the obstacle avoidance as the agent moves through the environment.

Note that, in the videos, sometimes when a new local path is provided, a lingering dashed line stays in the video, representing the rest of the previous local path. Also, in



(a) Full path for first environment.



(b) Full path for second environment.

Figure 45 – Full path for each environment. The green cells represent the local goals, with darker tone to represent the passing of time. The dotted boxes represent the local grid used to plan each local path with the color coding representing the passing of time.

the lower right part of them, there is an error measurement between the reference path and the actual path the agent is taking. It is possible to note that the control algorithm provided smooth connection and low error between the way-points provided by the local path planner, where the highest offsets happened in moments where there was path recalculation.

4.6 Path Planning in Intelligent Logistics Scenarios

This section regards the results for the multi-agent path planner for warehouse logistics applications. Different simulations using the scenario proposed in Figure 20 were done in order to validate the proposed algorithm. They consist in different combinations of RL, TL and CL to showcase the implications of each of them. It is important to make it clear that all the simulations described below involve the use of Q-Learning, strategies were incorporated in order to accelerate the learning process and task execution. The tests are described as follows:

- **Reinforcement Learning** - Baseline for comparing the other approaches. Pure $Q-$Learning was used directly on the final stage of training, with multiple agents as well as dynamic and static obstacles.
- **Curriculum learning with homogeneous episode distribution** - RL algorithm with the progressive difficulties between the four stages of training with the same amount of episodes for each one of them. When transiting from one stage to another the current $Q-$table is reused.
- **Curriculum learning with heterogeneous episode distribution** - RL algorithm with the progressive difficulties between the four stages of training. When transiting from one stage to another the current $Q-$table is reused. The amount of episodes per stage is tested with increasing number of episodes per stage as well as with a decreasing number.
- **Transfer Learning** - RL algorithm directly on one agent with the task of taking objects to the desired goals. To add new agents, TL is used in order to provide knowledge to them, accelerating the accomplishment of the desired tasks.
- **Transfer Learning and Curriculum Learning** - This test was done using all three techniques combined where instead of reusing the agent's $Q-$table from one stage to the other, TL is used to initialize the new $Q-$table. A progressive number of episodes per stage is used in this test.
- **Multiple Agents Scalability** - In order to test the scaling capabilities of the proposed algorithm, the same environment (Figure 20) is used for simulations involving 2 up to 7 agents.

For each simulation, the number of static obstacles (operation desks and shelves), dynamic obstacles (agents, humans), grid size and destination goals are summarized in Table 11.

Table 11 – Environment Details

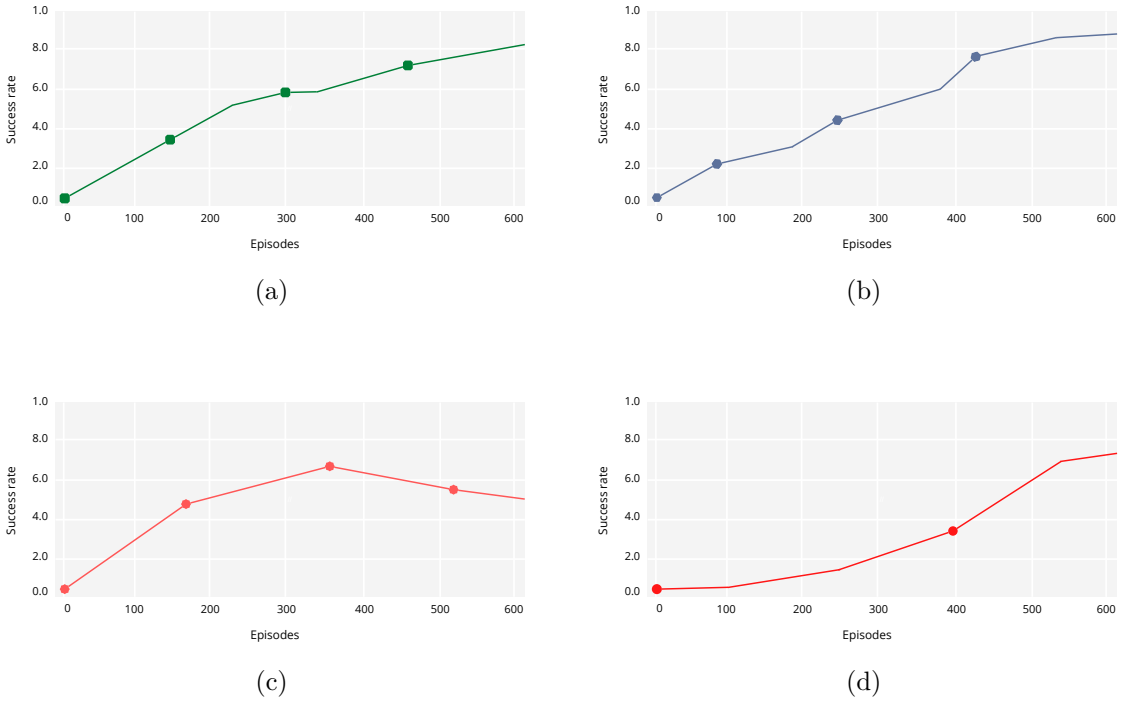
	Grid Size	Agents	Humans	Operations Desks	Shelves	Goals
Environment Details	21 × 21	3	2	3	20	3

The next sections present the results for the proposed simulations and a general discussion of the obtained results.

4.6.1 Curriculum Learning Tests

There are three tests done using CL alone, the first one with the same number of episodes for each stage of training, the second one with a increasing number of episodes for each stage, and the third with a decreasing number of episodes per stage. The number of episodes per stage is 1500 for the first test, 900, 1400, 1800 and 1900 for the second, and 1900, 1800, 1400, and 900 for the third.

The mean of the success rate for the last 10 episodes in each stage of each test can be seen in Figure 46. Each marker represents the beginning of an stage.



(a) (b)

(c) (d)

Figure 46 – Mean of the success rate of the last 10 episodes.

It is evident from analyzing the presented findings in Figure 46(c) that using fewer learning episodes in the more complicated stages produces subpar performance. In contrast to the other two stages, the third one, where dynamic obstacles are added, it is possible to see a decline in success rate, resulting in a success rate of only 52% overall.

Comparing a homogeneous distribution in number of episodes as seen in Figure 46(a) with the increasing number of episodes per stage on Figure 46(b), it is possible to notice that both have success rate growth, but the homogeneous number of episodes falls short on the overall performance of the last stage, reaching 82% success rate while the heterogeneous approach reaches 89%.

4.6.2 Transfer Learning Test

In this test, a single agent is trained directly in the environment with both static and dynamic obstacles over the course of 4000 episodes. Then, TL is used to incorporate the new agents into the system and to prepare for the next 2000 episodes. In Figure 46(d), it is possible to view the mean success rate for the last 10 episodes in each stage in this test. With an overall performance of 74%, the success rate keeps rising as more agents are introduced to the environment.

4.6.3 Transfer Learning and Curriculum Learning Test

This test is done progressively increasing the difficulty of the tasks throughout four stages of training and at the beginning of each new stage, the $Q-$table is initialized using TL with the past stage as knowledge source, as shown in Figure 48. As the results with heterogeneous and homogeneous episode distribution in the CL tests show, using an higher number of episodes for more complex stages yields better results, therefore this test uses the same number of episodes per stage as the third CL test.

The results obtained combining curriculum, transfer and reinforcement learning outperformed all the other approaches, reaching a 94% overall success rate. The simulation in stage 4, after used transfer learning to two other agents, can be viewed at: <https://www.youtube.com/shorts/fB-7-QZDsY8>.

4.6.4 Scalability Tests

The warehouse environment is tested with 2 up to 7 agents. For each simulation, it is requested that 10 objects where taken from the operation desk to a specific goal. The time requires to complete this task is measured for each number of agents using curriculum learning alone or the proposed method combining CL with TL. Each simulation is repeated 15 times. The results can be seen in Figure 47.

Curriculum learning alone has better performance for 2 and 3 agents, but the proposed method outperformed it for more agents. This could mean that the benefits of TL are more significant when more agents are acting together.In addition, these tests show that the proposed method can be scaled 2 up to 7 agents.

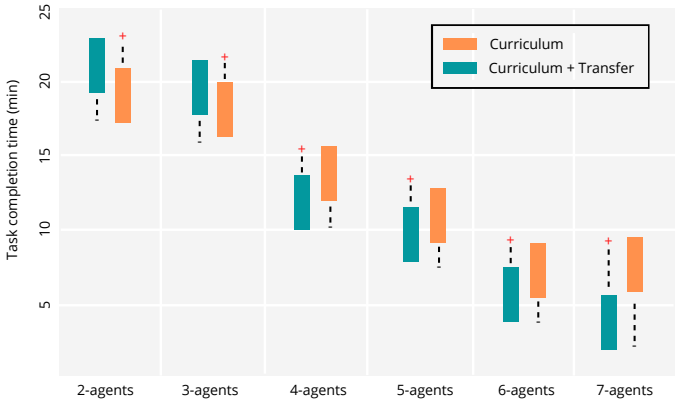


Figure 47 – Scalability test in warehouse scenario with 2 up to 7 agents.

4.6.5 General Results Discussion

A summary of the results can be found in Table 12. In order to facilitate the visualization of the results, some markers were added to the graph in Figure 48. Each marker represents a change in the training stage. For tests that involved the CL curriculum strategy, at these points, there is an increase or decrease in the number of episodes.

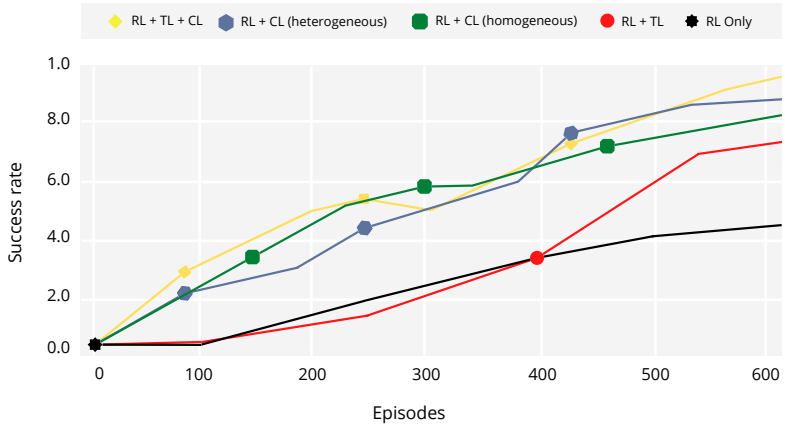


Figure 48 – Reinforcement learning training results.

Table 12 – The success rates of tasks according to training episode lengths

Learning type	**The Number of Training Episodes**	**Success rate**
TL + CL	(900, 1400, 1800, 1900)	0.94
TL only	(4000, 2000)	0.74
Homogeneous CL	(1500, 1500, 1500, 1500)	0.82
Heterogeneous CL	(900, 1400, 1800, 1900)	0.89
	(1900, 1800, 1400, 900)	0.52
RL only	6000	0.48

It is clear that the proposed method outperforms other combinations of techniques as well as the baseline $Q-$Learning algorithm. Analyzing the graphs, it is possible to see

that even when changing to more complex scenarios, the $Q-$table initialization provided by the applied transfer learning technique, allows for further success rate increase.

Comparing the performance of the simulations using curriculum learning with the ones it does not, it is possible to see that presenting easier environments before attempting training in more complex ones can sharply increase and accelerate the overall success rate when compared to the simulations that did not use this approach.

Finally, it is possible to notice that the red graph (RL+TL) has a similar shape as the black one (RL only), until it adds more agents and transfer the knowledge of the previous agent to them before continuing training.

4.7 State Representation Selection for Mapless Autonomous Navigation with Deep Reinforcement Learning

To assess the impact of different state representations in terms of the number of sections and the quantization method, training and testing were conducted. Training was performed in the First environment, while testing was carried out across all three environments. Additionally, specific metrics were defined to analyze the results, as detailed in the next subsection.

After training is complete, each model (for each DSRM) is put to test in order to compare performance based on the previously defined metrics. During this phase, no training is allowed, and only the Selector θ is used for navigation.

Sequentially, 1k episodes in each one of 5 tests were performed for each of the 12 DSRMs, targeted at reaching the five goals presented in the simulation environments (Figure 21). Specifically, the tests were conducted with the following pairs of bases and goals:

$$\text{Test 1 - Environment 1} : (B1) \rightarrow (G1)$$
$$\text{Test 2 - Environment 1} : (B1) \rightarrow (G2)$$
$$\text{Test 3 - Environment 2} : (B2) \rightarrow (G3)$$
$$\text{Test 4 - Environment 2} : (B2) \rightarrow (G4)$$
$$\text{Test 5 - Environment 3} : (B3) \rightarrow (G5)$$

Test 1 mirrors the validation steps in training, whereas Test 2 verifies that no overfit occurred, as they are set at the training Environment 1. Test 3 and 4 check the resulting generalization achieved by the model, since they are set in an unseen and larger Environment. Finally, Test 5 regards the ability of the model to minimize steps towards reaching the goal, as Environment 3 is similarly shaped to Environment 1, but contains no objects.

4.7.1 Analyzed Metrics

Based on similar literature, a selection of metrics was used for defining the success and collision rate, as well as measures of safety and energy efficiency during navigation.

The first one defined the Total Distance Traveled by the agent, which measures the efficiency based on path length. The sum of the distance travelled in each iteration results in the TDT over an episode, and is defined as:

$$\text{TDT(Episode)} = \sum_{i=2}^{N_{iter}} \|\xi_i - \xi_{i-1}\| \tag{4.1}$$

where ξ_i and ξ_{i-1} are respectively the current and last positions of the agent at each iteration step.

Success Rate (SR), a common used metric in the literature, is the average of the amount of episodes where the agent reaches the vicinity of the goal. The Final Distance To Goal, a support measure, is defined as:

$$\text{FDTG}(\xi_F, \xi_G) = \|\xi_F - \xi_G\| \tag{4.2}$$

where ξ_F and ξ_G are respectively the agent final position and the position of the current goal. Based on the agent's FDTG, the Success Rate is counted and averaged over episodes of each DSRM as:

$$\text{SR(DSRM)} = \frac{1}{N_{ep}} \sum_{i=2}^{N_{ep}} \left\{ \begin{array}{ll} 1 & \text{, if FDTG} \leq d_{reach} \\ 0 & \text{, if FDTG} > d_{reach} \end{array} \right\} \tag{4.3}$$

where d_{reach} is the minimal distance to the goal that should be considered a Success, used as 0.5 m in this work.

To evaluate the safety of the navigation the metric Mean Distance To Obstacles (MDTO) is defined by averaging the minimal distance that the agent read to objects within the environment, computed as:

$$\text{MDTO(Episode)} = \frac{1}{N_{iter}} \sum_{i=1}^{N_{iter}} \min_{d \in \mathcal{S}_d}(d) \tag{4.4}$$

where d is the distance measured and quantized in each state in $\mathcal{S}_d$, whose size depends on the DSRM in use.

and finaly, Collision Rate (CR), similarly to the SR is the average of the amount of episodes where the agent collides into objects, mathematically defined as:

$$\text{CR(DSRM)} = \frac{1}{N_{ep}} \sum_{i=1}^{N_{ep}} \left\{ \begin{array}{l} 1, \text{ if } \min_{d \in \mathcal{S}_d}(d) \leq d_{safe} \\ 0, \text{ if } \min_{d \in \mathcal{S}_d}(d) > d_{safe} \end{array} \right\} \tag{4.5}$$

where d_{safe} is the minimal distance 0.3 m in the algorithms within this work.

4.7.2 Training Convergence and Validation

The algorithm for training and testing the 12 DSRMs over 25k episodes was applied sequentially, and the present section discusses the convergence analysis during validation loops. Next, a comparison of the performance of each model is presented based on standard metrics.

During training, performance data was measured for 10 episodes after each 100 episodes of training. For each of the 12 DSRMs, the reward value for each of theses episodes is presented in Figure 49. Each plot presents the results for a given sampling method, comparing the results for a increasing number of sectors. Successful model training is seen in Figure 49 as progressive growth towards positive values. In contrast, a model that fails to learn the task displays stagnant values over episodes, as seen for the DSRM mode with 10 sectors.

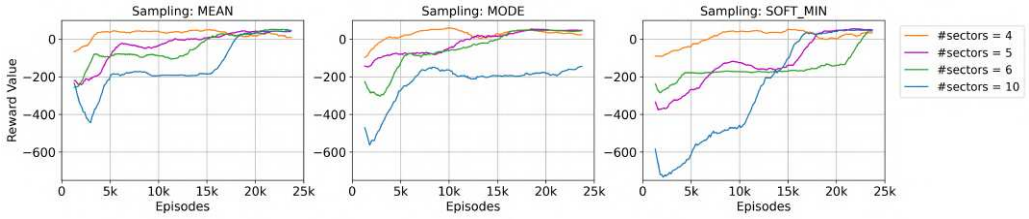


Figure 49 – Rewards results are shown for each DSRM in terms of sampling and sector number, measured during validation loops within the training stage. Initial rewards are mostly negative, arising from the collision and step reward functions, and converge to positive values as the model learns to succeed in collision-free mapless navigation. Note that mode10 DSRM fails to converge, reaching a learning plateau at -200 reward points.

For the every sampling method, Figure 49 shows that DSRMs with 4 sectors reach positive rewards faster, before 8k episodes. In addition, DSRMs with 5 and 6 sectors converged in after 15k to 20k episodes in most cases and successfully converged. Moreover, DSRMs with 10 sectors begin training with the most negative amount of rewards, and required more episodes to converge than other methods. Furthermore, the DSRM "mode_10" failed to converge, as seen on the blue line of the top-right plot in Figure 49, reaching a plateau after 10k episodes. It is possible that adjusting hyperparameters such as the DDQN architecture, ϵ decay, learning rate decay, maximum number of episodes and the episode interval between training loops may result in convergence of the "mode_10" DSRM model.

Overall, increasing the number of sectors resulted in a increased number of episodes for achieving stable positive results, representing a stage where navigation is successful. In terms of success and collision rates, similar behaviour to the reward convergence is observed, where the collision rate tends to 0% as the success rate tends to 100% (Figure 50) at the same episode range where rewards converged to positive values. In short, most

DSRM converged and reached similar final rewards, SR and CR at later episodes (20k to 25k).

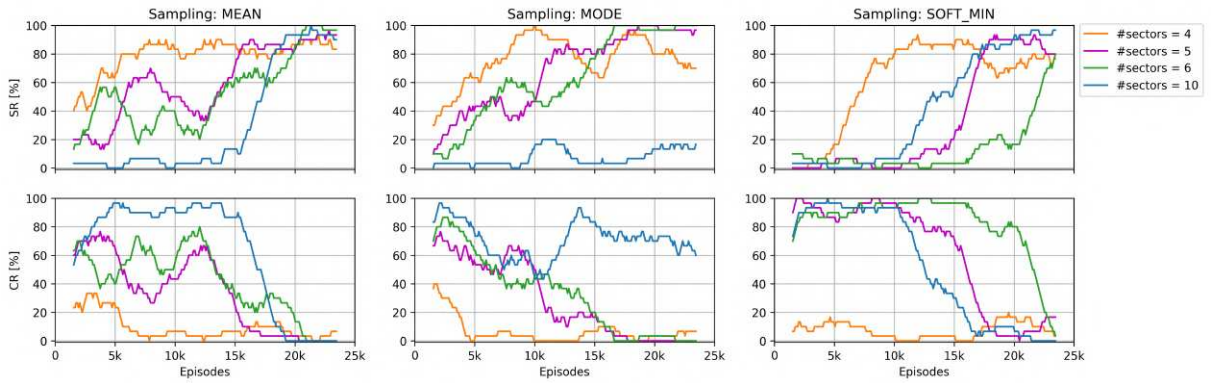


Figure 50 – Success and collision rates are seen for each DSRM over 25k training episodes. Model training is shown to result in less collisions and more successes as desired. As in Fig. 49, mode10 failed to converge as noted as low SR and high CR values.

4.7.3 Performance Testing

Upon completion of the training process, each of the 12 models were applied to five tests as presented in the beggining of this section, resulting in a total of 1k episodes of navigation in a mix of seen and unseen base points, goal points and environments. The mean value across all five tests was computed in a centered rolling window of 25 episodes. Figure 51 presents the performance achieved by each model, where the mode of the result is shown in red and the standard deviation is colored based on the number of sectors of each DSRM, for ease of comparison. In addition, Table 13 presents a numerical summary of the data show in Figure 51. Therefore, in short, DSRMs using mean sampling display consistent performance progression such that mode10 is the best performing model, despite similarities to the softmin5 DSRM in Success and Collision Rate.

In terms of best performance, the "soft_min" DSMR with 5 sectors displayed the smallest standard deviation across SR and CR, achieving 92% mean success rate over all tests, despite not presenting the best TDT as seen in Figure 51. Furthermore, this method achieved the highest mean MDTO, meaning that the agent is able to keep a larger distance to obstacles in general while successfully fulfilling the task.

The second-to-best performance was achieved by the "mean" DSMR with 10 sectors, resulting in 90% mean success rate, despite larger standard deviation when compared with the "soft_min5" DSRM. Regarding the "mean" sampling method over its variations, notice in Figure 51 that the SR and CR increases monotonically as more sectors are added, which is not observed for other measures. In addition, the "mean" sampling method achieved similar mean TDT regardless of the number of sectors, but an increasing mean MDTO as the number of sectors increases.

Table 13 – Median of test metrics averaged over 1k episodes for each DSRM demonstrate that mode sampling displays the most consistent adjustment, such that mode10 is the ideal DSRM with similar results to the top-performing model (*softmin5*).

DSRM	Success Rate [%]	Collision Rate [%]	Total Distance Traveled [m]	Mean Distance To Objects [m]
mean4	29	67	19.66	1.29
mean5	65	34	21.23	1.31
mean6	70	28	20.41	1.30
***mean10**	**90**	**10**	**21.18**	**1.34**
softmin4	30	60	27.47	1.21
softmin5	***92***	***5***	***22.80***	***1.43***
softmin6	69	22	20.07	1.20
softmin10	81	15	20.89	1.37
mode4	27	73	21.23	1.24
mode5	40	60	20.17	1.32
mode6	10	87	21.59	1.23
mode10	0	98	15.14	0.97

In relation to the "mode" sampling method, mirroring training results, the metrics show that this method presents the least stable performance, displaying high standard deviation across all measures. Also, note that the "mode6" DSRM failed to navigate new environments (SR = 1%, CR = 96%), despite having shown convergent behaviour in training. Similarly, the "mode10" DSRM resulted in the worst mean performance across all metrics, having failed to converge during training.

Overall, the mode sampling method shows overfit behaviour, having low success and high collision rates. The "soft_min" sampling method achieved the top performance across all DSRM when using 5 sectors, but has high variability when adding or removing sectors, making it a unstable and counter-intuitive sampling method.

Moreover, the "mean" sampling method showed very similar results in its best

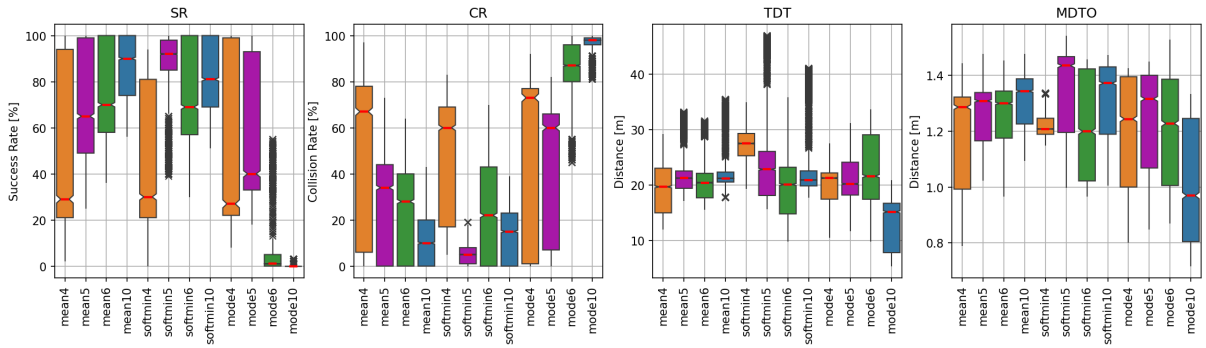


Figure 51 – Model performance in 5 tests is averaged over 1k episodes and measured on Success Rate (SR), Collision Rate (CR), Total Distance Traveled (TDT) and Mean Distance to Objects (MDTO) from left to right. Increasing sector number resulted in SR and decreasing CR for the mean sampling methods, while softmin and mode had varying results. The top individual performing model is softmin5, resulting in the highest SR, lowest CR and highest MDTO, representing respectively a successful and safe model. However, mean sampling is considered the best representation methodology following from its performance increase consistency based on sector number adjustment.

performing DSRM, and sector addition monotonically improves performance across mean metric results. Note, also, that the "mean10" results yield similar performance to the "soft_min5" in all metrics, and as expected, "mean10" displays lower TDT at the cost of a higher MDTO.

In practice, this means that the "mean" sampling method is a more stable DSRM to tune and has the overall best convergence adjustment. Still, it requires more attention to safety guarantees, since its best MDTO tends to be worse than the best "soft_min".

From these empirical results, choosing the mean sampling method achieves the most reliable performance for fine-tuning DSRMs, resulting in monotonically improving metrics as more sectors are added. Still, the "soft_min" sampling methods can yield satisfactory results, at the cost of a less intuitive hyperparameter tuning. The most clear trade-off is that using the "mean" sampling method will not guarantee the most optimal safety performance (higher MDTO when compared with "soft_min"), but yields better energy efficiency with similar or identical success and collision rates.

5 Concluding Remarks

This work proposes to use Reinforcement Learning techniques in mobile robotics scenarios. The contributions done include a global path planning algorithm that can be tailored for each application in respect of the intensity of each priority it takes in account between path length, distance and energy consumption. This algorithm was further improved and adapted to be used in dynamic scenarios as well. There was also contributions done in logistic applications in warehouse scenarios, with an initial approach to multi-agent solutions for this type of environment. Finally, the effects of different state representation using depth sensors in mapless navigation is discussed. These contributions, as well as the published survey on machine learning applications in UAV navigation, are summarized in which is summarized in Figure 52.

When it comes to the global path planning algorithm, it was originally proposed for 2D scenarios and it was latter expanded to be applied in 3D maps as well. It was validated via simulations and also experiments to highlight its performance. The research community approved the proposed approach by accepting five manuscripts, so far, in international conferences and journals. Analyzing the results from each produced paper, it is possible to see a clear evolution in performance metrics such as the path themselves or the training time needed to achieve satisfactory results. Given that the algorithm is meant to be used in offline planning, the final execution time allows it to be used in real world applications.

In the context of local planning, the algorithm was able to leverage the three priorities seen in the global path planner, but adapting it to the local planning context being able to avoid common pitfalls to this type of algorithm such as U-shaped an narrow or door-like passages.

Although the above proposed algorithm used UAV applications as validation scenarios for their performance, it is important to note that they can also be used for path planning in ground circumstances. Given their limited flying autonomy, UAVs are more seriously limited in regards to energy consumption, yet ground robots can still profit from it. Energy consumption rate will be more effective with a smoother velocity profile. Less frequent and abrupt turns will reduce the occurrence of motion blur, which is an image distortion caused by movement. This is because the outputted paths when the energy consumption priority is taken into consideration have longer straight lines.

Given that there are not many studies in the literature allowing for multi-agent solutions for this application, the initial findings for the multi-agent path planning method for warehouse situations are encouraging. Working with up to seven agents, the proposed algorithm was able to attain excellent success rates. Other methods, such as Transfer and

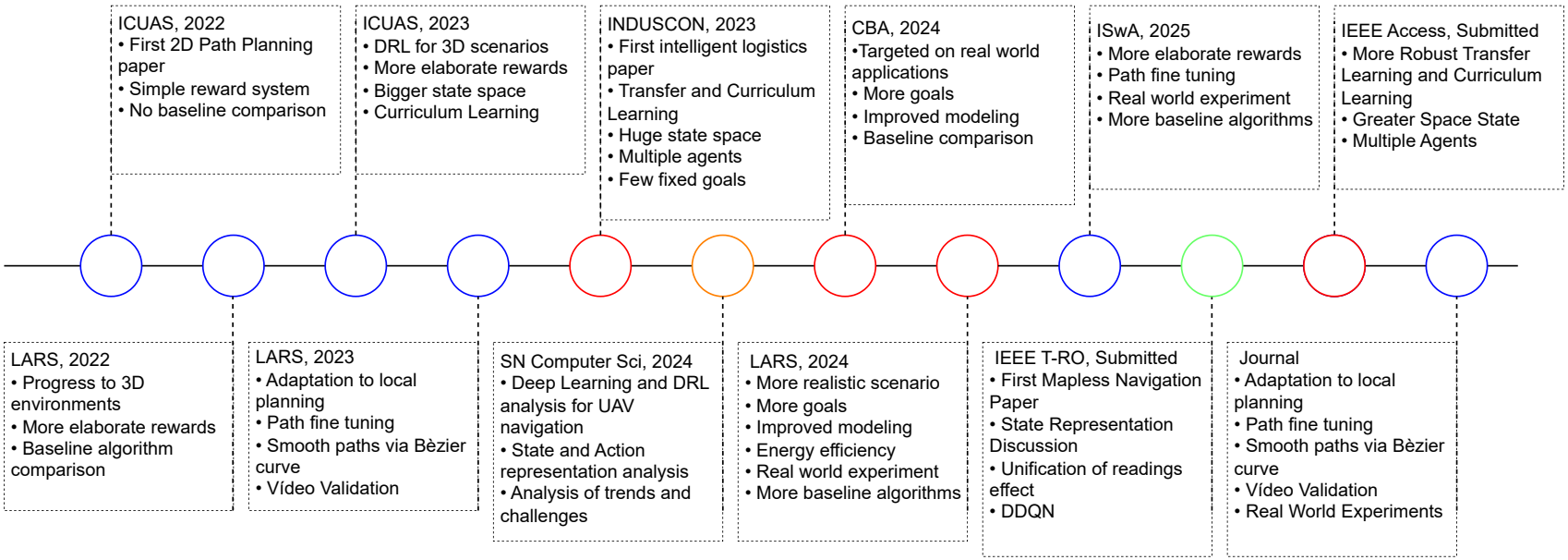


Figure 52 – Timeline illustrating the progression of works developed throughout this research. Blue circles correspond to single-agent path planning contributions, red circles indicate works related to intelligent logistics applications, the orange circle represents the survey on machine learning techniques for UAV navigation, and the green circle highlights the mapless navigation contribution. The last paper is still to be submitted.

Curriculum Learning, were used to speed up convergence rates and overall performance because the problem has a state space dimension.

This thesis presents simulations and discussions on state representation, hyperparameter selection, and Q-table initialization to provide insights for new researchers. It examines how different quantizations and segmentations of LiDAR readings impact mapless navigation and explores the effects of varying hyperparameters, such as the learning rate and explore/exploit ratio, in the context of a global path planning algorithm. These contributions aim to guide readers in the early stages of developing and modeling reinforcement learning problems.

It is worth highlighting that most of the aforementioned results were obtained using an easy to implement algorithm, i.e, Q-Learning, as base for its development.

Bibliography

ABBAS, M. et al. Alexnet based real-time detection and segregation of household objects using scorbot. In: IEEE. *2020 4th international conference on computational intelligence and networks (CINE)*. Kolkata, India, 2020. p. 1–6. Cited on page 43.

AKROUR, R. et al. Regularizing reinforcement learning with state abstraction. In: IEEE. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, 2018. p. 534–539. Cited 2 times on pages 31 and 32.

ALMISREB, A. A.; JAMIL, N.; DIN, N. M. Utilizing alexnet deep transfer learning for ear recognition. In: IEEE. *2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP)*. Madrid, Spain, 2018. p. 1–5. Cited on page 43.

ARAI, T. et al. Reinforcement learning based on state space model using growing neural gas for a mobile robot. In: IEEE. *2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS)*. Toyama, Japan, 2018. p. 1410–1413. Cited 2 times on pages 31 and 32.

ARINEZ, J. F. et al. Artificial intelligence in advanced manufacturing: Current status and future outlook. *Journal of Manufacturing Science and Engineering*, American Society of Mechanical Engineers Digital Collection, v. 142, n. 11, 2020. Cited on page 24.

ARMBRUST, C.; CUBBER, G. D.; BERNS, K. Icarus–control systems for search and rescue robots. *Book on Field and Assistive Robotics*, 2014. Cited on page 22.

ASSISTIVE-2022. 2022. <http://asblab.mie.utoronto.ca/research-areas/assistive-robotics>. Accessed: 2022-11-29. Cited on page 22.

BAIRD, L.; MOORE, A. Gradient descent for general reinforcement learning. *Advances in neural information processing systems*, v. 11, 1998. Cited on page 41.

BARBEHENN, M. A note on the complexity of dijkstra's algorithm for graphs with weighted vertices. *IEEE transactions on computers*, IEEE, v. 47, n. 2, p. 263, 1998. Cited on page 30.

BATISTA, H. de O. et al. Multi-goal robot path planning based on q-learning for library logistics. Cited on page 35.

BATISTA, H. O. et al. Q-learning-based multi-objective global path planning for ugv navigation. In: IEEE. *2024 Latin American Robotics Symposium (LARS)*. Arequipa, Peru, 2024. p. 1–6. Cited on page 35.

BOUHAMED, O. et al. Autonomous uav navigation: A ddpg-based deep reinforcement learning approach. In: IEEE. *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. Seville, Spain, 2020. p. 1–5. Cited on page 26.

BRO, R.; SMILDE, A. K. Principal component analysis. *Analytical methods*, Royal Society of Chemistry, v. 6, n. 9, p. 2812–2831, 2014. Cited on page 25.

BRUNNBAUER, A. et al. Model-based versus model-free deep reinforcement learning for autonomous racing cars. *arXiv preprint arXiv:2103.04909*, 2021. Cited on page 39.

CAO, Z. et al. Deep transfer learning mechanism for fine-grained cross-domain sentiment classification. *Connection Science*, Taylor & Francis, v. 33, n. 4, p. 911–928, 2021. Cited on page 27.

CARVALHO, K. B. de; BASÍLIO, V. T.; BRANDÃO, A. S. Action recognition for educational proposals applying concepts of social assistive robotics. *Cognitive Systems Research*, Elsevier, v. 71, p. 1–8, 2022. Cited on page 34.

CARVALHO, K. B. de et al. A 3d q-learning algorithm for offline uav path planning with priority shifting rewards. In: IEEE. *2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE)*. São Bernardo do Campo, Brazi, 2022. p. 169–174. Cited 2 times on pages 34 and 35.

CARVALHO, K. B. de et al. Q-learning global path planning for uav navigation with pondered priorities. *Intelligent Systems with Applications*, Elsevier, p. 200485, 2025. Cited on page 35.

CARVALHO, K. B. de et al. Q-learning based local path planning for uavs with different priorities. In: IEEE. *2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE)*. Salvador, Brazil, 2023. p. 89–94. Cited 2 times on pages 34 and 35.

CARVALHO, K. B. de; OLIVEIRA, I. R. L. de; BRANDÃO, A. S. Av navigation in 3d urban environments with curriculum-based deep reinforcement learning. In: IEEE. *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*. Warsaw, Poland, 2023. p. 1249–1255. Cited 2 times on pages 34 and 35.

CARVALHO, K. B. de et al. Q-learning based path planning method for uavs using priority shifting. In: IEEE. *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. Dubrovnik, Croatia, 2022. p. 421–426. Cited 2 times on pages 34 and 35.

CARVALHO, K. B. de et al. Gestures-teleoperation of a heterogeneous multi-robot system. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 118, n. 5, p. 1999–2015, 2022. Cited 2 times on pages 34 and 92.

CAUBRIÈRE, A. et al. Curriculum-based transfer learning for an effective end-to-end spoken language understanding and domain portability. *arXiv preprint arXiv:1906.07601*, 2019. Cited on page 27.

CELEBI, M. E.; AYDIN, K. *Unsupervised learning algorithms*. Berlin/Heidelberg, Germany: Springer, 2016. Cited on page 25.

CHAKRABORTY, A.; BANERJEE, J. S. An advance q learning (aql) approach for path planning and obstacle avoidance of a mobile robot. *International Journal of Intelligent Mechatronics and Robotics (IJIMR)*, IGI Global, v. 3, n. 1, p. 53–73, 2013. Cited on page 31.

CHEN, G. et al. What should be the input: Investigating the environment representations in sim-to-real transfer for navigation tasks. *Robotics and Autonomous Systems*, Elsevier, v. 153, p. 104081, 2022. Cited on page 66.

CHEN, X.; LI, Y.; LIU, L. A coordinated path planning algorithm for multi-robot in intelligent warehouse. In: IEEE. *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. Dali, China, 2019. p. 2945–2950. Cited on page 32.

CLABAUGH, C. et al. Long-term personalization of an in-home socially assistive robot for children with autism spectrum disorders. *Frontiers in Robotics and AI*, Frontiers Media SA, v. 6, p. 110, 2019. Cited on page 22.

CUNHA, B. et al. Deep reinforcement learning as a job shop scheduling solver: A literature review. In: SPRINGER. *International Conference on Hybrid Intelligent Systems*. Porto, Portugal, 2018. p. 350–359. Cited on page 38.

DAS, P. K. et al. Extended q-learning algorithm for path-planning of a mobile robot. In: SPRINGER. *Asia-Pacific Conference on Simulated Evolution and Learning*. Kanpur, India, 2010. p. 379–383. Cited on page 40.

DAS, P. K. et al. An improved q-learning algorithm for path-planning of a mobile robot. *International Journal of Computer Applications*, Citeseer, v. 51, n. 9, 2012. Cited on page 39.

DAY, O.; KHOSHGOFTAAR, T. M. A survey on heterogeneous transfer learning. *Journal of Big Data*, Springer, v. 4, n. 1, p. 1–42, 2017. Cited 4 times on pages 27, 41, 42, and 43.

DUBÉ, R. et al. 3d localization, mapping and path planning for search and rescue operations. In: IEEE. *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. Lausanne, Switzerland, 2016. p. 272–273. Cited on page 21.

EVJEMO, L. D. et al. Trends in smart manufacturing: Role of humans and industrial robots in smart factories. *Current Robotics Reports*, Springer, v. 1, n. 2, p. 35–41, 2020. Cited on page 22.

FAGUNDES-JUNIOR, L. A. et al. Machine learning for unmanned aerial vehicles navigation: An overview. *SN Computer Science*, Springer, v. 5, n. 2, p. 256, 2024. Cited 3 times on pages 35, 37, and 66.

FALLOOH, N. et al. Robot path planning using enhanced q-learning algorithm based on single parameter. *Engineering and Technology Journal*, University of Technology-Iraq, p. 1–15, 2025. ISSN 1681-6900. Disponível em: <https://etj.uotechnology.edu.iq/article_186176.html>. Cited on page 26.

FAROUKI, R. T. *Pythagorean—hodograph Curves*. New York, USA: Springer, 2008. Cited on page 59.

FERGUSON, D.; STENTZ, A. The delayed d* algorithm for efficient path replanning. In: IEEE. *Proceedings of the 2005 IEEE international conference on robotics and automation*. Barcelona, Spain, 2005. p. 2045–2050. Cited on page 30.

FLORENSA, C. et al. Reverse curriculum generation for reinforcement learning. In: PMLR. *Conference on robot learning*. Mountain View, California, USA, 2017. p. 482–495. Cited on page 45.

FU, M. C. Simulation-based algorithms for markov decision processes: Monte carlo tree search from alphago to alphazero. *Asia-Pacific Journal of Operational Research*, World Scientific, v. 36, n. 06, p. 1940009, 2019. Cited on page 39.

FUSIC, S. J.; RAMKUMAR, P.; HARIHARAN, K. Path planning of robot using modified dijkstra algorithm. In: *2018 National Power Engineering Conference (NPEC)*. Madurai, India: IEEE, 2018. p. 1–5. Cited on page 30.

GARIVIER, A.; MOULINES, E. On upper-confidence bound policies for switching bandit problems. In: SPRINGER. *International Conference on Algorithmic Learning Theory*. Espoo, Finland, 2011. p. 174–188. Cited on page 38.

GONG, C. et al. Multi-modal curriculum learning for semi-supervised image classification. *IEEE Transactions on Image Processing*, IEEE, v. 25, n. 7, p. 3249–3260, 2016. Cited on page 27.

GOPALAN, R.; LI, R.; CHELLAPPA, R. Domain adaptation for object recognition: An unsupervised approach. In: IEEE. *2011 international conference on computer vision*. Barcelona, Spain, 2011. p. 999–1006. Cited on page 27.

GRAF, T.; PLATZNER, M. Adaptive playouts in monte-carlo tree search with policy-gradient reinforcement learning. In: SPRINGER. *Advances in Computer Games*. Maastricht, Netherlands, 2015. p. 1–11. Cited on page 40.

GRONDMAN, I. et al. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, IEEE, v. 42, n. 6, p. 1291–1307, 2012. Cited 2 times on pages 40 and 41.

HARWIN, L.; SUPRIYA, P. Comparison of sarsa algorithm and temporal difference learning algorithm for robotic path planning for static obstacles. In: IEEE. *2019 Third International Conference on Inventive Systems and Control (ICISC)*. Coimbatore, India, 2019. p. 472–476. Cited on page 39.

HASSELT, H. V.; GUEZ, A.; SILVER, D. Deep reinforcement learning with double q-learning. In: *Proceedings of the AAAI conference on artificial intelligence*. Phoenix,USA: PKP|PS, 2016. v. 30, n. 1. Cited on page 72.

HASSELT, H. van; MAHMOOD, A. R.; SUTTON, R. S. Off-policy td (λ) with a true online equivalence. In: *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*. Quebec City, Canada: ACM, 2014. p. 330–339. Cited on page 39.

HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. San Juan, PR, USA: IEEE, 2016. p. 770–778. Cited on page 43.

HUANG, Q. Model-based or model-free, a review of approaches in reinforcement learning. In: IEEE. *2020 International Conference on Computing and Data Science (CDS)*. Stanford, CA, USA, 2020. p. 219–221. Cited on page 39.

IMANBERDIYEV, N. et al. Autonomous navigation of uav by using real-time model-based reinforcement learning. In: IEEE. *2016 14th international conference on control, automation, robotics and vision (ICARCV)*. Phuket, Thailand, 2016. p. 1–6. Cited on page 26.

INDUSTRIAL-2022. 2022. <https://www.thomasnet.com/insights/industrial-robots-can-do-more-than-just-pick-and-place/>. Accessed: 2022-11-29. Cited on page 22.

JARADAT, M. A. K.; AL-ROUSAN, M.; QUADAN, L. Reinforcement based mobile robot navigation in dynamic environment. *Robotics and Computer-Integrated Manufacturing*, Elsevier, v. 27, n. 1, p. 135–149, 2011. Cited on page 40.

JIA, L. et al. Fabric defect inspection based on lattice segmentation and template statistics. *Information Sciences*, Elsevier, v. 512, p. 964–984, 2020. Cited on page 40.

JULIAN, K. D.; KOCHENDERFER, M. J. Distributed wildfire surveillance with autonomous aircraft using deep reinforcement learning. *Journal of Guidance, Control, and Dynamics*, American Institute of Aeronautics and Astronautics, v. 42, n. 8, p. 1768–1778, 2019. Cited on page 21.

KÄHLER, O.; PRISACARIU, V. A.; MURRAY, D. W. Real-time large-scale dense 3d reconstruction with loop closure. In: SPRINGER. *European Conference on Computer Vision*. Amsterdam, The Netherlands, 2016. p. 500–516. Cited on page 21.

KAMOSHIDA, R.; KAZAMA, Y. Acquisition of automated guided vehicle route planning policy using deep reinforcement learning. In: IEEE. *2017 6th IEEE International Conference on Advanced Logistics and Transport (ICALT)*. Bali, Indonesia, 2017. p. 1–6. Cited on page 32.

KHAMIDEHI, B.; SOUSA, E. S. A double q-learning approach for navigation of aerial vehicles with connectivity constraint. In: IEEE. *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. Seoul, South Korea, 2020. p. 1–6. Cited on page 40.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, AcM New York, NY, USA, v. 60, n. 6, p. 84–90, 2017. Cited on page 43.

KRUEGER, K. A.; DAYAN, P. Flexible shaping: How learning in small steps helps. *Cognition*, Elsevier, v. 110, n. 3, p. 380–394, 2009. Cited on page 44.

KULATHUNGA, G. A reinforcement learning based path planning approach in 3d environment. *arXiv preprint arXiv:2105.10342*, 2021. Cited on page 30.

LAMINI, C.; BENHLIMA, S.; ELBEKRI, A. Genetic algorithm based approach for autonomous mobile robot path planning. *Procedia Computer Science*, Elsevier, v. 127, p. 180–189, 2018. Cited on page 30.

LANGER, A. et al. Trust in socially assistive robots: Considerations for use in rehabilitation. *Neuroscience & Biobehavioral Reviews*, Elsevier, v. 104, p. 231–239, 2019. Cited on page 22.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015. Cited on page 24.

LEE, H.; JEONG, J. Mobile robot path optimization technique based on reinforcement learning algorithm in warehouse environment. *Applied sciences*, MDPI, v. 11, n. 3, p. 1209, 2021. Cited on page 33.

LEIKE, J. et al. Ai safety gridworlds. *arXiv preprint arXiv:1711.09883*, 2017. Cited on page 86.

LI, M. P. et al. Task selection by autonomous mobile robots in a warehouse using deep reinforcement learning. In: IEEE. *2019 Winter Simulation Conference (WSC)*. National Harbor, MD, USA, 2019. p. 680–689. Cited on page 32.

LIEBISCH, F. et al. Flourish—a robotic approach for automation in crop management. In: *Workshop computer-Bildanalyse und Unbemannte Autonom Fliegende Systeme in der Landwirtschaft*. Potsdam, Germany: Leibniz-Institut für Agrartechnik und Bioökonomie, 2016. v. 21, p. 2016. Cited on page 21.

LIM, J.; HA, S.; CHOI, J. Prediction of reward functions for deep reinforcement learning via gaussian process regression. *IEEE/ASME Transactions on Mechatronics*, IEEE, v. 25, n. 4, p. 1739–1746, 2020. Cited on page 32.

LU, S.; LU, Z.; ZHANG, Y.-D. Pathological brain detection based on alexnet and transfer learning. *Journal of computational science*, Elsevier, v. 30, p. 41–47, 2019. Cited on page 43.

LUO, S.; KASAEI, H.; SCHOMAKER, L. Accelerating reinforcement learning for reaching using continuous curriculum learning. In: IEEE. *2020 International Joint Conference on Neural Networks (IJCNN)*. Glasgow, UK, 2020. p. 1–8. Cited 2 times on pages 27 and 45.

MACHINE-LEARNING. 2022. <https://www.aitude.com/supervised-vs-unsupervised-vs-reinforcement/>. Accessed: 2022-11-29. Cited on page 24.

MAMMONE, A.; TURCHI, M.; CRISTIANINI, N. Support vector machines. *Wiley Interdisciplinary Reviews: Computational Statistics*, Wiley Online Library, v. 1, n. 3, p. 283–289, 2009. Cited on page 24.

MAOUDJ, A.; HENTOUT, A. Optimal path planning approach based on q-learning algorithm for mobile robots. *Applied Soft Computing*, Elsevier, v. 97, p. 106796, 2020. Cited on page 31.

MAPPING. 2022. <https://www.researchgate.net/figure/MLS-map-and-mobile-robot-used-for-the-localization-experiments-The-area-represented-by_fig2_221312505>. Accessed: 2022-11-29. Cited on page 22.

MNIH, V. et al. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. Cited on page 26.

NAGATA, F. et al. Orientation detection using a cnn designed by transfer learning of alexnet. In: *Proceedings of the 8th IIAE International Conference on Industrial Application Engineering*. Shimane, Japan: IIAE, 2020. v. 5, p. 26–30. Cited on page 43.

NARVEKAR, S. et al. Curriculum learning for reinforcement learning domains: A framework and survey. *arXiv preprint arXiv:2003.04960*, 2020. Cited 2 times on pages 44 and 45.

NASCIMENTO, E. de S. et al. Understanding development process of machine learning systems: Challenges and solutions. In: IEEE. *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. Porto de Galinhas, Brazil, 2019. p. 1–6. Cited on page 26.

NASTESKI, V. An overview of the supervised machine learning methods. *Horizons. b*, v. 4, p. 51–62, 2017. Cited on page 24.

NGUYEN, H.; LA, H. Review of deep reinforcement learning for robot manipulation. In: IEEE. *2019 Third IEEE International Conference on Robotic Computing (IRC)*. Naples, Italy, 2019. p. 590–595. Cited on page 38.

OLIFF, H. et al. Reinforcement learning for facilitating human-robot-interaction in manufacturing. *Journal of Manufacturing Systems*, Elsevier, v. 56, p. 326–340, 2020. Cited on page 22.

OLIVEIRA, I. R. L. D.; CARVALHO, K. B. D.; BRANDÃO, A. S. Curriculum-based reinforcement learning for an effective multi-agent path planning algorithm in warehouse scenarios. In: IEEE. *2023 15th IEEE International Conference on Industry Applications (INDUSCON)*. [S.l.], 2023. p. 471–477. Cited on page 35.

OU, J. et al. Gpu-based global path planning using genetic algorithm with near corner initialization. *Journal of Intelligent & Robotic Systems*, Springer, v. 104, n. 2, p. 1–17, 2022. Cited on page 23.

PAN, S. J. Transfer learning. *Learning*, v. 21, p. 1–2, 2020. Cited on page 42.

PAN, S. J.; YANG, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, IEEE, v. 22, n. 10, p. 1345–1359, 2009. Cited on page 41.

PERERA, A.; KAMALARUBAN, P. Applications of reinforcement learning in energy systems. *Renewable and Sustainable Energy Reviews*, Elsevier, v. 137, p. 110618, 2021. Cited on page 26.

PICCINELLI, N.; MURADORE, R. Hybrid motion planner integrating global voronoi diagrams and local velocity obstacle method. In: IEEE. *2018 European Control Conference (ECC)*. Limassol, Cyprus, 2018. p. 26–31. Cited on page 23.

POLYDOROS, A. S.; NALPANTIDIS, L. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, Springer, v. 86, n. 2, p. 153–173, 2017. Cited on page 26.

QIN, H. et al. Autonomous exploration and mapping system using heterogeneous uavs and ugvs in gps-denied environments. *IEEE Transactions on Vehicular Technology*, IEEE, v. 68, n. 2, p. 1339–1350, 2019. Cited on page 23.

RAHMANI, B. et al. Review of vision-based robot navigation method. *IAES International Journal of Robotics and Automation*, IAES Institute of Advanced Engineering and Science, v. 4, n. 4, 2015. Cited on page 23.

REN, J.; HUANG, X. Potential fields guided deep reinforcement learning for optimal path planning in a warehouse. In: IEEE. *2021 IEEE 7th International Conference on Control Science and Systems Engineering (ICCSSE)*. Qingdao, China, 2021. p. 257–261. Cited on page 32.

REY, R. et al. Human-robot co-working system for warehouse automation. In: IEEE. *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Zaragoza, Spain, 2019. p. 578–585. Cited on page 22.

RODRÍGUEZ, M. et al. Wilderness search and rescue with heterogeneous multi-robot systems. In: IEEE. *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. Athens, Greece, 2020. p. 110–116. Cited on page 21.

ROMERO-MARTÍ, D. P. et al. Navigation and path planning using reinforcement learning for a roomba robot. In: IEEE. *2016 XVIII Congreso Mexicano de Robotica*. Ciudad de Mazatlán, Sinaloa, 2016. p. 1–5. Cited on page 40.

RUBIO, F.; VALERO, F.; LLOPIS-ALBERT, C. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, SAGE Publications Sage UK: London, England, v. 16, n. 2, p. 1729881419839596, 2019. Cited on page 21.

SEKARAN, S. A. R.; LEE, C. P.; LIM, K. M. Facial emotion recognition using transfer learning of alexnet. In: IEEE. *2021 9th International Conference on Information and Communication Technology (ICoICT)*. [S.l.], 2021. p. 170–174. Cited on page 43.

SERRANO-PÉREZ, O. et al. Offline robust tuning of the motion control for omnidirectional mobile robots. *Applied Soft Computing*, Elsevier, v. 110, p. 107648, 2021. Cited on page 23.

SHAO, K.; ZHU, Y.; ZHAO, D. Starcraft micromanagement with reinforcement learning and curriculum transfer learning. *IEEE Transactions on Emerging Topics in Computational Intelligence*, IEEE, v. 3, n. 1, p. 73–84, 2018. Cited on page 27.

SHIMODAIRA, H. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, Elsevier, v. 90, n. 2, p. 227–244, 2000. Cited on page 41.

SHIVGAN, R.; DONG, Z. Energy-efficient drone coverage path planning using genetic algorithm. In: IEEE. *2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR)*. Newark, NJ, USA, 2020. p. 1–6. Cited on page 49.

SHOJAEE, G. K.; MASHHADI, H. R. Optimistic initial value analysis in a greedy selection approach to mab problems. In: IEEE. *2017 7th International Conference on Computer and Knowledge Engineering (ICCKE)*. Mashhad, Iran, 2017. p. 419–424. Cited on page 38.

SINAGA, K. P.; YANG, M.-S. Unsupervised k-means clustering algorithm. *IEEE access*, IEEE, v. 8, p. 80716–80727, 2020. Cited on page 25.

SONG, Y. et al. An efficient initialization approach of q-learning for mobile robots. *International Journal of Control, Automation and Systems*, Springer, v. 10, n. 1, p. 166–172, 2012. Cited on page 40.

SUPERVISED. 2022. <https://www.enjoyalgorithms.com/blogs/supervised-unsupervised-and-semisupervised-learning>. Accessed: 2022-11-29. Cited on page 24.

SURVAILLANCE. 2022. <https://www.zenadrone.com/top-7-benefits-of-using-drone/>. Accessed: 2022-11-29. Cited on page 22.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. London, England: MIT press, 2018. Cited 2 times on pages 26 and 37.

SZCZEPANSKI, R.; BEREIT, A.; TARCZEWSKI, T. Efficient local path planning algorithm using artificial potential field supported by augmented reality. *Energies*, MDPI, v. 14, n. 20, p. 6642, 2021. Cited on page 23.

TAI, L.; LIU, M. A robot exploration strategy based on q-learning network. In: IEEE. *2016 ieee international conference on real-time computing and robotics (rcar)*. [S.l.], 2016. p. 57–62. Cited on page 31.

TAN, C. S.; MOHD-MOKHTAR, R.; ARSHAD, M. R. A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms. *IEEE Access*, IEEE, 2021. Cited on page 30.

TAYLOR, M. E.; STONE, P. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, v. 10, n. 7, 2009. Cited 2 times on pages 43 and 44.

TSANG, K. F. E. et al. A novel warehouse multi-robot automation system with semi-complete and computationally efficient path planning and adaptive genetic task allocation algorithms. In: IEEE. *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. Singapore, 2018. p. 1671–1676. Cited on page 32.

TSENG, F. H. et al. A star search algorithm for civil uav path planning with 3g communication. In: IEEE. *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. Kitakyushu, Japan, 2014. p. 942–945. Cited on page 23.

UNSUPERVISED. 2022. <https://towardsdatascience.com/a-brief-introduction-to-unsupervised-learning-20db46445283>. Accessed: 2022-11-29. Cited on page 25.

VASCONCELOS, J. V. R.; BRANDÃO, A. S.; SARCINELLI-FILHO, M. Real-time path planning for strategic missions. *Applied Sciences*, MDPI, v. 10, n. 21, p. 7773, 2020. Cited 2 times on pages 30 and 31.

WALCZAK, S. Artificial neural networks. In: *Advanced methodologies and technologies in artificial intelligence, computer simulation, and human-computer interaction*. [S.l.]: IGI global, 2019. p. 40–53. Cited on page 24.

WANG, C. et al. Deep-reinforcement-learning-based autonomous uav navigation with sparse rewards. *IEEE Internet of Things Journal*, IEEE, v. 7, n. 7, p. 6180–6190, 2020. Cited on page 26.

WEINSHALL, D.; COHEN, G.; AMIR, D. Curriculum learning by transfer learning: Theory and experiments with deep networks. In: PMLR. *International Conference on Machine Learning*. Stockholm, Swedenw, 2018. p. 5238–5246. Cited on page 44.

WEISS, K.; KHOSHGOFTAAR, T. M.; WANG, D. A survey of transfer learning. *Journal of Big data*, SpringerOpen, v. 3, n. 1, p. 1–40, 2016. Cited 2 times on pages 41 and 42.

WHANG, S. E.; LEE, J.-G. Data collection and quality challenges for deep learning. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 13, n. 12, p. 3429–3432, 2020. Cited on page 26.

WUNDER, M.; LITTMAN, M. L.; BABES, M. Classes of multiagent q-learning dynamics with epsilon-greedy exploration. In: CITESEER. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. Madison, WI, United States, 2010. p. 1167–1174. Cited on page 37.

XIE, J.; CARRILLO, L. R. G.; JIN, L. Path planning for uav to cover multiple separated convex polygonal regions. *IEEE Access*, IEEE, v. 8, p. 51770–51785, 2020. Cited on page 23.

XU, J. et al. Reinforcement learning to rank with pairwise policy gradient. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, United States: Association for Computing Machinery, 2020. p. 509–518. Cited on page 40.

XUE, X. et al. A deep reinforcement learning method for mobile robot collision avoidance based on double dqn. In: IEEE. *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*. Vancouver, BC, Canada, 2019. p. 2131–2136. Cited on page 31.

YAN, C.; XIANG, X. A path planning algorithm for uav based on improved q-learning. In: IEEE. *2018 2nd International Conference on Robotics and Automation Sciences (ICRAS)*. Brisbane, Australia, 2018. p. 1–5. Cited on page 31.

YAN, C.; XIANG, X.; WANG, C. Towards real-time path planning through deep reinforcement learning for a uav in dynamic environments. *Journal of Intelligent & Robotic Systems*, Springer, v. 98, n. 2, p. 297–309, 2020. Cited on page 25.

YAO, Q. et al. Path planning method with improved artificial potential field—a reinforcement learning perspective. *IEEE Access*, IEEE, v. 8, p. 135513–135523, 2020. Cited on page 32.

ZHAO, Y.; ZHENG, Z.; LIU, Y. Survey on computational-intelligence-based uav path planning. *Knowledge-Based Systems*, Elsevier, v. 158, p. 54–64, 2018. Cited on page 23.

ZHU, Z.; LIN, K.; ZHOU, J. Transfer learning in deep reinforcement learning: A survey. *arXiv preprint arXiv:2009.07888*, 2020. Cited on page 43.

ZHUANG, F. et al. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, IEEE, v. 109, n. 1, p. 43–76, 2020. Cited 3 times on pages 26, 27, and 41.

Appendix

APPENDIX A – Hyperparemeter Analysis

Hyperparameter tuning plays a critical role in the success of reinforcement learning algorithms, as the choice and adjustment of these parameters can significantly influence the agent's learning efficiency and overall performance. Understanding the specific effects of each hyperparameter is essential, as it allows for a more informed and strategic approach to fine-tuning the learning process. In this appendix, the impact of varying key hyperparameters is explored across three dimensions: Epsilon and Alpha, Epsilon and Exploring Starts frequency, and different Q-Table Initializations. Deeper insights on how each of the evaluated parameters can influence different aspects of training are aimed with this analysis, guiding future development in similar reinforcement learning model applications.

A.1 Comparison of Epsilon and Alpha Hyperparameters

The initial analysis involves an examination of the impact of varying initial and terminal values for the hyperparameters ϵ and α. These parameters regulate the agent's likelihood to take random actions over the best-known action and the learning rate at which Q-Values are updated every episode, respectively.

In order to clearly outline how the analysis was done, the simulations must be defined as well as the terminology that will be further used to qualitatively discuss the results. Each simulation is composed by several episodes, which are composed by steps (or iterations). In each episode, the agent will try to reach from its starting cell to the predetermined goal throughout its actions. Each action consist in one step (or iteration) and the episode lasts until the agent reaches its destination, crashes or exceeds the predetermined maximum number of steps. After an episode ends, a new one begins until the predetermined maximum number of episodes. Each simulation was conducted iteratively for 10 repetitions in order to have more data for each hyperparameter set.

For this analysis, the map represented in Figure 53 was used as the learning environment, where the green cell is the origin point and the blue cell is the destination point. It is worth highlighting that during training, the agent may start in cells different from this origin point, due to the exploring starts strategy. In this Figure, the blue line represents the path taken by the agent and its length corresponds to the Path Length metric. Each time the agent takes a turn in its path, it adds up to the Total Turns metric, represented by orange dashed circles. Finally, to address safety concerns, the Euclidean

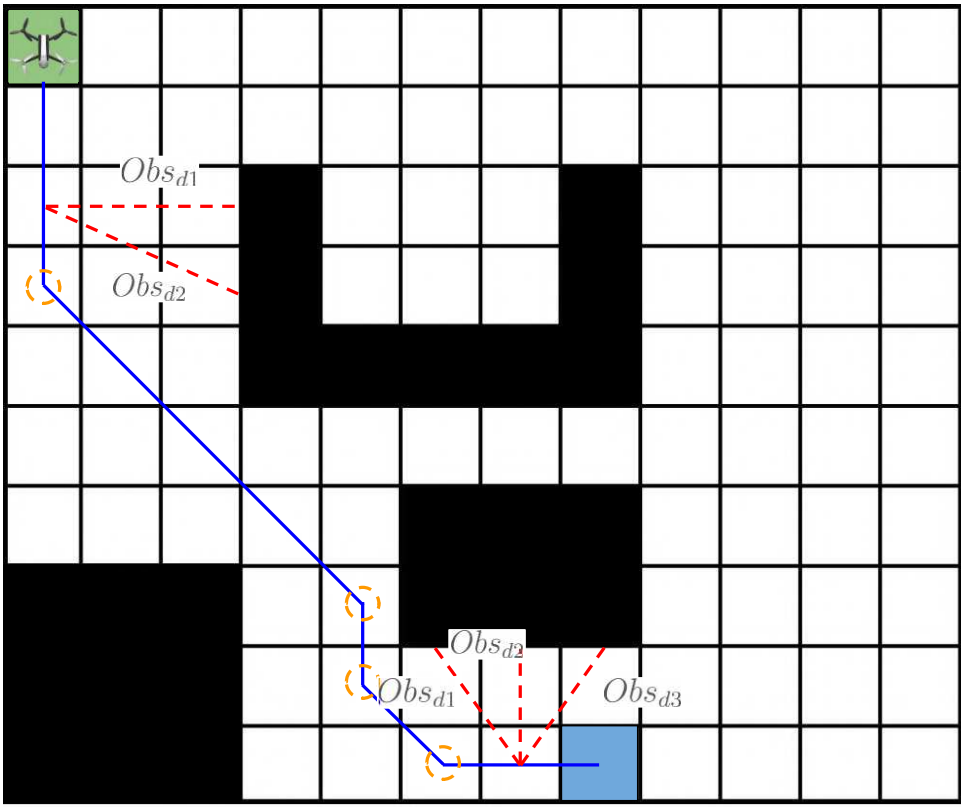


Figure 53 – Map used for each simulation in this section. Green and blue cells represent the starting and ending point respectively. Red dashed lines represent the euclidean distance to occupied cells, orange dashed circles represent each turn.

distance to the N-closest obstacles for each cell of the agent's path is calculated and this is exemplified by the red dashed lines in the image with Obs_{d_i} being the distance to the i-th obstacle. For further reference, whenever the text refers to the "Main Path", it means the path that connects pre determined origin to the destination point and whenever the text refers to "All Paths", it means the group of paths originating from every cell of the map reaching the final destination point, also including the Main Path.

Given the nature of this work, that overlaps fields of interest in machine learning as well as in robotics, there are some terms that may have different common meanings in these contexts. For the sake of clarifications, words such as oscillatory, noise and erratic, will be used to refer to the shape of the graph, denoting a non-smooth behavior of the analyzed image.

A.1.1 Performance Metrics and Representations

In conducting a performance analysis, it is imperative to not only determine the hyperparameters to be manipulated and the manner of their variation, but also the performance metrics for analysis and subsequent data visualization. Each of these components is described below.

- **Values of ϵ and α** - 72 combinations involving constant values and diverse linear decay ranges for both ϵ and α were employed for evaluation.
  - **Constant Values** - ϵ: 0.9, 0.7, 0.5, 0.3 - α: 0.3, 0.2, 0.1, 0.05;
  - **Decaying Range Upper and Lower Limits** - ϵ: (1, 0.1), (1,0.3), (0.8,0.1), (0.8,0.3), (0.6,0.3) - α: (0.3,0.05), (0.2,0.05), (0.3,0.1), (0.2,0.1)
- **Data Visualization** - Three distinct plotting techniques were utilized to emphasize various aspects of the results, as described below.
  - **Parallel Coordinates Plots** - These plots enable the grouping of all simulation metrics into fewer visualizations, allowing for an overall view of simulation performances and the identification of trends and final values in hyperparameter sets. The metrics employed in these plots were the final value averaged over the 10 repeated simulations for each set. When depicting the number of episodes taken to reach the pre determined threshold, this word was abbreviated to "T".
  - **Regular 3D Plots** - These plots were utilized to observe the progression of each of the Final Performance Values for each of the three priorities throughout training. This analysis will discuss aspects such as over and undershoots during training, as well as the plot form and convergence time for each metric. The graphics were categorized into four groups to aggregate similar hyperparameter sets: constant ϵ and α values, constant ϵ and variable α, variable ϵ and constant α, variable ϵ and α.
  - **Box Plots** - These were employed to evaluate the numerical distribution across the 10 repeated simulations for each set of hyperparameters. This analysis will discuss aspects such as final overall value as well as dispersion over the simulations. Given the volume of data, these results were presented across various subfigures for ease of viewing. In these plots, distribution of the final values as well as the thresholds are exhibited.
- **Performance Metrics** - Metrics related to two distinct aspects of our study were chosen to assess simulation performance: Main Path (MP), given the significance of establishing an optimal path for the agent, and All Paths Means (AP), considering that this algorithm emphasizes not only generating a viable path utilizing designated

priorities from the starting point to the goal, but also producing quality paths from any location on the map. The following metrics are employed for both aspects and can be categorized into two groups, resulting in a total of 12 components, as outlined below.

- **Final Performance Values** - These assess the values attained at the conclusion of training for each of the three priorities: Path Length, Mean Obstacle Distance, and Total Turns, which will henceforth be referred by the acronyms PL, MOD and TT. These attributes evaluate the overall final performance for each set of hyperparameters.
- **Performance Threshold** - These evaluate how quickly the agent was able to achieve and maintain performance within a certain threshold of the training's best performance for each of the three priorities. This evaluates not the absolute value, but the speed at which the specific hyperparameter set approached a performance level close to that obtained after all episodes. The threshold range employed was $\pm 2\%$ of the final value.

Finally, all tests were done using the same Reward Constants of -0.6, -0.1, -0.6 and 100 for the Safety, Step, Energy and Goal Rewards, respectively.

A.1.2 Parallel Coordinates Plots

Parallel Coordinate Plots were employed as a visualization tool to comprehensively present the metrics and thresholds as well as more easily visualize trends throughout the hyperparameters configurations. In this type of plot, each line represents one simulation. In Figure 54 The four columns inside the green box represent the hyperparameters used in each simulation and the six columns inside the red box represent the evaluated metrics for each of them. As an example to indicate how to read this plot, the highlighted green line denotes initial and final ϵ values (In. Eps and Final Eps) of 0.9, along with initial and final α values (In. Alp. and Final Alp.) of 0.2. Traversing the line across all columns allows access to the final metrics values averaged across the 10 repetitions of each simulation. A second example is possible to seen in the highlighted red line. It has an initial ϵ value of 0.8 and final value of 0.3, indicating that the used interval is [0.3, 0.8], and initial and final values of α of 0.3, indicating that used a constant α value of 0.3. It is possible to notice that it blends within the trend that other red lines follow with low metric values for both AP and MP. The color bar within Figures 54 and 55 corresponds to the index of each simulation, as referenced in Table 14. Their ordering follows the permutation of constant values for ϵ and α, constant ϵ and variable α, constant α and variable ϵ and, finally, variable ϵ and α values, which were grouped in four groups, A, B, C and D, respectively, which will be plotted, in this order, with the following colors: light blue, light green, light yellow and

light coral. For further reference, when referring to the simulation indexes, the notation Idx_n will refer to the the index n, Idx_{n-m} will refer to indexes from n to m and $Idx_{x,y,z}$ will refer to indexes x, y and z.

Final Performance Metrics In Figure 54, the metrics obtained from the simulations are depicted. A prominent trend reveals that red lines consistently achieve lower metric values. These red lines correspond to simulations utilizing variable ϵ and α values. Notably, yellow lines generally yield better results than green lines, although not as good as the red ones. This suggests that using constant ϵ paired with variable or constant α tends to outperform simulations with constant values for both hyperparameters. Additionally, a greater disparity of results is observed in metrics related to paths from all cells, as indicated by the broader spectrum of lines in the AP columns compared to the MP columns. This difference is attributed to the inherent variability in optimizing multiple paths compared to a single path.

Performance Thresholds Figure 55 illustrates the threshold (T) results of the simulations. Compared to the previous figure, these results exhibit a more fluctuating behavior. Nevertheless, a recurring pattern persists: red lines yield the best results, followed by yellow lines, and lastly green lines. This echoes the pattern observed in the earlier discussions, reinforcing the superior performance of variable values for ϵ and α over constant or mixed combinations of values. It is essential to note the prevalence of simulations with high thresholds, particularly in AP Turns and MP Turns, especially with constant ϵ values (green lines). This high variability is attributed to the hyperparameters used, which result in frequent and significant update steps, preventing the training metrics from consistently staying within the 2% threshold limit.

A.1.3 3D Plots for All Paths

Regular 3D plots will make way for an in-depth analysis of the evolution of each metric throughout training. Figures 56 through 58 illustrate the evolution of these metrics concerning the values of ϵ and α throughout the training regimen for All Paths. It is noteworthy that for certain hyperparameter configurations, the values of ϵ and α decay over time, with this variation serving as an indicator of episode progression. In subfigures (b), (c), and (d), it becomes evident that the metric values initiate at low levels and progressively increase with the gradual decay of either α, ϵ, or both. This trend can be attributed to the time required to identify feasible paths for all cells. In the initial stages of training, the paths that converge first are those proximate to the goal, leading to lower initial values for the average metrics associated with paths originating from each cell. The presented values in these figures result from the averaging of metrics derived from 10

Table 14 – ϵ and α Permutation Indexes. Group A has constant values for ϵ and α, B constant values for ϵ and variable for α, C variable values for ϵ and constant for α and D has variable values for both.

A	Index	**0**	**1**	**2**	**3**	**4**
	ϵ/α	0.9, 0.3	0.9, 0.2	0.9, 0.1	0.9, 0.05	0.7, 0.3
	Index	**5**	**6**	**7**	**8**	**9**
	ϵ/α	0.7, 0.2	0.7, 0.1	0.7, 0.05	0.5, 0.3	0.5, 0.2
	Index	**10**	**11**	**12**	**13**	**14**
	ϵ/α	0.5, 0.1	0.3, 0.05	0.3, 0.3	0.3, 0.2	0.3, 0.1
	Index	**15**				
	ϵ/α	0.3, 0.05				
B	Index	**16**	**17**	**18**	**19**	**20**
	ϵ/α	0.9, [0.3, 0.05]	0.9, [0.2, 0.05]	0.9, [0.3, 0.1]	0.9, [0.2, 0.1]	0.7, [0.3, 0.05]
	Index	**21**	**22**	**23**	**24**	**25**
	ϵ/α	0.7, [0.2, 0.05]	0.7, [0.3, 0.1]	0.7, [0.2,0.1]	0.5, [0.3, 0.05]	0.5, [0.2, 0.05]
	Index	**26**	**27**	**28**	**29**	**30**
	ϵ/α	0.5, [0.3, 0.1]	0.3, [0.2, 0.1]	0.3, [0.3, 0.05]	0.3, [0.2, 0.05]	0.3, [0.3, 0.1]
	Index	**31**				
	ϵ/α	0.3, [0.2, 0.1]				
C	Index	**32**	**33**	**34**	**35**	**36**
	ϵ/α	[1, 0.1], 0.3	[1, 0.1], 0.2	[1, 0.1], 0.1	[1, 0.1], 0.05	[1, 0.3], 0.3
	Index	**37**	**38**	**39**	**40**	**41**
	ϵ/α	[1, 0.3], 0.2	[1, 0.3], 0.1	[1, 0.3], 0.05	[0.8, 0.1], 0.3	[0.8, 0.1], 0.2
	Index	**42**	**43**	**44**	**45**	**46**
	ϵ/α	[0.8, 0.1], 0.1	[0.8, 0.1], 0.05	[0.8, 0.3], 0.3	[0.8, 0.3], 0.2	[0.8, 0.3], 0.1
	Index	**47**	**48**	**49**	**50**	**51**
	ϵ/α	[0.8, 0.3], 0.05	[0.6, 0.3], 0.3	[0.6, 0.3], 0.2	[0.6, 0.3], 0.1	[0.6, 0.3], 0.05
D	Index	**52**	**53**	**54**	**55**	**56**
	ϵ/α	[1, 0.1], [0.3, 0.05]	[1, 0.1], [0.2, 0.05]	[1, 0.1], [0.3, 0.1]	[1, 0.1], [0.2, 0.1]	[1, 0.3], [0.3, 0.05]
	Index	**57**	**58**	**59**	**60**	**61**
	ϵ/α	[1, 0.3], [0.2, 0.05]	[1, 0.3], [0.3, 0.1]	[1, 0.3], [0.2, 0.1]	[0.8, 0.1], [0.3, 0.05]	[0.8, 0.1], [0.2, 0.05]
	Index	**62**	**63**	**64**	**65**	**66**
	ϵ/α	[0.8, 0.1], [0.3, 0.1]	[0.8, 0.1], [0.2, 0.1]	[0.8, 0.3], [0.3, 0.05]	[0.8, 0.3], [0.2, 0.05]	[0.8, 0.3], [0.3, 0.1]
	Index	**67**	**68**	**69**	**70**	**71**
	ϵ/α	[0.8, 0.3], [0.2, 0.1]	[0.6, 0.3], [0.3, 0.05]	[0.6, 0.3], [0.2, 0.05]	[0.6, 0.3], [0.3, 0.1]	[0.6, 0.3], [0.2, 0.1]

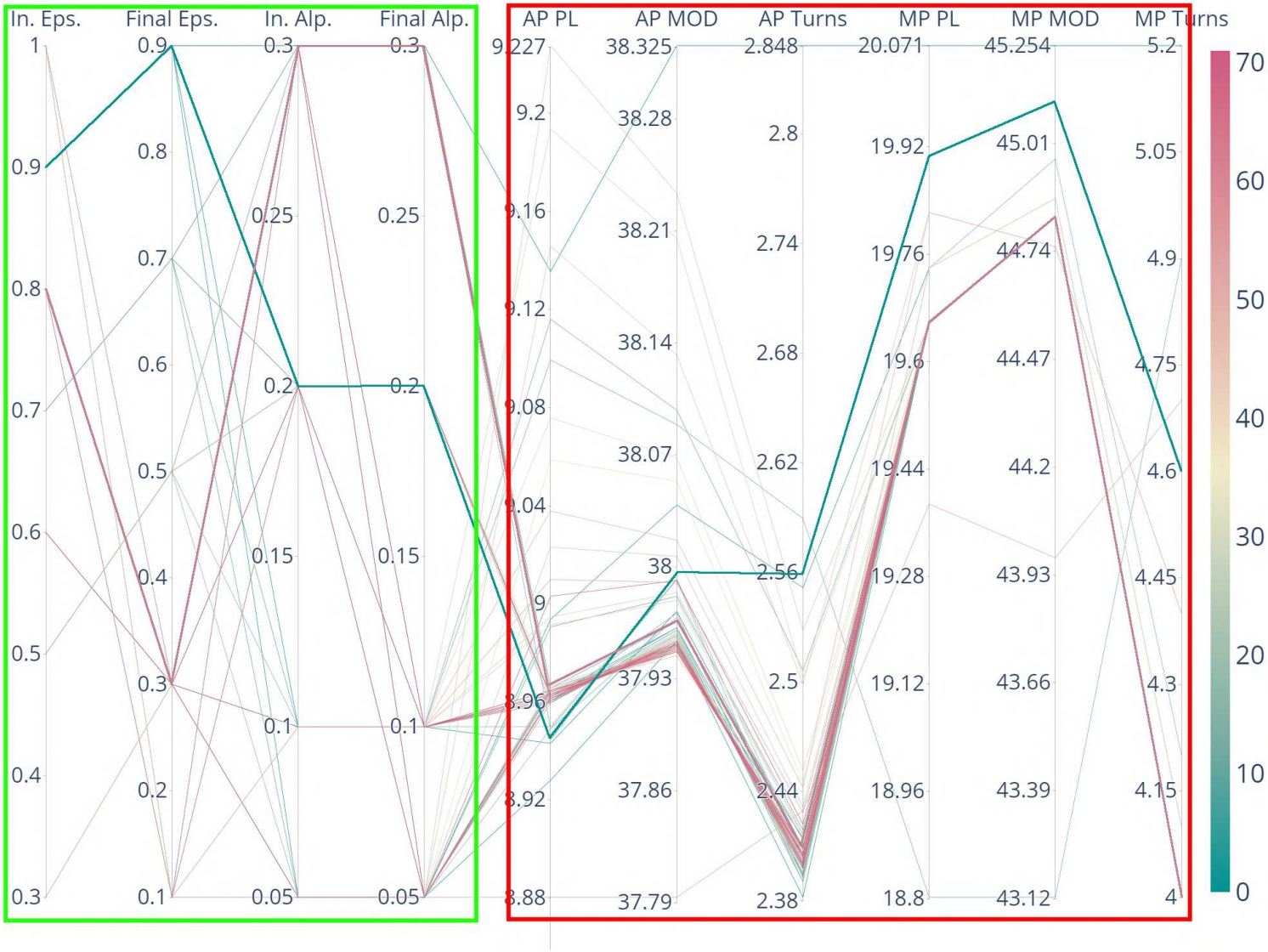


Figure 54 – Parallel Coordinates Plot for Metrics in all Simulations. Axis inside green box are the hyperparameters used, and axis inside red box are the evaluated metrics. Green highlighted line constant value of 0.9 for ϵ and constant value of 0.2 for α. Red highlighted line used a variable interval of [0.3, 0.8] for ϵ and a constant value of 0.3 for α

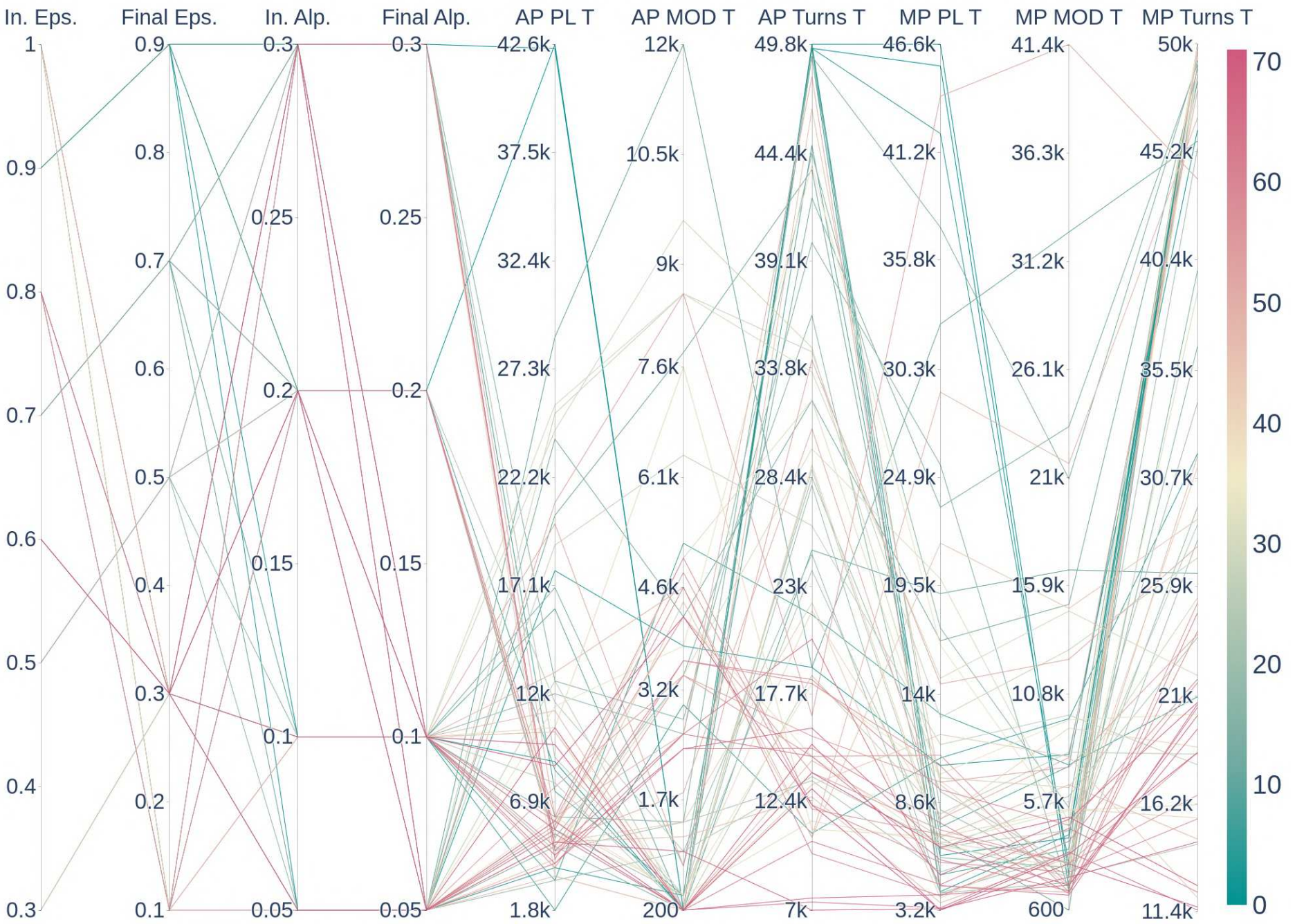


Figure 55 – Parallel Coordinates Plot for Threshold in all Simulations.

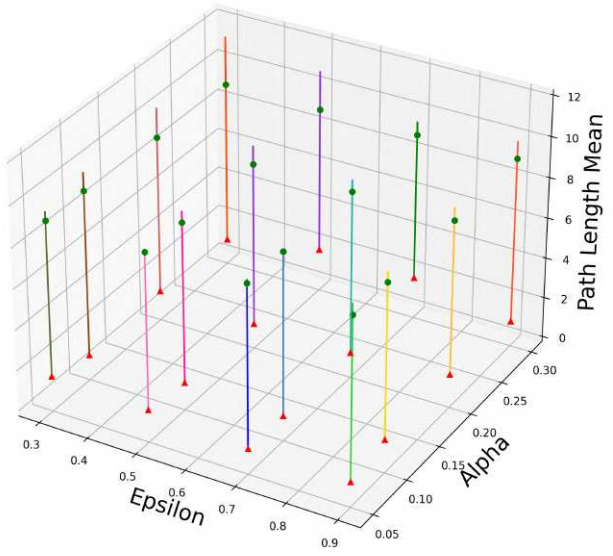
repetitions conducted for each simulation. To mitigate the inherent noise in the values, we applied a moving average window with a span of 100 episodes.

A.1.3.1 All Paths Path Length Metric Analysis

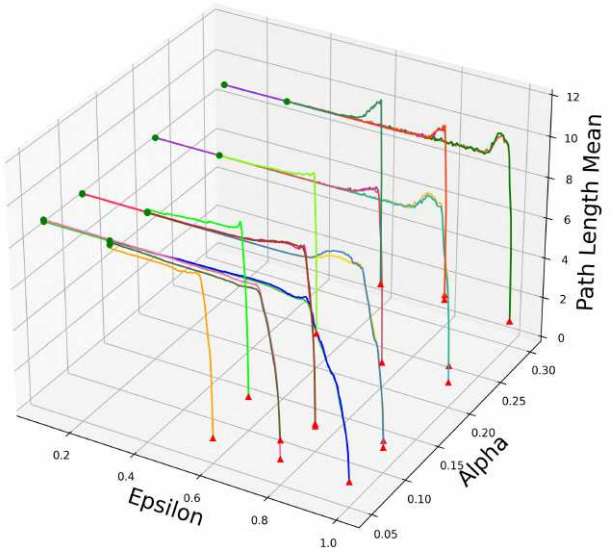
Figure 56 depicts the mean Path Length metric computed for paths associated with every cell in the map, where sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively.

AP PL - Constant ϵ and α values Figure 56 (a) shows the variation throughout training for constant ϵ and α values for the PL metric regarding AP. Given that the passing of time was represented by the decay of either of the hyper parameters, this graph will only provide the final value (red triangle) and final one (green circle). It is possible to see the lowest overshoot values when using $\epsilon = 0.9$ and $\alpha = 0.05$ while the highest one occurs for $\epsilon = 0.3$ and $\alpha = 0.3$. Analyzing the trend throughout the other configurations, it is possible to notice that lower ϵ values and higher α lead to more overshoot. With lower α, the updates are less aggressive and with higher ϵ, there is more search for different options, leading to bad actions not impacting as heavily and good actions being found quickly, with the counterpart leading to lower exploration and higher Q-value updates, taking longer to correct mistakes as well as taking more time to find better actions.

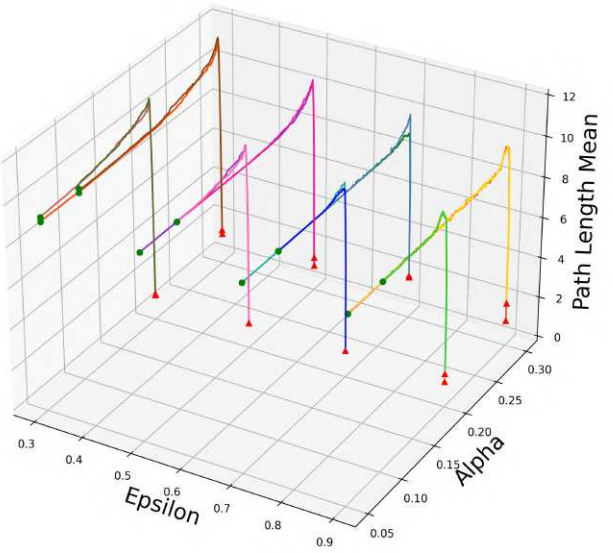
AP PL - Variable ϵ and constant α values Figure 56 (b) shows the variation throughout training for variable ϵ and constant α values for the PL metric regarding AP. It is possible to observe an initial overshoot at the beginning of training. This overshoot tends to be more pronounced for higher values of constant α and initial ϵ such as seen with $\epsilon \in [0.3, 0.9]$ and $\alpha = 0.3$. This can be elucidated by the heightened exploration rate, which may result in suboptimal choices in the early episodes, leading to larger updates due to the higher learning rate, thereby yielding a more substantial peak. Moreover, in scenarios with elevated initial ϵ values, these peaks tend to attenuate more rapidly, especially when accompanied by higher α values. This phenomenon may arise because a higher initial ϵ could result in a more substantial overshoot, but with increased exploration, the optimal actions are identified more swiftly, resulting in a more rapid attenuation of the overshoot. This effect is accentuated in situations involving a higher learning rate due to larger step sizes during each iteration. An intriguing observation pertains to the plots featuring a constant α value of 0.05, which exhibit significantly lower overshoot. This is complemented by a less steep ascent in the graph, signifying that the convergence for all paths takes longer, particularly in cases with higher initial ϵ values. This suggests that a lower learning rate may require a more extended period to converge with respect to path length, concerning



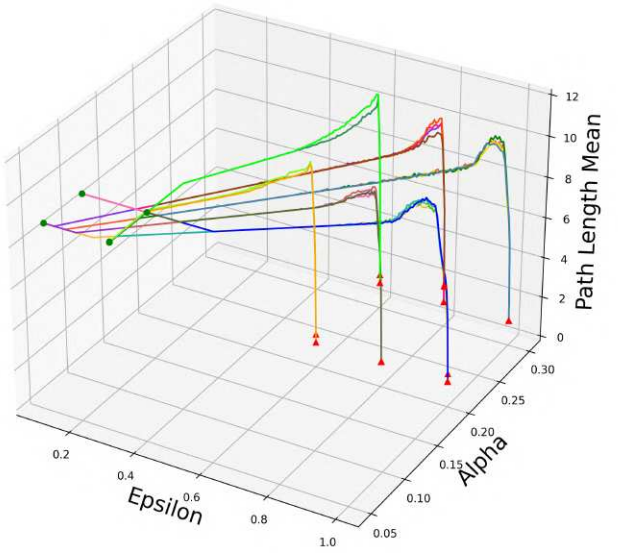
(a) Constant ϵ and α values.



(b) Variable ϵ and constant α values.



(c) Constant ϵ and variable α values.



(d) Variable ϵ and α values.

Figure 56 – All Paths Path Length average throughout training. Sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively. Starting value is represented by a red triangle and final value by a green circle.

overall path performance, while experiencing reduced overshooting. Furthermore, in such instances, an inclination towards exploration, particularly through the adoption of a higher initial ϵ value, leads to slower convergence for all cells concerning the path length metric.

AP PL - Constant ϵ and variable α values - In Figure 56 (c) we observe that overshoots exhibit a sharper and slower attenuation compared to cases with variable ϵ and

constant α. Moreover, lower ϵ and higher initial α values result in more pronounced peaks such as seen in $\epsilon = 0.3$ and $\alpha \in [0.05, 0.3]$. This phenomenon suggests that higher constant ϵ values serve to dampen peak values during the initial phases of training. This behavior stems from the fact that the agent commences training without prior knowledge of navigating the environment. Lower ϵ values lead to a lower exploration-to-exploitation ratio, potentially causing the agent to persist in suboptimal actions, which are mistakenly regarded as the best choices. This behavior propagates throughout the environment, resulting in the higher peaks and longer attenuation, as it takes more episodes to explore extensively, as opposed to scenarios with higher ϵ values. Furthermore, higher initial α values lead to prolonged influence of each action on the Q-Value due to steeper value updates, contributing to higher peaks and shorter attenuation periods.

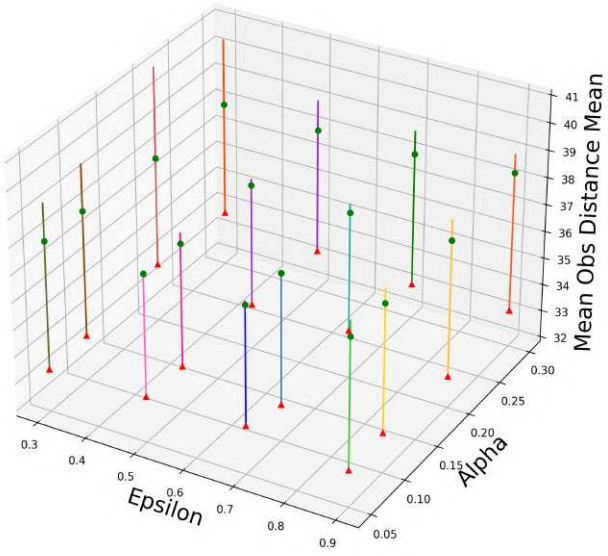
AP PL - Variable ϵ and α values Figure 56 (d) illustrates similar trends when variable values are employed for both ϵ and α. Specifically, higher α values lead to higher peaks and quicker attenuation, while lower ϵ values result in higher peaks with extended attenuation times.

Overall, it is possible to see that higher α values will lea the agent to be more heavily penalized for bad actions possibly leading to higher overshoots. While higher ϵ values lead to finding better actions quicker so overshoots are softened. In the next section it will be discussed the AP metrics for MOD.

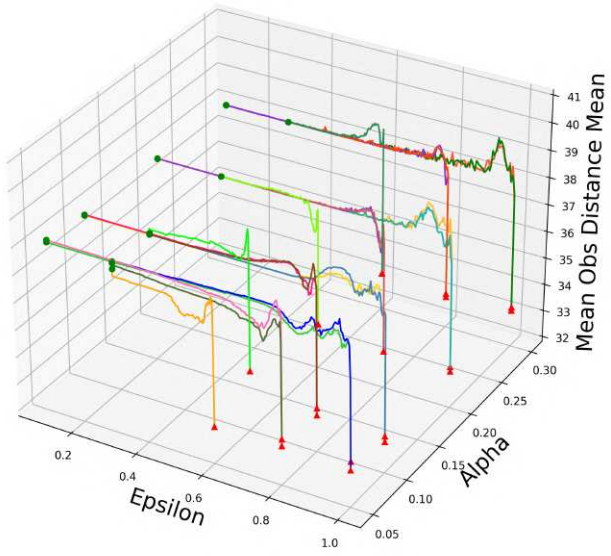
A.1.3.2 All Paths Mean Obstacle Distance Metric Analysis

Figure 57 depicts the Mean Obstacle Distance metric computed for paths associated with every cell in the map, where sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively.

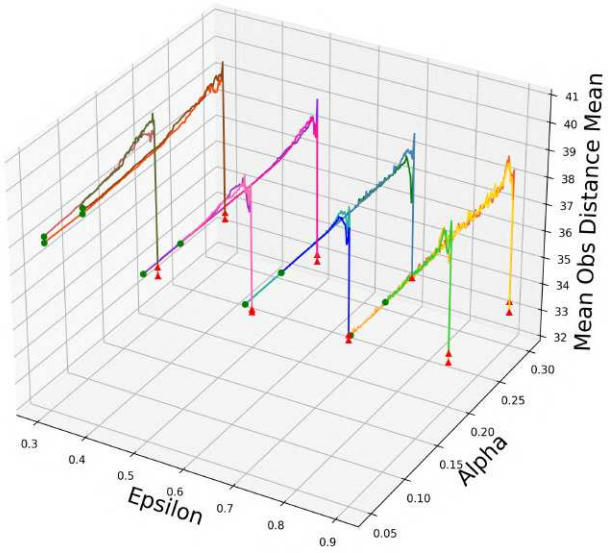
AP MOD - Constant ϵ and α values Figure 57 (a) shows the variation throughout training for constant ϵ and α values for the MOD metric regarding AP. Given that the passing of time was represented by the decay of either of the hyper parameters, this graph will only provide the final value (red triangle) and final one (green circle). It is possible to see a similar behavior when compared to the PL metric with lowest overshoot values when using $\epsilon = 0.9$ and $\alpha = 0.05$ while the highest one occurs for $\epsilon = 0.3$ and $\alpha = 0.2$. Analyzing the trend throughout the other configurations, it is possible to notice that lower ϵ values and higher α lead to more overshoot. This shows that this metric responds in a similar way compared to the PL metric in regards of using constant values for both hyperparameters.



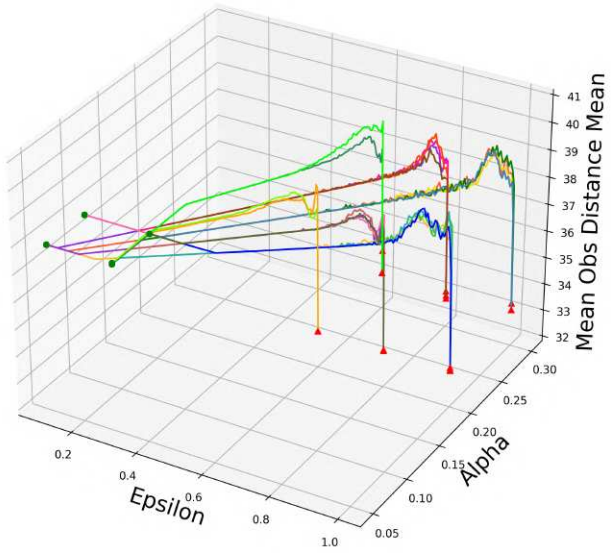
(a) Constant ϵ and α values.



(b) Variable ϵ and constant α values.



(c) Constant ϵ and variable α values.



(d) Variable ϵ and α values.

Figure 57 – All Paths Mean Obstacle Distance average throughout training. Sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively. Starting value is represented by a red triangle and final value by a green circle.

AP MOD - Variable ϵ and constant α values Figure 57 (b) shows the variation throughout training for variable ϵ and constant α values for the MOD metric regarding AP. We can observe a more oscillatory pattern for lower constant α values and larger maximum values for higher αs. Similarly to the previous metric, higher α values lead to higher peaks, but when combined with higher ϵ values, the oscillatory behavior diminishes more rapidly in comparison to scenarios with lower ϵ values. An interesting pattern arises in certain instances featuring lower α values, characterized

by undershoots. This phenomenon suggests that, given the prescribed priorities, the training process initially identifies feasible paths and subsequently refines them for safety in the later stages of training, as observed in the yellow plot for an α value of 0.05 and an initial ϵ value of 0.6.

AP MOD - Constant ϵ and variable α values - In Figure 57 (c), it is possible to see that lower ϵ values result in higher peaks and longer attenuation times, requiring more episodes to explore and identify optimal actions. Larger peak values are observed when starting with higher α values, due to the utilization of larger update steps. The undershoot pattern only emerges with ϵ values exceeding 0.5, likely indicating the identification of feasible paths leading to the goal, albeit with limited safety considerations. This behavior is not observed when beginning with an α value of 0.3 and an ϵ value of 0.9, possibly due to larger update steps and heightened exploration rates, which mitigate the intensification of specific behaviors.

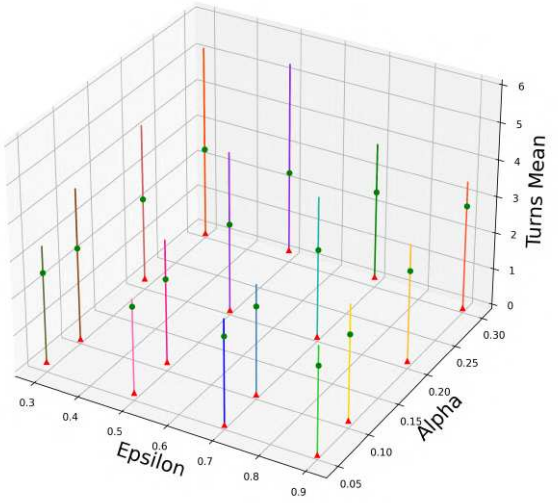
AP MOD - Variable ϵ and α values In Figure 57 (d), where both hyperparameters exhibit variable values, it is demonstrated that fewer instances of the undershoot pattern observed in the previous two sub-figures. Such behavior occurs primarily with lower initial values of α and ϵ. The maximum overshoot and attenuation time in these settings are lower than when one of the hyperparameters is held constant.

In the context of the Mean Distance to Obstacles metric for paths originating from all cells, the training process exhibits a more erratic profile when compared to the previous analysed metric. This pattern can be attributed to the greater degree of fine-tuning associated with this metric compared to the Path Length metric. Overall this sections results suggests that this metric is more prone to different behaviors such as overshoot and undershoots when it comes to fine-tuning these hyperparameters. This indicates that adaptively tuning these hyperparameters can significantly enhance the algorithm's performance, providing a nuanced approach to achieving safer routes. In the next section it will be discussed the AP metrics for the TT metric.

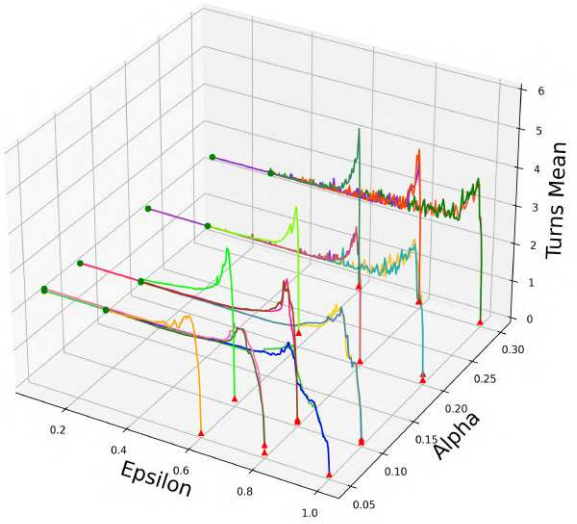
A.1.3.3 All Paths Total Turns Metric Analysis

Figure 58 depicts the Total Turns metric computed for paths associated with every cell in the map, where sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively.

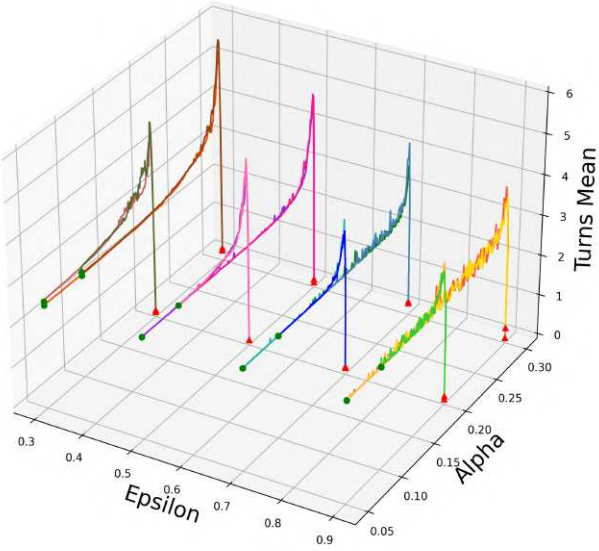
AP TT - Constant ϵ and α values Figure 57 (a) shows the variation throughout training for constant ϵ and α values for the TT metric regarding AP. Given that the passing of time was represented by the decay of either of the hyper parameters, this graph will



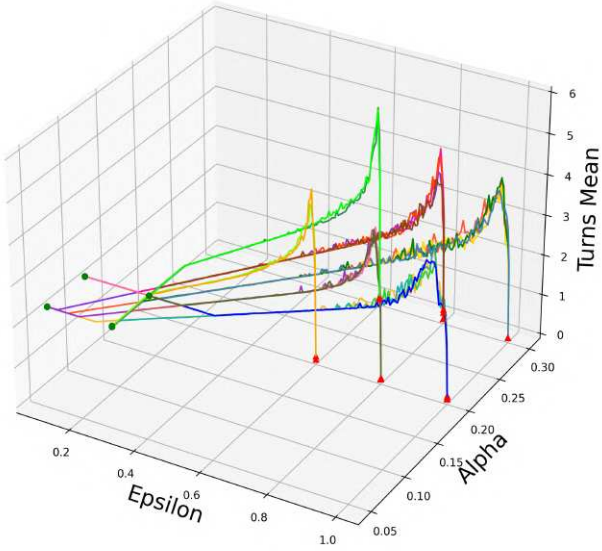
(a) Constant ϵ and α values.



(b) Variable ϵ and constant α values.



(c) Constant ϵ and variable α values.



(d) Variable ϵ and α values.

Figure 58 – All Paths Total Turns average throughout training. Sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively. Starting value is represented by a red triangle and final value by a green circle.

only provide the final value (red triangle) and final one (green circle). It is possible to see a similar behavior when compared to the other two metrics with lowest overshoot values when using $\epsilon = 0.9$ and $\alpha = 0.03$ while the highest one occurs for $\epsilon = 0.3$ and $\alpha = 0.2$. Analyzing the trend throughout the other configurations, it is possible to notice that lower ϵ values and higher α lead to more overshoot. This shows that this metric responds in a similar way compared to the other two metrics in regards of using constant values for both hyperparameters.

AP TT - Variable ϵ and constant α values Figure 57 (b) shows the variation throughout

training for variable ϵ and constant α values for the TT metric regarding AP. It is observed that the TT metric for constant learning rates and variable ϵ values. This metric exhibits heightened oscillatory behavior in training when an initially large learning rate is employed, particularly when combined with higher initial ϵ values. Lower values of α, along with high initial ϵ values, extend the time required for training to reach its peak value, implying that more time is needed to discover feasible paths for distant cells. Additionally, a clear reduction in the noisy pattern is observed when using lower values of α due to less drastic updates in the Q-Value at each iteration.

AP TT - Constant ϵ and variable α values - In Figure 57 (c), we observe higher overshoots and longer attenuation times as ϵ values decrease. This can be attributed to low exploration during the initial stages, leading to the repeated selection of suboptimal actions due to the agent's limited knowledge of the environment. This intensifies undesirable behaviors over an extended period before sufficient episodes permit effective exploration to occur. Furthermore, for higher ϵ values, there is an increased noise during training, which is mitigated when starting with lower values of α.

AP TT - Variable ϵ and α values In Figure 57 (d), where both hyperparameters exhibit variable values, a less intense oscillatory behavior is observed in comparison to the preceding two sub-figures. Larger initial α values yield higher yet more narrow overshoots. Moreover, simulations starting with higher initial ϵ values exhibit noticeable oscillations. Lower initial α values, such as 0.20, require a slightly longer training duration to reach their peak, suggesting a slower convergence for distant cells.

When it comes to the TT metric, the graphs depicted in Figure 58 reveal a conspicuously more erratic behavior in comparison to the two preceding metrics. This erratic pattern can be elucidated by examining the scale of variation in this metric relative to its maximum value. Given that adjustments in the number of turns within a path always result in integer values, any modifications in this metric represent a significant proportion of the maximum recorded value. For instance, if a path is refined from 3 turns to 2, while the maximum number of turns observed in any path was 4, this alteration signifies a 25% variation from the maximum value, thereby contributing to the pronounced noise observed. In contrast, when compared to the Mean Obstacle Distance metric in Figure 56, where the maximum value encountered was approximately 10 meters, variations in this metric may be less than 1 meter, constituting a substantially smaller fraction of the maximum value and resulting in a less noisy profile.

In conclusion, the metrics associated with paths originating from various cells exhibit similar trends for different combinations of constant and variable ϵ and α values.

This behavior tends to manifest in terms of extended attenuation times and higher overshoot values for lower constant ϵ values, a more stable and gradual increase toward convergence for low constant α values, and a more pronounced oscillatory pattern for higher α values. When variable values are used for both hyperparameters, indications of analogous albeit less pronounced patterns emerge, as observed with only one of the hyperparameters exhibiting variability. Each metric possesses its distinct characteristics, such as the presence of undershoot in some Mean Obstacle Distance simulations and the more pronounced oscillatory pattern observed in the Number of Turns metric. Ultimately, distinct ranges of variation in each hyperparameter yield similar graph structures in this analysis. In the next section it will be discussed the MP metrics.

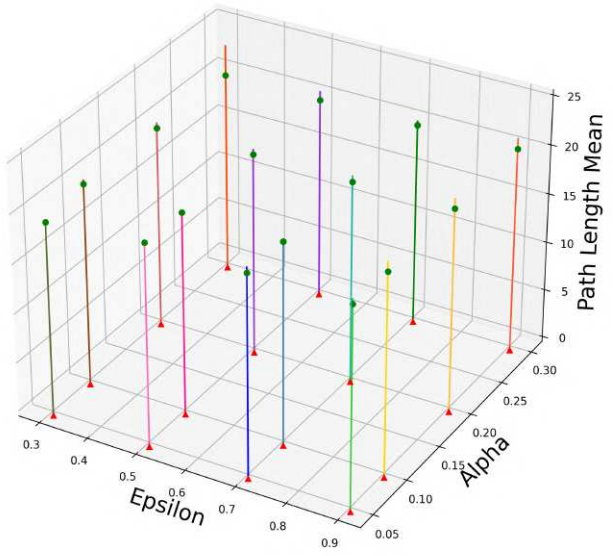
A.1.4 3D Plots for Main Path

Regular 3D plots will make way for an in-depth analysis of the evolution of each metric throughout training. Figures 59 through 61 illustrate the evolution of these metrics concerning the values of ϵ and α throughout the training regimen. It is noteworthy that for certain hyperparameter configurations, the values of ϵ and α decay over time, with this variation serving as an indicator of episode progression. In subfigures (b), (c), and (d), it is clear that these values initiate at zero and progressively increase. This behavior stems from the number of episodes required to obtain a viable path originating from the starting cell. The longer the length of this zero value phase in the training, the longer it took to find the first feasible path since during the initial stages of training, there is no clear path from the starting cell to the goal. To mitigate the inherent noise in the values, we applied a moving average window with a span of 100 episodes.

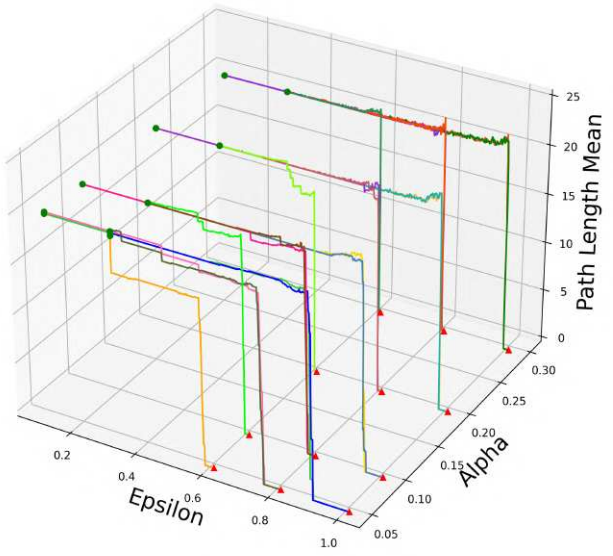
A.1.4.1 Main Path Path Length Metric Analysis

Figure 59 depicts the mean Path Length metric computed for paths associated with every cell in the map, where sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively.

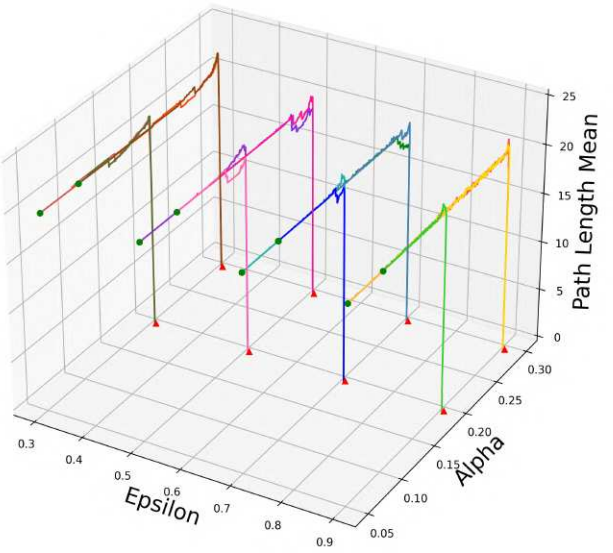
MP PL - Constant ϵ and α values Figure 57 (a) shows the variation throughout training for constant ϵ and α values for the PL metric regarding MP. Given that the passing of time was represented by the decay of either of the hyper parameters, this graph will only provide the final value (red triangle) and final one (green circle). The overshoots in this image are only apparent for values of α of 0.2 and above, with the highest one happening when pairing with $\epsilon = 0.3$, where there is a high update steps of the Q-value, with low exploration ratings.



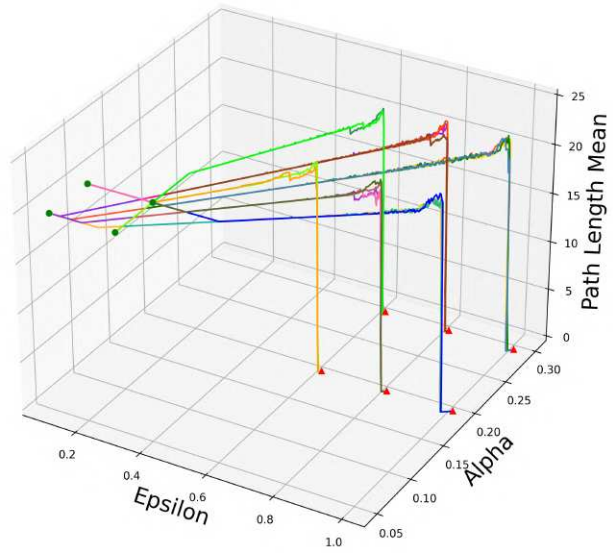
(a) Constant ϵ and α values.



(b) Variable ϵ and constant α values.



(c) Constant ϵ and variable α values.



(d) Variable ϵ and α values.

Figure 59 – Main Path Path Length average throughout training. Sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively. Starting value is represented by a red triangle and final value by a green circle.

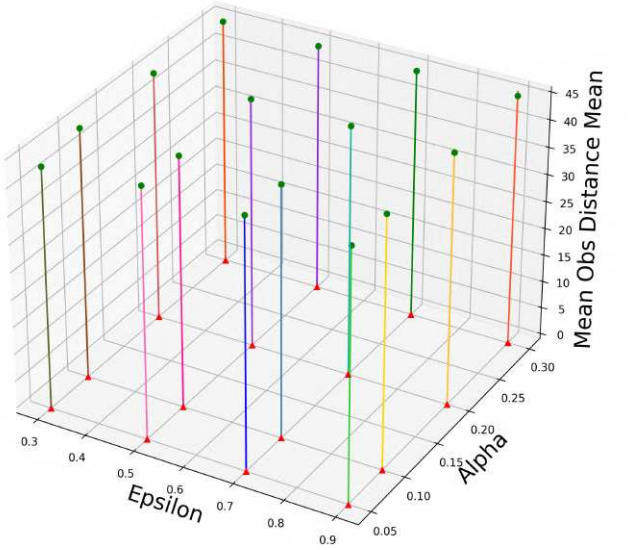
MP PL - Variable ϵ and constant α values Figure 59 (b) shows the variation throughout training for variable ϵ and constant α values for the PL metric regarding MP. This image provides an insight into the evolution of the path length metric over the course of episodes for varying ϵ and constant α values. Notably, there is an overshoot in scenarios featuring higher α values, especially with elevated initial ϵ values. Simulations employing higher α values also tend to identify feasible paths more rapidly, as they exhibit a smaller proportion of zero values. The simulations

with the quickest path discovery featured an initial ϵ value of 0.6 and constant α values of 0.2 and 0.3. A comparison between these two simulations reveals that the one with an α value of 0.2 initially identifies a shorter path but subsequently requires additional episodes to optimize not only path length but also the other two priorities. In contrast, when utilizing a learning rate of 0.3, the first path discovered exhibits a path length closer to the final value, suggesting that a higher learning rate guides the agent towards learning a path with higher scores on each of the three priorities. Lower α values correspond to an extended duration for identifying a feasible path, particularly with elevated initial ϵ values. These simulations also exhibit a plateau phase before reaching the final path, indicating that they initially discover a shorter path and subsequently explore solutions that offer superior scores across all metrics.

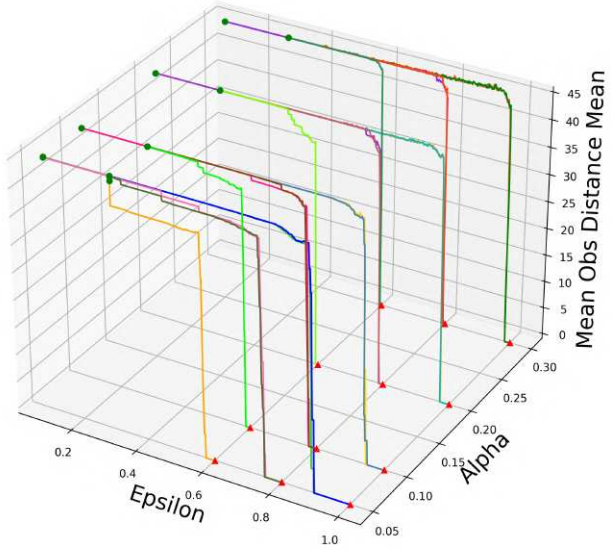
MP PL - Constant ϵ and variable α values - In Figure 59 (c), the agent identifies a viable path from the starting cell to the goal more rapidly than in the previous sub-figure. Overshoots are more prominent with lower ϵ values and higher initial α values. Moreover, the tendency to initially discover shorter yet suboptimal paths concerning all three priorities is less pronounced. This behavior is entirely absent with higher values of ϵ and manifests with reduced intensity compared to Figure 59 (b) when lower ϵ values are employed. This underscores the impact of constant learning rates on this aspect. The simulations that exhibit a sharp overshoot with a lengthier attenuation period are those with an ϵ value of 0.3, suggesting that lower exploration ratios swiftly lead to longer paths that require more time to optimize while considering all priorities.

MP PL - Variable ϵ and α values In Figure 59 (d), where both hyperparameters exhibit variable values, it is possible to observe results that require more time to discover an initial path than seen in Figure 59 (c), yet they are faster than those in Figure 59 (b). The overshoots are slightly more prominent with marginally extended attenuation times compared to the previous two analyses. Finally, the tendency to discover shorter yet suboptimal paths concerning all three priorities is substantially less pronounced in this context.

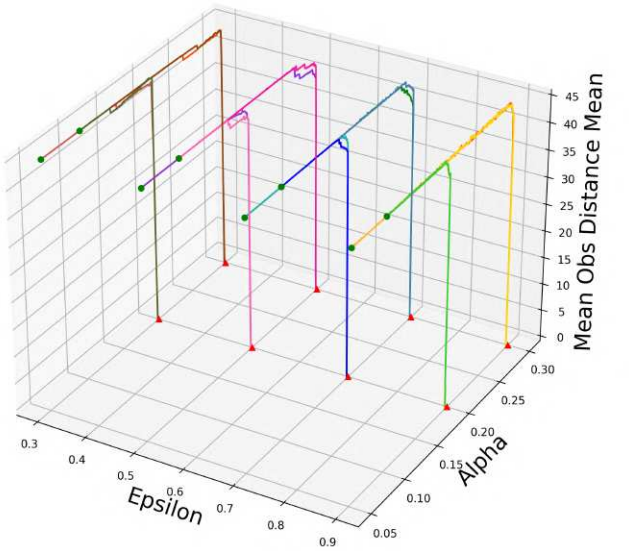
Overall it is possible to see faster convergence a substantially softer oscillations when compared to any AP metric. An interesting behavior arises with lower alpha values in the for of an early plateau, where the agent finds an initial feasible path, then further optimizes it taking in account other priorities, increasing the total PL in scenarios with constant ϵ and variable α as well as variable ϵ and constant α. This behavior is not observed with both hyperparameters taking variable values. In the next section it will be discussed the MP metrics for MOD.



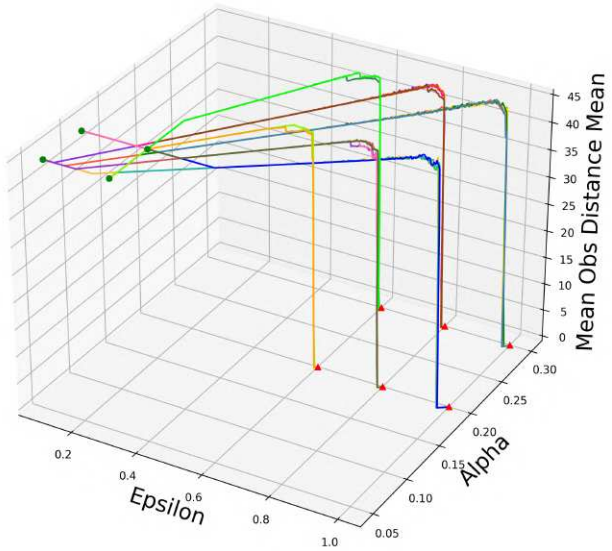
(a) Constant ϵ and α values.



(b) Variable ϵ and constant α values.



(c) Constant ϵ and variable α values.



(d) Variable ϵ and α values.

Figure 60 – Main Path Mean Obstacle Distance average throughout training. Sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively. Starting value is represented by a red triangle and final value by a green circle.

A.1.4.2 Main Path Mean Obstacle Distance Metric Analysis

Figure 60 depicts the Mean Obstacle Distance metric computed for paths associated with every cell in the map, where sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively.

MP MOD - Constant ϵ and α values Figure 57 (a) shows the variation throughout training for constant ϵ and α values for the MOD metric regarding MP. Given that the passing of time was represented by the decay of either of the hyper parameters, this graph will only provide the final value (red triangle) and final one (green circle). It is possible to see that there is no overshoot in any episode. This may happen given that overshoots tend to be associated with suboptimal results, and the suboptimal results in this metric are lower values, therefore they would not show with higher values than the last one.

MP MOD - Variable ϵ and constant α values Figure 59 (b) shows the variation throughout training for variable ϵ and constant α values for the MOD metric regarding MP. This image highlights the trend of initially discovering a less optimal path before subsequently optimizing for all three priorities, particularly evident in scenarios with lower values of α and initial ϵ. Notably, a discernible "plateau" shape is observed, rather than the "stair" shape, especially with an α value of 0.05 and lower initial ϵ values. This shape indicates that this behavior with respect to this metric is mitigated by higher exploration rates.

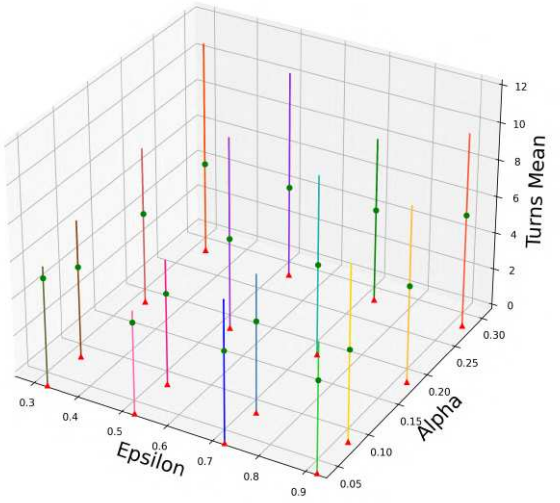
MP MOD - Constant ϵ and variable α values - In Figure 59 (c), it is possible to see that for constant values of ϵ and variable values of α there is substantially less of the pattern of finding a suboptimal path first and having to adapt it to better fit the other priorities. Also it reinforces that this "plateau" shape is more prevalent seen with lower values of ϵ.

MP MOD - Variable ϵ and α values In Figure 59 (d), where both hyperparameters exhibit variable values, the progression over episodes becomes notably smoother, with minimal occurrence of "plateau" shaped graphs. This observation suggests that striking a balance between these variations may lead to earlier instances of feasible paths more closely aligning with the final path concerning this metric

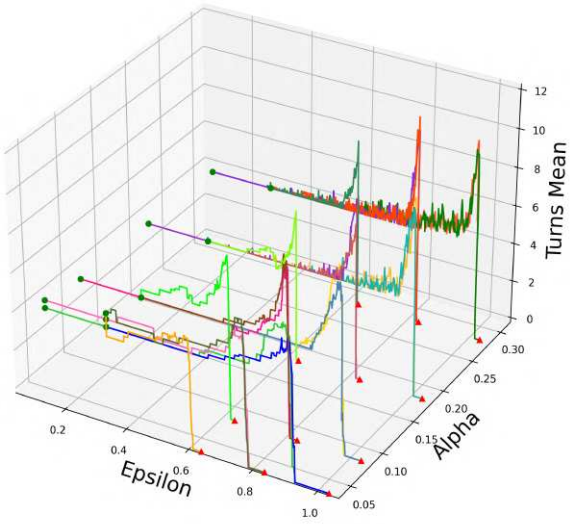
Overall, it is possible to see a much softer behavior when compared to the AP MOD metric as well as when compared to the MP PL. Given that this analysis comprehends only the MP metrics, there are less possible configurations for possible obstacle distances than when taking in account AP, which can explain this behavior. In the next section it will be discussed the AP metrics for TT.

A.1.4.3 Main Path Total Turns Metric Analysis

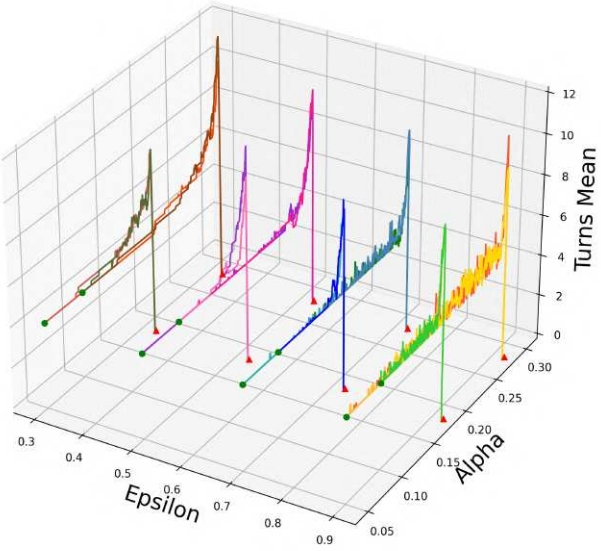
Figure 61 depicts the Total Turns metric computed for paths associated with every cell in the map, where sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively.



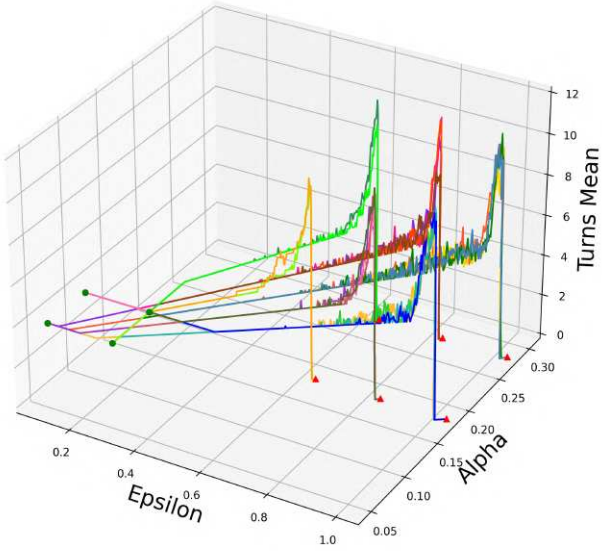
(a) Constant ϵ and α values.



(b) Variable ϵ and constant α values.



(c) Constant ϵ and variable α values.



(d) Variable ϵ and α values.

Figure 61 – Main Path Total Turns average throughout training. Sub figures (a), (b), (c) and (d) show the results for constant ϵ and α values, variable ϵ and constant α values, constant ϵ and variable α values and variable ϵ and α values, respectively. Starting value is represented by a red triangle and final value by a green circle.

MP TT - Constant ϵ and α values Figure 57 (a) shows the variation throughout training for constant ϵ and α values for the TT metric regarding MP. Given that the passing of time was represented by the decay of either of the hyper parameters, this graph will only provide the final value (red triangle) and final one (green circle). In this image, it is possible to see overshoots when using low ϵ which are severely increased when paired with high α values.

MP TT - Variable ϵ and constant α values Figure 59 (b) shows the variation throughout

training for variable ϵ and constant α values for the TT metric regarding MP. An oscillatory pattern is evident across all simulations, with higher amplitude, higher frequency "spikes" corresponding to larger α values, and lower frequency "step" shapes associated with smaller α values. These oscillations persist until the final stages of training in cases with α values of 0.1 and 0.05, indicating that this metric is among the last priorities to be adequately optimized during training. This phenomenon may be attributed to the fact that modifying the number of turns in a path represents a more significant step of variation concerning the maximum number of turns when compared to the other two metrics. Additionally, higher overshoots are observable in cases with larger learning rates due to the more substantial and abrupt update steps in each iteration.

MP TT - Constant ϵ and variable α values - In Figure 59 (c), it is possible to see that for constant values of ϵ and variable values of α, distinct oscillation patterns emerge for constant ϵ and variable α values in comparison to the previous analysis. This discrepancy arises when α reaches lower values, such as 0.05 or 0.1, at which point the agent has acquired sufficient knowledge for sound decision-making, avoiding the "step" shapes or multiple-peaked oscillations. The trend of higher peaks and longer attenuation times for lower ϵ values is similarly observed for this metric, and the presence of noise is more pronounced here than in other metrics pertaining to the main path.

MP TT - Variable ϵ and α values In Figure 59 (d), where both hyperparameters exhibit variable values, it is demonstrated a common pattern characterized by a pronounced overshoot peak and an oscillatory attenuation. This oscillatory behavior ceases in earlier stages compared to the preceding two analyses. An exception is found in the yellow plot with initial values of 0.2 for α and 0.6 for ϵ. This plot exhibits a "step" shape in the descent to convergence, and its attenuation period is notably longer, with a lower peak overshoot. These observations suggest that the lower starting values for the hyperparameters lead to less aggressive exploration and more cautious decision-making, preventing the agent from lingering in the same number of turns for prolonged periods during training.

In the MP TT metric, it is possible to see that noise is much more present than on the other metrics for MP, as well as stronger overshoots that take longer to settle. This can be attributed to this metric having such a large difference each step it changes its value in respect to the maximum possible value. For example, the highest obtained value was was near 10, and each step changing this metric is at least 1 turn in size, which represents approximately 10% of the maximum value, being much larger than the variation steps in the other metrics.

Overall when compared to the analysis of the metrics for paths from every cell in the map, the examination of the main path demonstrates a swifter convergence and a notable reduction in overshoot occurrences, except in the case of the Number of Turns metric. This outcome is not surprising, as there is considerably more flexibility for refining solutions when optimizing multiple paths. It is worth noting that the infrequent presence of "plateau" shaped graphs in the metrics for all paths signifies situations where the agent becomes temporarily entrapped in suboptimal solutions. These isolated instances are averaged out across all cells, thus mitigating this behavior in the presented graphs. In the next section, metrics using Box Plots will be discussed.

A.1.5 Box Plots for All Paths Metrics

Box plots were used in order to have insights about the metrics distribution over the 10 repeated simulations. The final value regarding AP for each metric as well as the number of episodes to reach the pre determined performance threshold for each of them are shown in Figures 62-64. The indexes in the x axis in the plots are numerical values that represent the combination of ϵ and α values used for that simulation. These indexes can be found in Table 14.

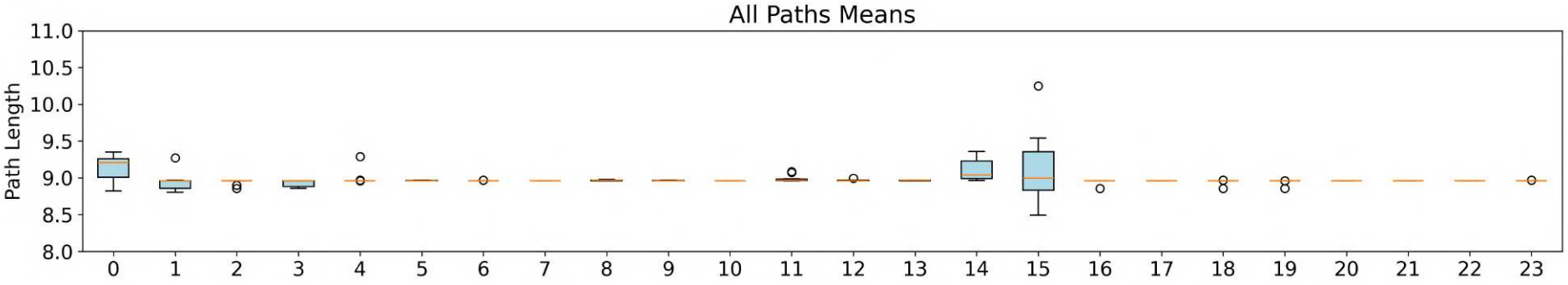
A.1.5.1 All Paths Path Length metric

Figure 62 depicts the Path Length metric computed for paths associated with every cell in the map, where the indexes and groups are detailed in Table 14.

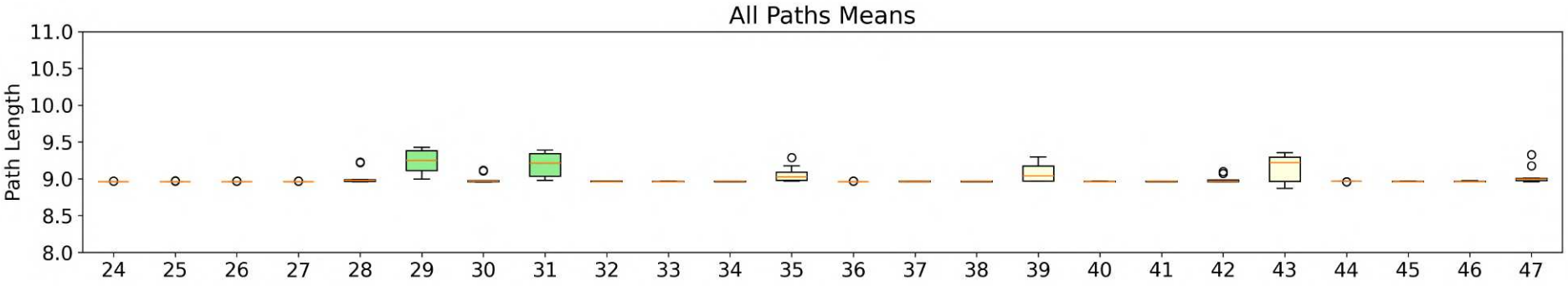
AP PL - Group A - Constant ϵ and α values Figure 62 (a) shows the simulations results of Group A in Idx_{0-15}. It is possible to see very little dispersion over the results, with the few significant examples on Idx_0, which uses a high value for ϵ and Idx_{15}, which combines a low value of $\epsilon = 0.5$ with a low value of $\alpha = 0.05$. This may happen due to high variety of action choices in Idx_0 leading to inconsistent results and low variety of actions paired with a low learning step in Idx_{15}, suggesting that this level of exploration does not consistently yield the same results.

AP PL - Group B - Constant ϵ and variable α values Figures 62 (a) and (b) shows the simulations results of Group B in Idx_{16-31}. When it comes to this group, some variance was only observed for ϵ=0.3, with intervals of $\alpha \in [0.05, 0.2]$ and $\alpha \in [0.1, 0.2]$ in $Idx_{29,31}$. This suggests that with low exploration, lower values of learning rate can lead the agent to struggle to show higher consistency in its results.

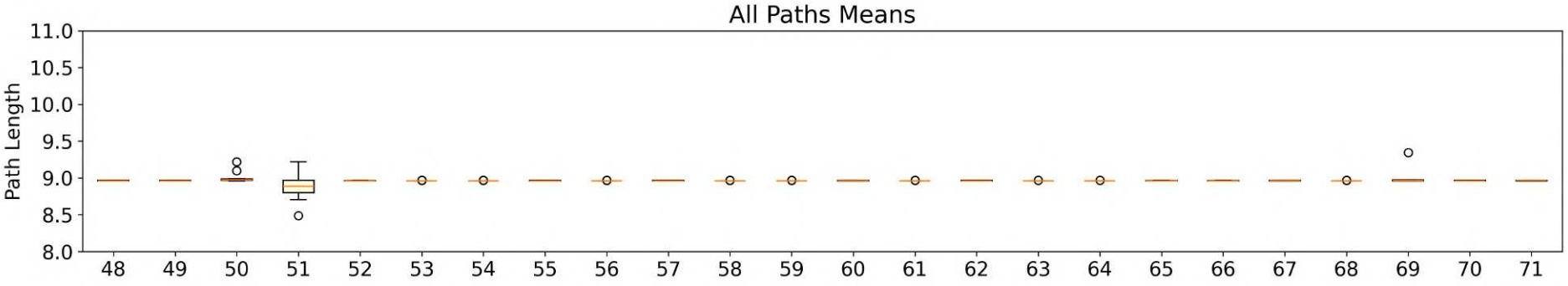
AP PL - Group C - Variable ϵ and constant α values - Figures 62 (b) and (c) shows the simulations results of Group C in Idx_{32-51}. Significant dispersion are found with low α values with higher inconsistency between repetitions for higher ϵ intervals such as



(a) Results for Groups A and B.



(b) Results for Groups B and C.



(c) Results for Groups C and D.

Figure 62 – Box Plot for All Paths Path Length average metric. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively.

seen in $Idx_{35,39,43}$, which have ϵ starting at 1 or 0.8. With a larger interval, it comes a faster decay, these results suggest that lower α values don't work as well as higher ones with higher ϵ intervals.

AP PL - Group D - Variable ϵ and α values Figure 62 (c) shows the simulations results of Group D in Idx_{52-71}. There was no significant dispersion in these simulations, which suggests that variable intervals for both hyperparameters lead to more consistent results. This implies that higher update steps in the exploration phase of training can foster a crude knowledge of the environment that can be successfully fine tuned in later stages of training using smaller update steps with lower exploration ratios.

Overall there was little dispersion in the AP PL metric results. The main cases that stood out either had high constant exploration/exploitation ratio of to little of a learning rate, indicating that most inconsistencies for this metric arises from high variety of choices through training or little update steps with low exploration. In the next section, the results for the AP MOD metric will be discussed.

A.1.5.2 All Paths Mean Obstacle Distance metric

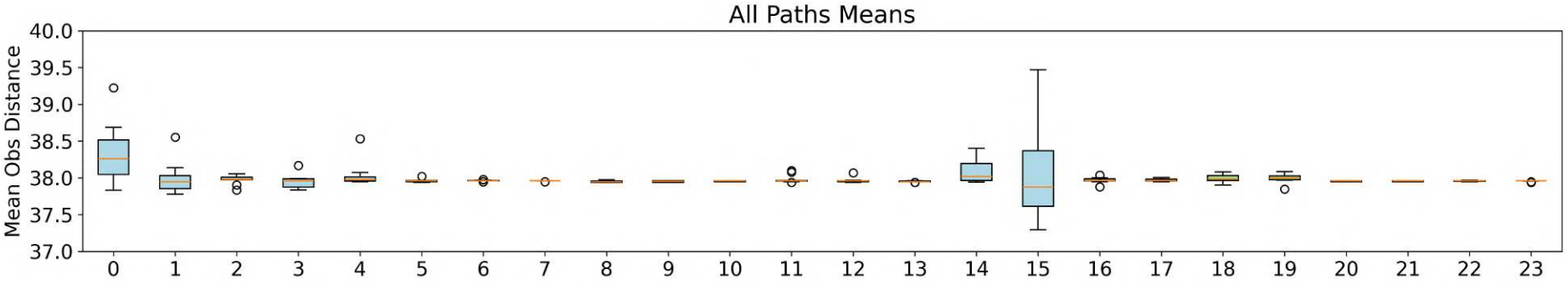
Figure 63 depicts the Mean Obstacle Distance metric computed for paths associated with every cell in the map, where the indexes and groups are detailed in Table 14. In the next section, the AP MOD metric results will be discussed,

AP MOD - Group A - Constant ϵ and α values Figure 63 (a) shows the simulations results of Group A in Idx_{0-15}. It is possible to see very little dispersion over the results just like the AP PL metric, with the few significant examples on the same simulations, $Idx_{0,15}$ on Idx_0, which uses a high value for ϵ and Idx_{15}.

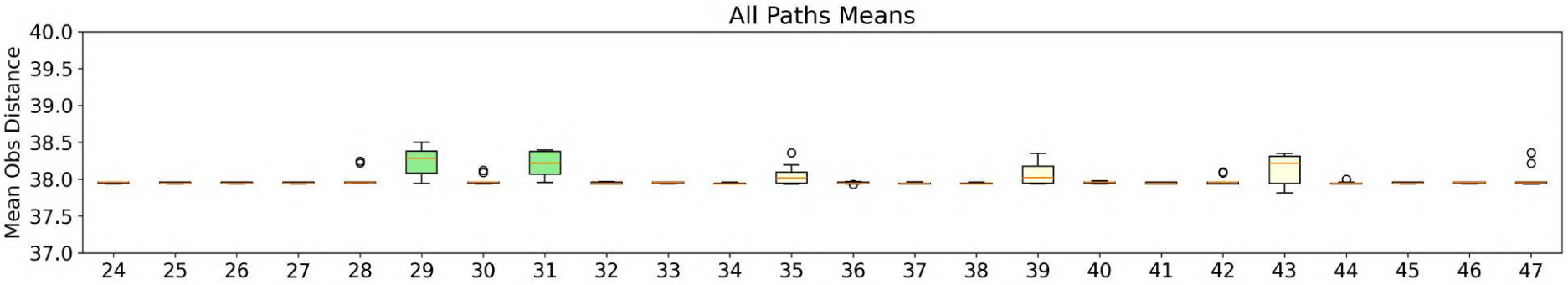
AP MOD - Group B - Constant ϵ and variable α values Figures 63 (a) and (b) shows the simulations results of Group B in Idx_{16-31}. When it comes to this group, some variance was only observed in $Idx_{29,31}$, just as seen in the AP PL metric. This suggests that with low exploration, lower values of learning rate can lead the agent to struggle to show higher consistency in its results also regarding safety concerns.

AP MOD - Group C - Variable ϵ and constant α values - Figures 63 (b) and (c) shows the simulations results of Group C in Idx_{32-51}. Significant dispersion are found with low α values with higher inconsistency between repeated simulations for higher ϵ intervals such as seen in $Idx_{35,39,43}$.

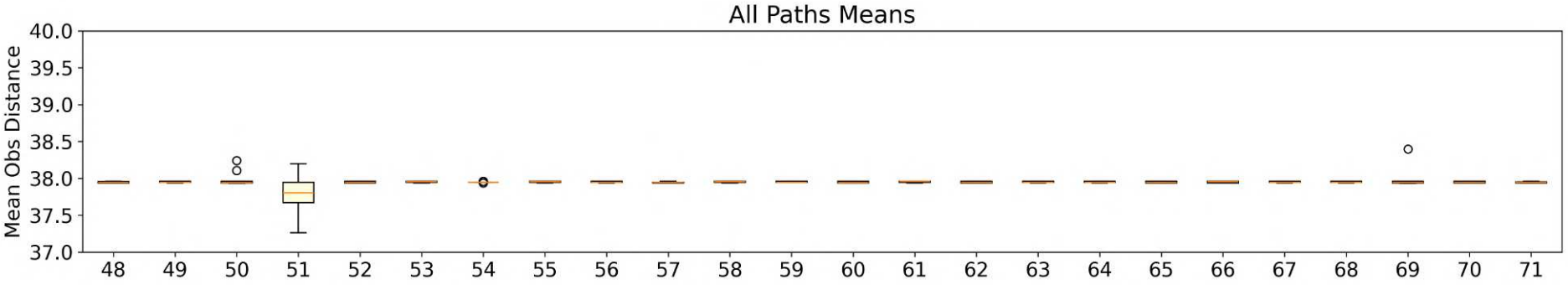
AP MOD - Group D - Variable ϵ and α values Figure 63 (c) shows the simulations results of Group D in Idx_{52-71}. There was no significant dispersion in these simulations,



(a) Results for Groups A and B.



(b) Results for Groups B and C.



(c) Results for Groups C and D.

Figure 63 – Box Plot for All Paths Mean Obstacle Distance average metric. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively.

suggesting that a well balanced decaying interval for both hyperparameters leads to consistent results.

There is much similarity between the results of the AP MOD metric with the previous AP PL one, with the main difference being that, when it comes to the MOD metric, the results tend to have a higher dispersion spread, such as seen in $Idx_{0,15}$ when comparing both metrics. This may arise from the more numerous possible result combinations of this metric. In the next section, the results for the AP TT metric will be discussed.

A.1.5.3 All Paths Total Turns metric

Figure 64 depicts the Total Turns metric computed for paths associated with every cell in the map, where the indexes and groups are detailed in Table 14.

AP TT - Group A - Constant ϵ and α values Figure 64 (a) shows the simulations results of Group A in Idx_{0-15}. It is possible to see a high dispersion in Idx_0 a,d a less prominent one in Idx_{15}, which represents the both ends of t he spectrum for this group. Where the first one has high values for both hyperparameters and the second has low values form them, suggesting that both extremes have negative impacts in this metric consistency.

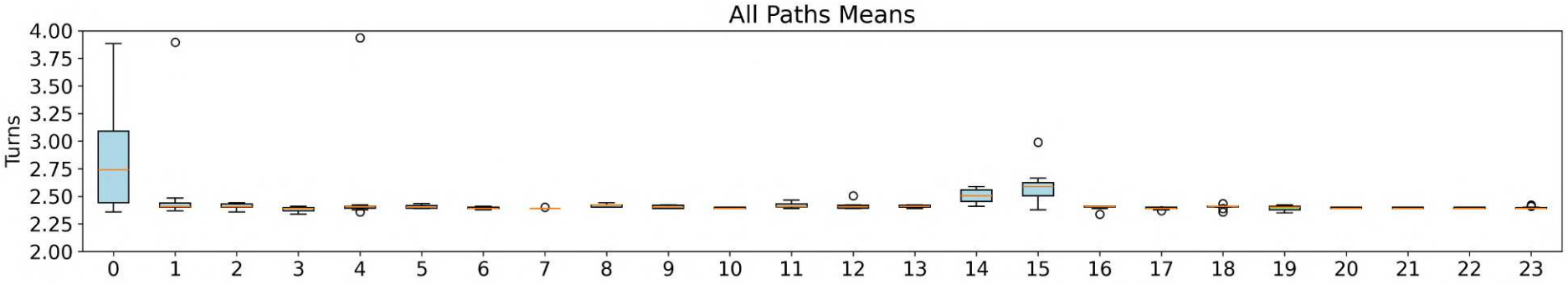
AP TT - Group B - Constant ϵ and variable α values Figures 64 (a) and (b) shows the simulations results of Group B in Idx_{16-31}. Similar to the other two metrics, some variance was only observed in $Idx_{29,31}$. This suggests that with low exploration, lower values of learning rate can lead the agent to struggle to show higher consistency in its results also regarding energy concerns.

AP TT - Group C - Variable ϵ and constant α values - Figures 64 (b) and (c) shows the simulations results of Group C in Idx_{32-51}. Significant dispersion are found with low α values with higher inconsistency between repeated simulations for higher ϵ intervals such as seen in $Idx_{35,39,43,47}$.

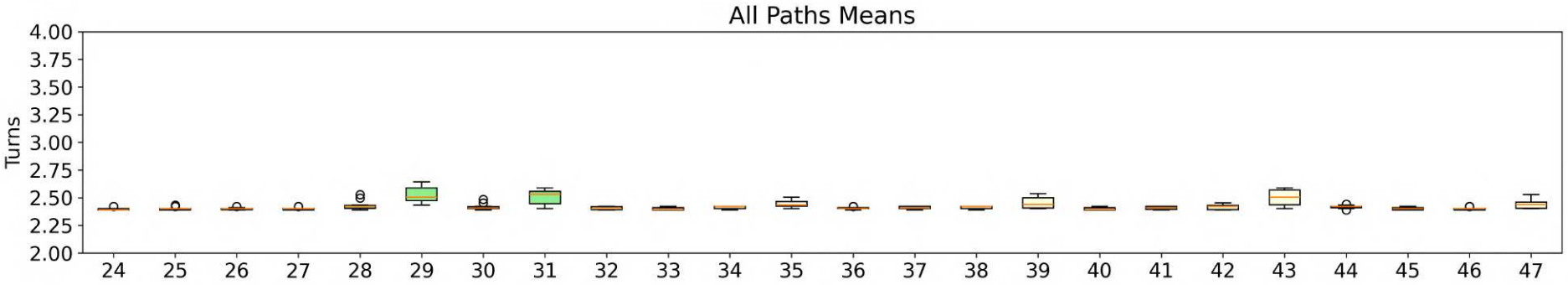
AP TT - Group D - Variable ϵ and α values Figure 64 (c) shows the simulations results of Group D in Idx_{52-71}. There was no significant dispersion in these simulations, suggesting that a well balanced decaying interval for both hyperparameters leads to consistent results.

Overall, the AP TT metric had similar behavior than the other ones, with the only stand out case being Idx_0, where it was possible to observe the highest degree of dispersion between all metrics.

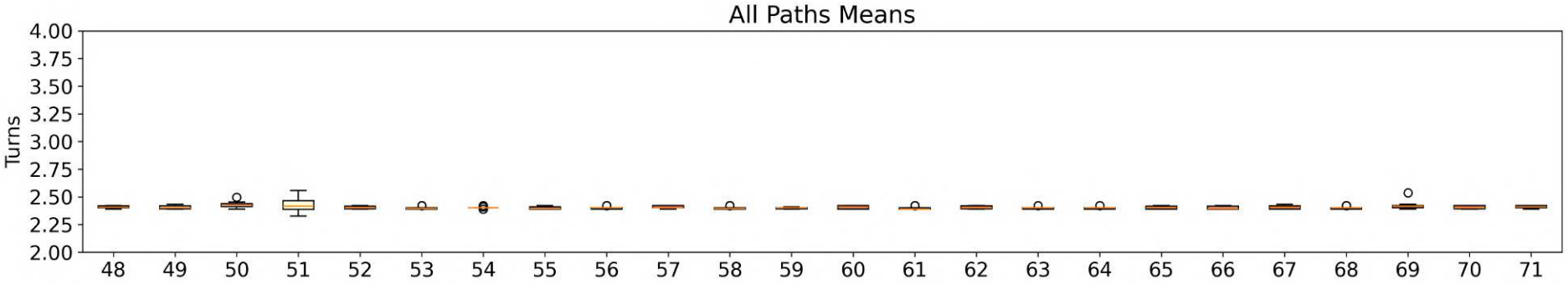
In regards to dispersion in AP metrics, there was few cases with significant inconsistencies between the simulation results, indicating that most hyperparameter configurations



(a) Results for Groups A and B.



(b) Results for Groups B and C.



(c) Results for Groups C and D.

Figure 64 – Box Plot for All Paths Total Turns average metric. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively.

lead to the same results throughout the repeated simulations. In the next section the metrics for the Main Path will be discussed.

A.1.6 Box Plots for Main Path Metrics

Box plots were used in order to have insights about the metrics distribution over the 10 repeated simulations. The final value regarding MP for each metric as well as the number of episodes to reach the pre determined performance threshold for each of them are shown in Figures 65-67. The indexes in the x axis in the plots are numerical values that represent the combination of ϵ and α values used for that simulation. These indexes can be found in Table 14. They were grouped in four groups, A, B, C and D, respectively, which will be plotted, in this order, with the following colors: light blue, light green, light yellow and light coral, akin to how it was displayed in the previous section.

A.1.6.1 Main Path Path Length metric

Figure 65 depicts the Path Length metric computed for the Main Path, where the indexes and groups are detailed in Table 14.

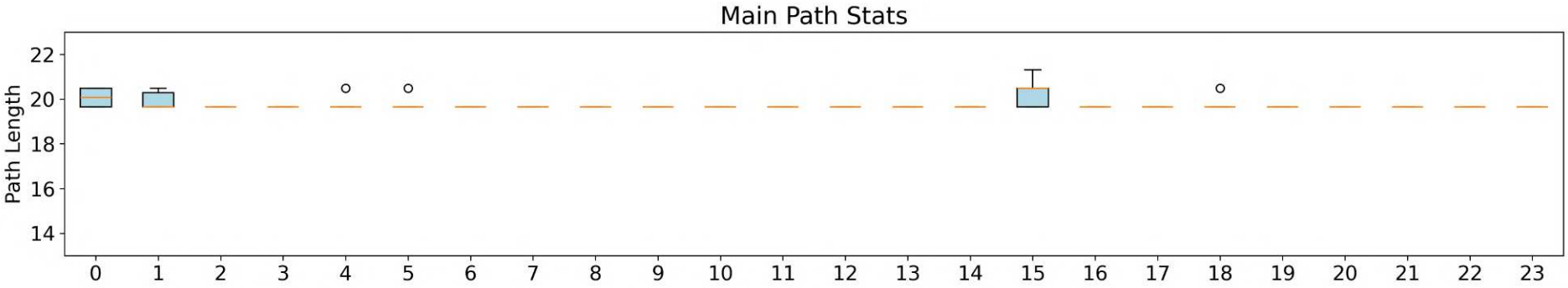
MP PL - Group A - Constant ϵ and α values Figure 65 (a) shows the simulations results of Group A in Idx_{0-15}. There is only observable dispersion in $Idx_{0,1,15}$ which either have high values for both ϵ and α, or very low values for both, indicating that this metric does not present consistent results in the extreme configurations of these hyper parameters.

MP PL - Group B - Constant ϵ and variable α values Figures 65 (a) and (b) shows the simulations results of Group B in Idx_{16-31}. The was no substantial dispersion in this group.

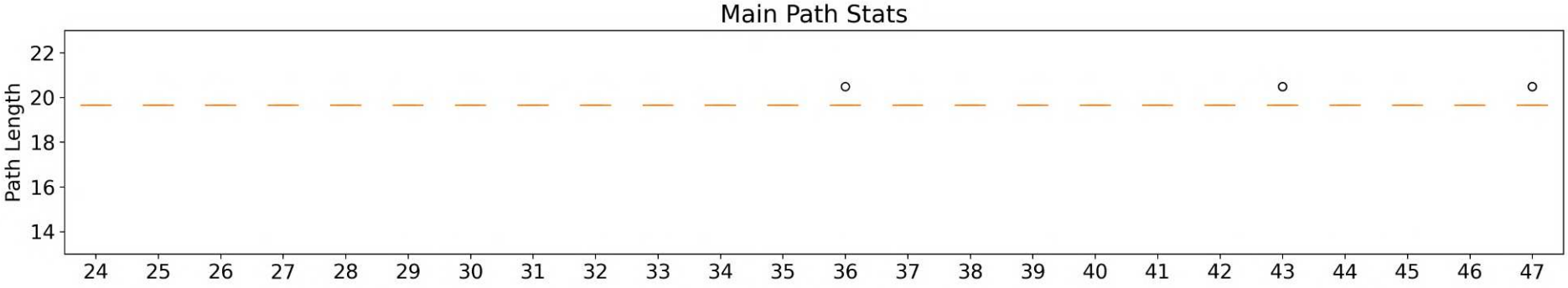
MP PL - Group C - Variable ϵ and constant α values - Figures 65 (b) and (c) shows the simulations results of Group C in Idx_{32-51}. The only observable case of dispersion is found in Idx_{51}, which uses $\epsilon \in [0.3, 0.6]$ and $\alpha = 0.05$. This suggests that with such low overall exploration rates, paired with a low learning rate, the agent does not learn consistently.

MP PL - Group D - Variable ϵ and α values Figure 65 (c) shows the simulations results of Group D in Idx_{52-71}. There was no substantial dispersion in this group.

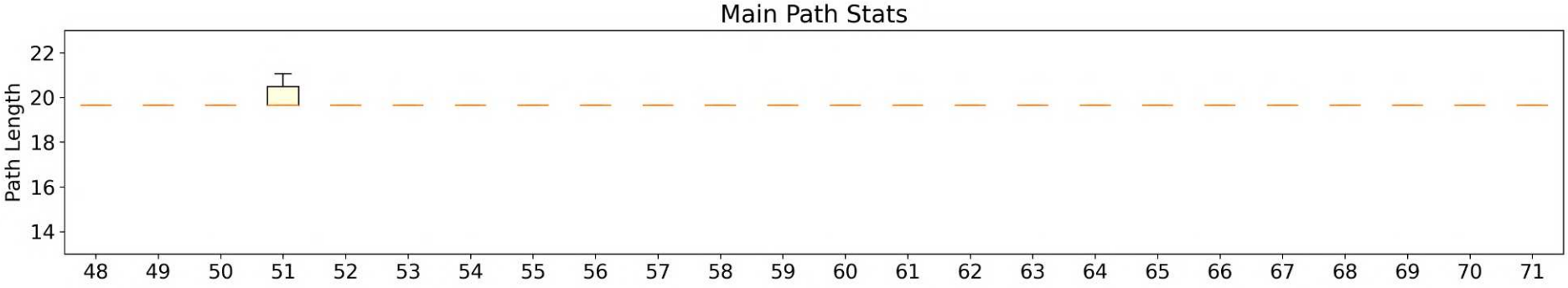
Overall there the MP PL metric shows much more consistent results, when compared to the AP metrics. This is expected given that it is a simpler task to optimize just the path from the starting cell to the destination. In the next section, the MP MOD metric is discussed.



(a) Results for Groups A and B.



(b) Results for Groups B and C.



(c) Results for Groups C and D.

Figure 65 – Box Plot for Main Path Path Length average metric. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively.

A.1.6.2 Main Path Mean Obstacle Distance metric

Figure 66 depicts the Mean Obstacle Distance metric computed for the Main Path, where the indexes and groups are detailed in Table 14.

MP MOD - Group A - Constant ϵ and α values Figure 66 (a) shows the simulations results of Group A in Idx_{0-15}. It is possible to see consistent results throughout all simulations with some dispersion on $Idx_{0,1,15}$, which have examples of the highest and lowest possible values for ϵ.

MP MOD - Group B - Constant ϵ and variable α values Figures 66 (a) and (b) shows the simulations results of Group B in Idx_{16-31}. There was no substantial dispersion in this group.

MP MOD - Group C - Variable ϵ and constant α values - Figures 66 (b) and (c) shows the simulations results of Group C in Idx_{32-51}. Observable dispersion is only seen in Idx_{51}, which uses the lowest interval for ϵ as well as the lowest value for α.

MP MOD - Group D - Variable ϵ and α values Figure 66 (c) shows the simulations results of Group D in Idx_{52-71}. There was no substantial dispersion in this group.

The MP MOD metric showed more consistent results when compared to both MP PL as well as the AP metrics. This is expected given that it is a simpler task to optimize just the path from the starting cell to the destination. In the next section, the MP TT metric is discussed.

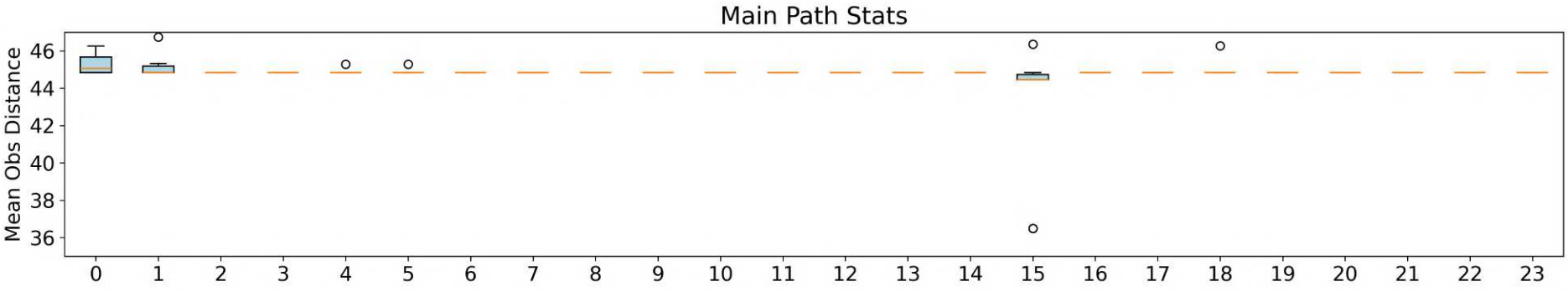
A.1.6.3 Main Path Total Turns metric

Figure 67 depicts the Total Turns metric computed for the Main Path, where the indexes and groups are detailed in Table 14.

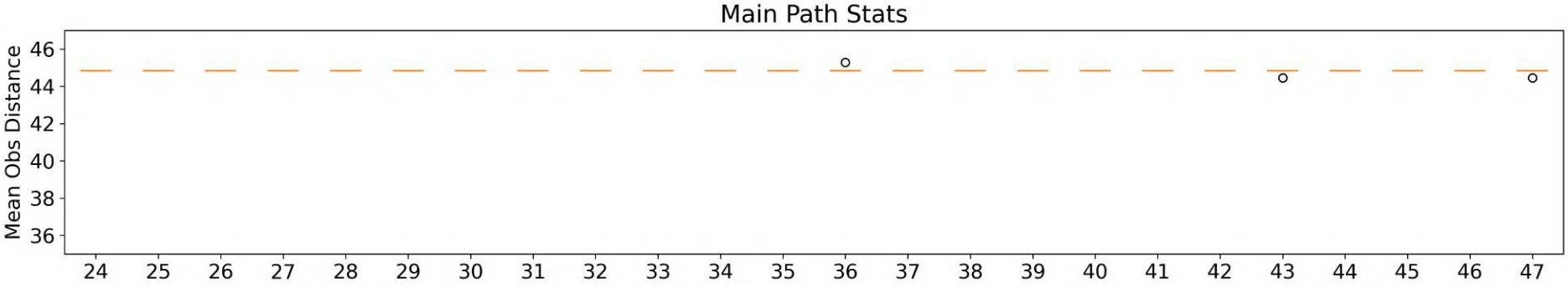
MP TT - Group A - Constant ϵ and α values Figure 67 (a) shows the simulations results of Group A in Idx_{0-15}. Substantial dispersion was observed on $Idx_{0,15}$, with high and low ϵ values respectively.

MP TT - Group B - Constant ϵ and variable α values Figures 67 (a) and (b) shows the simulations results of Group B in Idx_{16-31}. There was no substantial dispersion in this group.

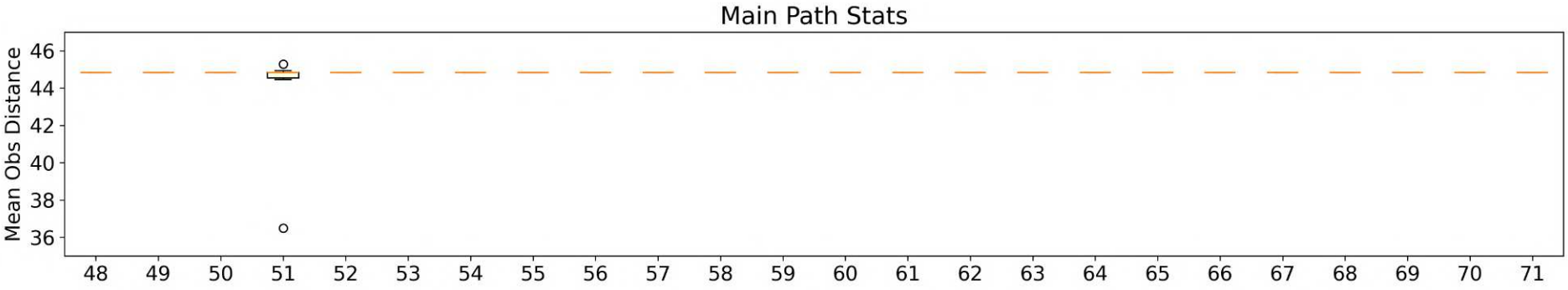
MP TT - Group C - Variable ϵ and constant α values - Figures 67 (b) and (c) shows the simulations results of Group C in Idx_{32-51}. Only Idx_{51} showed significant dispersion, due to lower ϵ intervals with lower learning rates.



(a) Results for Groups A and B.

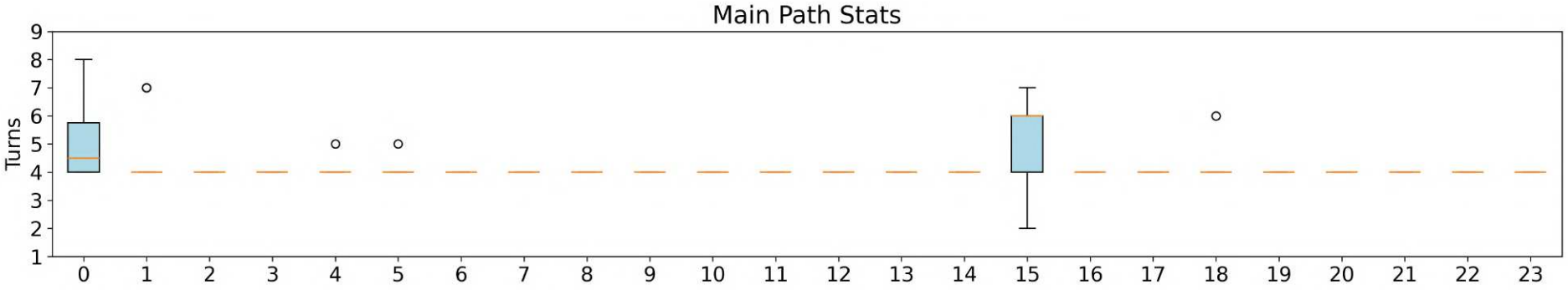


(b) Results for Groups B and C.

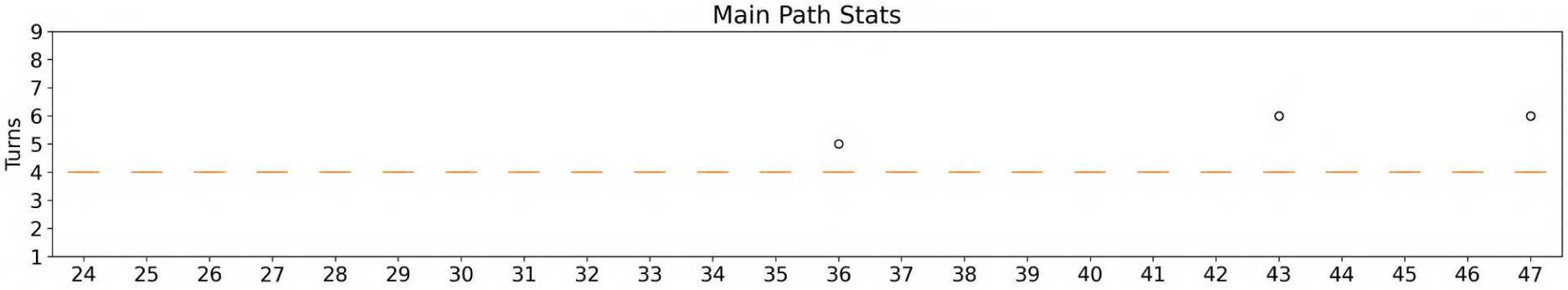


(c) Results for Groups C and D.

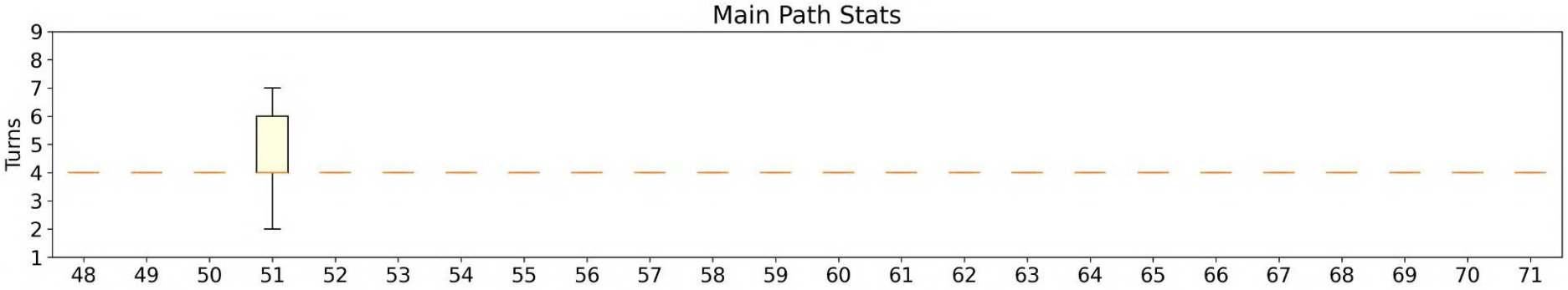
Figure 66 – Box Plot for Main Path Mean Obstacle Distance average metric. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively.



(a) Results for Groups A and B.



(b) Results for Groups B and C.



(c) Results for Groups C and D.

Figure 67 – Box Plot for Main Path Total Turns average metric. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively.

MP TT - Group D - Variable ϵ and α values Figure 67 (c) shows the simulations results of Group D in Idx_{52-71}. There was no substantial dispersion in this group.

There are few hyperparameter combinations that yield significant dispersion for the MP TT metric, but the few instances showed much higher dispersion range when compared to the other two MP metrics. This can be explained due to the variation step size off this metric when compared to the total value.

Overall for the MP metrics, it is possible to see more consistent results when contrasted with the AP metrics, given that it is a much simpler task to optimize these parameters for a single path than it is for multiple ones. In the next section, the thresholds for each metric will be analyzed.

A.1.7 Box Plots for All Paths Metrics Thresholds

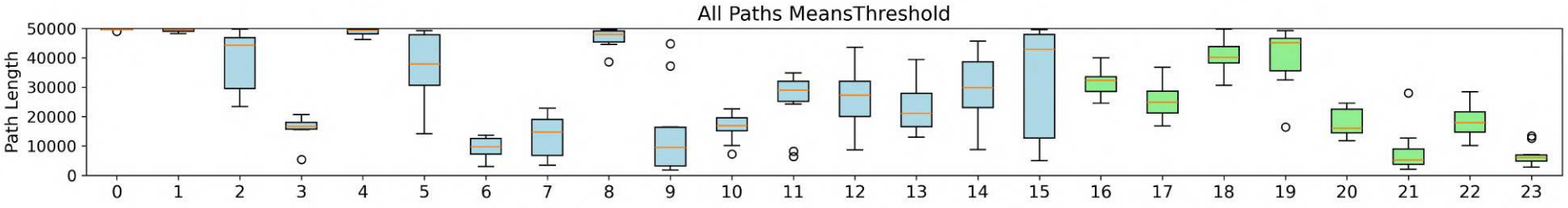
Box plots were used in order to have insights about the metric thresholds distribution over the 10 repeated simulations. These thresholds represent how fast each simulation reached and stayed in a region of ±2% of the best metric value. The number of episodes to reach the pre determined performance threshold for each of them are shown in Figures 68-70. The indexes in the x axis in the plots are numerical values that represent the combination of ϵ and α values used for that simulation. These indexes can be found in Table 14. They were grouped in four groups, A, B, C and D, respectively, which will be plotted, in this order, with the following colors: light blue, light green, light yellow and light coral, akin to how it was displayed in the previous section.

A.1.7.1 All Paths Path Length metric threshold

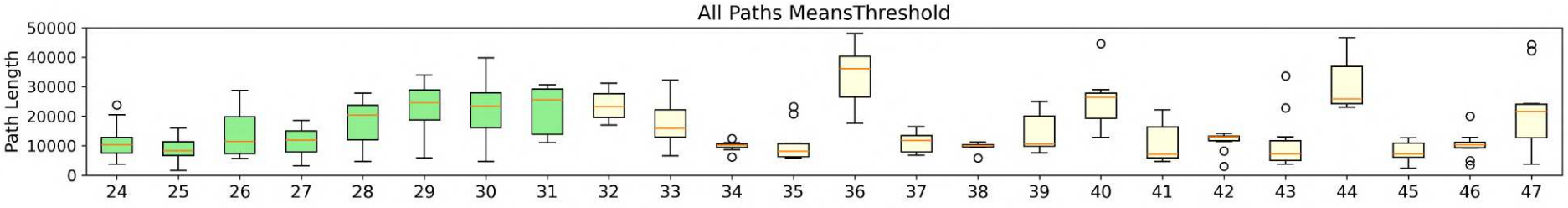
Figure 68 depicts the Path Length metric threshold computed for the Main Path, where the indexes and groups are detailed in Table 14.

AP PL Threshold - Group A - Constant ϵ and α values Figure 68 (a) shows the simulations results of Group A in Idx_{0-15}. With $\epsilon = 0.9$ and high α values such as seen in Idx_{0-2}, there is either high dispersion or high threshold values, meaning that in these instances, the agent struggled to converge consistently to its final value. Also the lower end for ϵ values such as seen in Idx_{12-15}, show high dispersion, which implies that the agent achieved the final value in different passes when using low exploration rates.

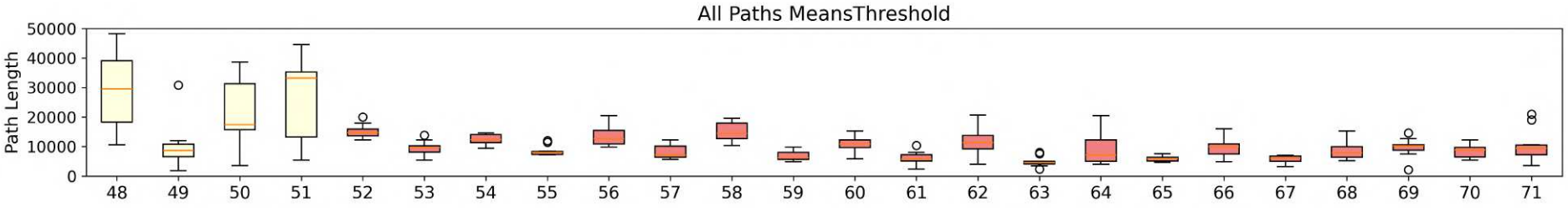
AP PL Threshold - Group B - Constant ϵ and variable α values Figures 68 (a) and (b) shows the simulations results of Group B in Idx_{16-31}. Similar to the previous group, the most divergent combinations where found int the higher indexes such as Idx_{28-31},



(a) Results for Groups A and B.



(b) Results for Groups B and C.



(c) Results for Groups C and D.

Figure 68 – Box Plot for All Paths Path Length metric threshold average. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively.

reinforcing that lower exploration rates for this metric leads to varying convergence speeds.

AP PL Threshold - Group C - Variable ϵ and constant α values - Figures 68 (b) and (c) shows the simulations results of Group C in Idx_{32-51}. Having a variable ϵ interval lead to not having high threshold values for the higher exploration rates such as seen in the first simulations in the previous two groups.c But the behavior of showing more varying thresholds for lower exploration rates such as seen in Idx_{47-51} is still present.

AP PL Threshold - Group D - Variable ϵ and α values Figure 68 (c) shows the simulations results of Group D in Idx_{52-71}. Using both hyperparameters with variable intervals substantially lowered the thresholds for this metric as well as softened the dispersion over the simulations.

When using constant ϵ values, there was a clear struggle to convergence due to high threshold values in the earlier indexes in groups A nd B, which was improved in groups C and D using variable ϵ values. The results in this section suggest that high exploration rates can lead to difficulties to either reach or maintain this metric close to its final value and also that lower exploration and/or learning rates can lead to inconsistent convergence speeds due to varying threshold values. Furthermore, variable values for both hyperparameters showed the best and most consistent results. In the next section, the AP MOD metric thresholds will be discussed.

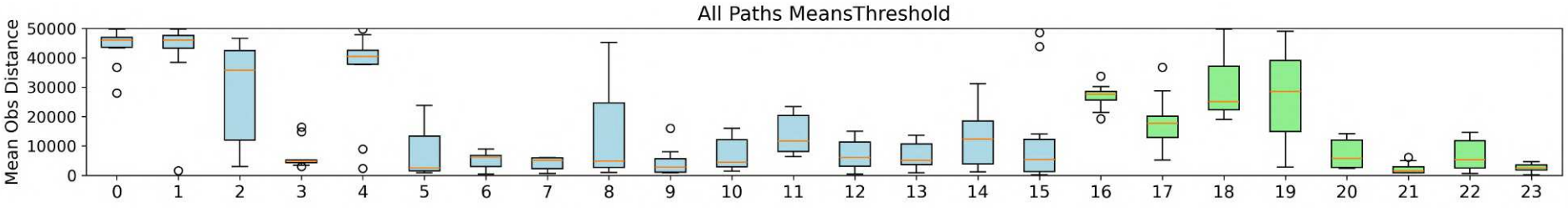
A.1.7.2 All Paths Mean Obstacle Distance metric threshold

Figure 69 depicts the Mean Obstacle Distance metric threshold computed for the Main Path, where the indexes and groups are detailed in Table 14.

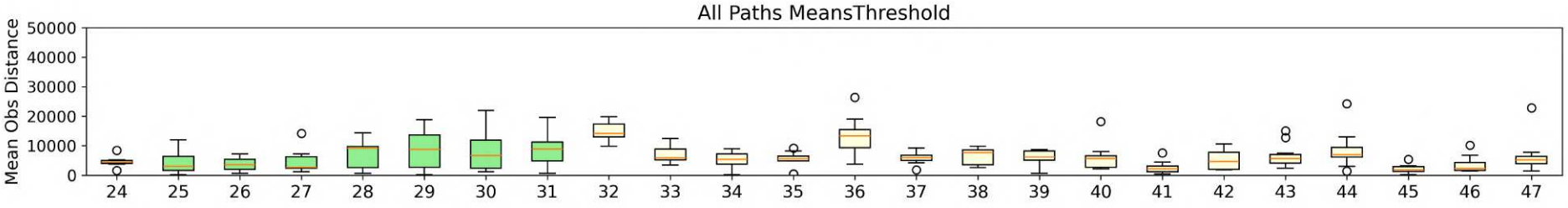
AP MOD Threshold - Group A - Constant ϵ and α values Figure 69 (a) shows the simulations results of Group A in Idx_{0-15}. Similarly to the behavior seen in the AP MP metric thresholds for this group, high ϵ values led to either high variance or high threshold values, indicating a struggle in convergence speed consistency.

AP MOD Threshold - Group B - Constant ϵ and variable α values Figures 69 (a) and (b) shows the simulations results of Group B in Idx_{16-31}. Higher exploration rates such as seen in Idx_{16-19} showed higher overall threshold values as well as more scattered results. Lower spread was found in using lower ϵ such as seen in Idx_{28-31}

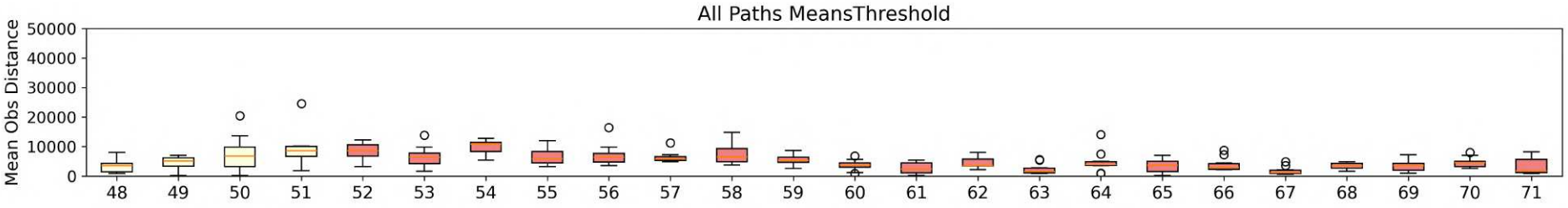
AP MOD Threshold - Group C - Variable ϵ and constant α values - Figures 69 (b) and (c) shows the simulations results of Group C in Idx_{32-51}. Introducing variable ϵ intervals led to a substantial decrease in both dispersion as well as threshold values,



(a) Results for Groups A and B.



(b) Results for Groups B and C.



(c) Results for Groups C and D.

Figure 69 – Box Plot for All Paths Mean Obstacle Distance metric threshold average. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively.

indicating that higher exploration in the earlier stages of training, paired with more exploitation on the latter stages yield more consistent convergence rates.

AP MOD Threshold - Group D - Variable ϵ and α values Figure 69 (c) shows the simulations results of Group D in Idx_{52-71}. Following the trend introduced in the previous group, using both hyperparameters with variable intervals resulted in lower thresholds for this metric as well as softened the dispersion over the simulations.

There was a great impact on this metric's thresholds throughout simulations when using variable ϵ intervals with overall performance improving slightly when paired with variable α values as well. Also there was more dispersion and highers threshold values than the previous metric. This may be attributed to the nature of the metric, as it considers the distance to the 10 closest obstacles, and small changes in the path can have a significant impact on the metric value, compared to PL. It may also indicate that, when considering all three priorities, optimizing MOD poses a more challenging task than finding a shorter path. Next section, the AP Total Turns metric thresholds will be discussed.

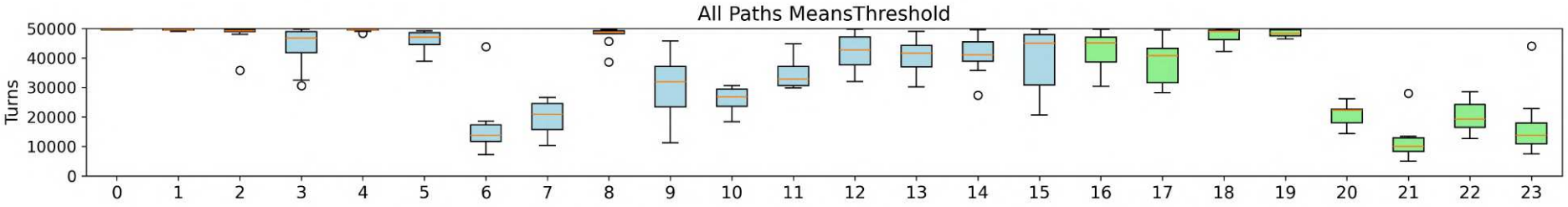
A.1.7.3 All Paths Total Turns metric threshold

Figure 70 depicts the Total Turns metric threshold computed for the Main Path, where the indexes and groups are detailed in Table 14.

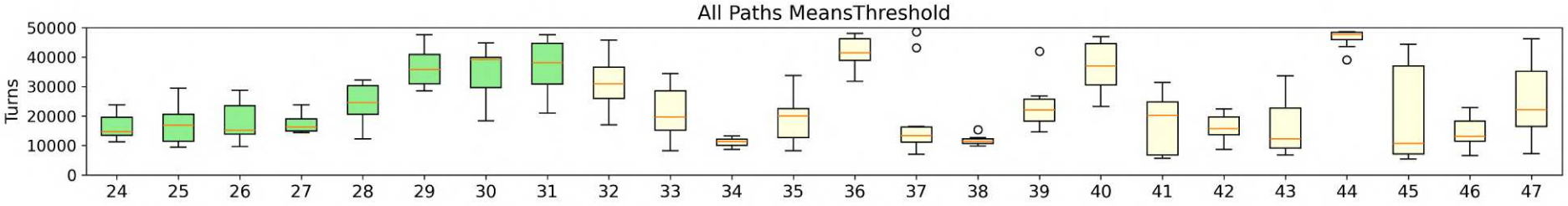
AP TT Threshold - Group A - Constant ϵ and α values Figure 70 (a) shows the simulations results of Group A in Idx_{0-15}. The most consistent results were found when pairing $\epsilon = 0.7$ with lower values of α such as seen in $Idx_{6,7}$, with overall higher dispersion and threshold values for combinations asside from this one, suggesting that this metric is more sensitive to hyperparameter variations in this group.

AP TT Threshold - Group B - Constant ϵ and variable α values Figures 70 (a) and (b) shows the simulations results of Group B in Idx_{16-31}. Introducing variable α values broadened the interval with more consistent results. Idx_{20-27} showed much lower dispersion and threshold values than the other combinations, suggesting that the highest and lowest exploration rates either lead to lower convergence speeds in this group.

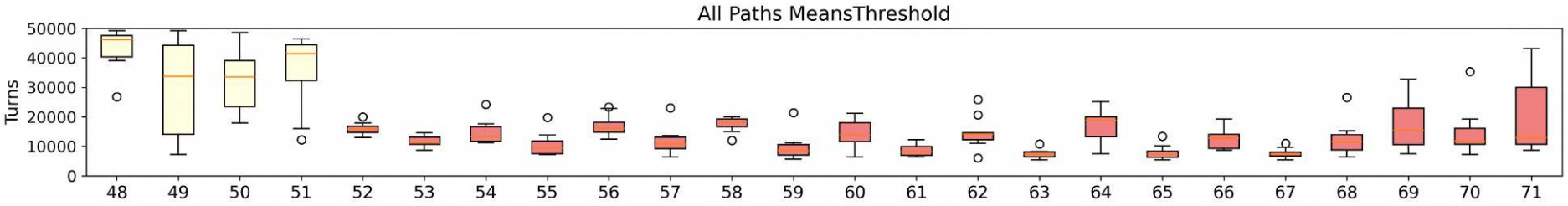
AP TT Threshold - Group C - Variable ϵ and constant α values - Figures 70 (b) and (c) shows the simulations results of Group C in Idx_{32-51}. Using variable ϵ and constant α values did not show the same behavior seen in the previous group, which leads to infer that this metric benefits much more of a variable learning rate than of a variable exploration rate.



(a) Results for Groups A and B.



(b) Results for Groups B and C.



(c) Results for Groups C and D.

Figure 70 – Box Plot for All Paths Total Turns metric threshold average. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively.

AP TT Threshold - Group D - Variable ϵ and α values Figure 70 (c) shows the simulations results of Group D in Idx_{52-71}. The combination of variable intervals for both hyperparameters yielded the best results with lower dispersion as well as threshold values.

There is an overall presence of higher dispersion as well as threshold values when compared to the other metrics. This happens due to the high step size when changing the TT values in each simulation in respect to the maximum possible value. This leads the agent to struggle to stay within the 2% threshold limit. Next section will discuss the MP metrics thresholds.

A.1.8 Box Plots for Main Path Metrics Thresholds

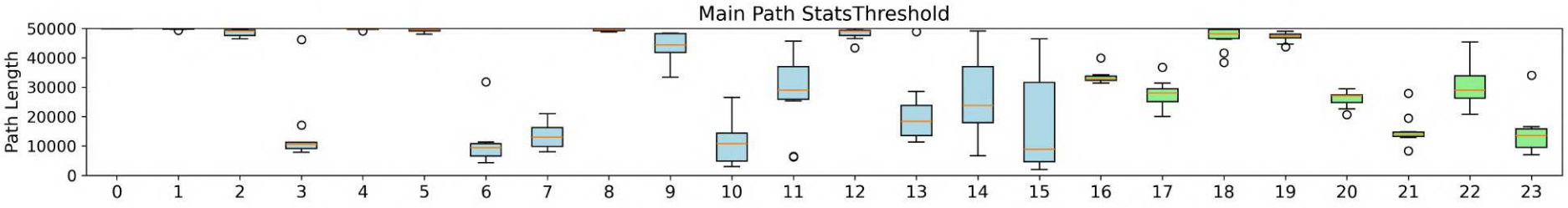
Box plots were used in order to have insights about the metric thresholds distribution over the 10 repeated simulations. These thresholds represent how fast each simulation reached and stayed in a region of $\pm 2\%$ of the best metric value. The final value regarding MP for each metric as well as the number of episodes to reach the pre determined performance threshold for each of them are shown in Figures 71-73. The indexes in the x axis in the plots are numerical values that represent the combination of ϵ and α values used for that simulation. These indexes can be found in Table 14. They were grouped in four groups, A, B, C and D, respectively, which will be plotted, in this order, with the following colors: light blue, light green, light yellow and light coral, akin to how it was displayed in the previous section.

A.1.8.1 Main Path Path Length metric threshold

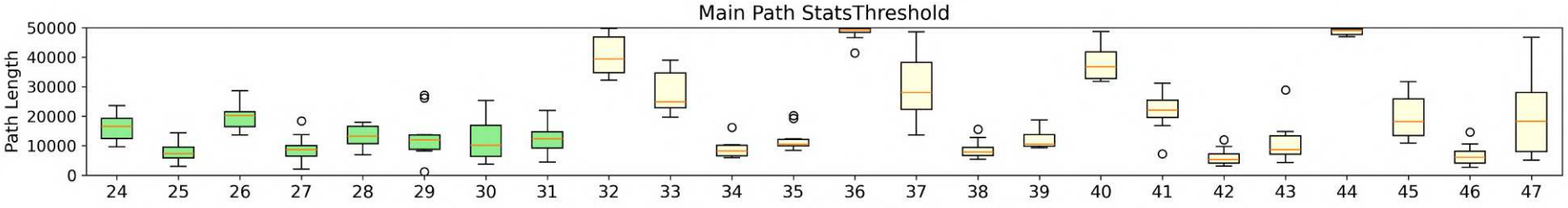
Figure 71 depicts the Path Length metric threshold computed for the Main Path, where the indexes and groups are detailed in Table 14.

MP PL Threshold - Group A - Constant ϵ and α values Figure 71 (a) shows the simulations results of Group A in Idx_{0-15}. $Idx_{6,7}$ were the only combinations with low threshold values, outliers and dispersion. This means that ϵ values higher or lower than 0.7 either offer too much exploration leading to an unstable learning path, or too low exploration not achieving enough variety in the agent's actions throughout training. Also this needs to be paired with lower learning rates to obtain consistent results.

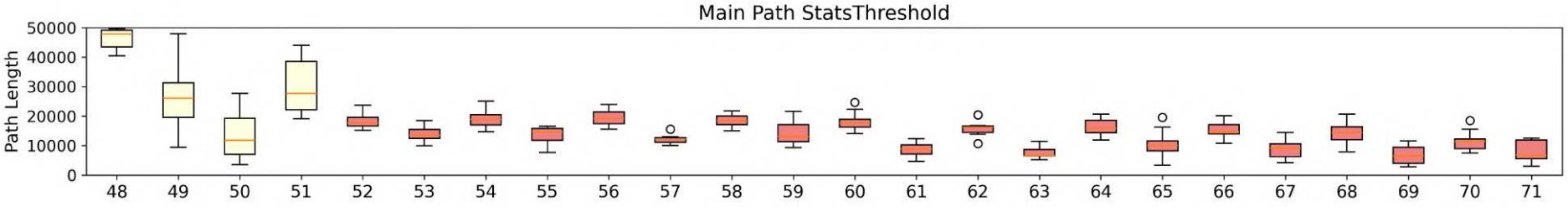
MP PL Threshold - Group B - Constant ϵ and variable α values Figures 71 (a) and (b) shows the simulations results of Group B in Idx_{16-31}. This group showed more stable results when using ϵ lower than 0.9, as it is possible to see a progressively lower



(a) Results for Groups A and B.



(b) Results for Groups B and C.



(c) Results for Groups C and D.

Figure 71 – Box Plot for Main Path Path Length metric threshold average. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively.

dispersion as well as lower threshold values for hyperparamenter configurations in Idx_{20-31}.

MP PL Threshold - Group C - Variable ϵ and constant α values - Figures 71 (b) and (c) shows the simulations results of Group C in Idx_{32-51}. This group showed more consistent results for lower values of α such as 0.1 or 0.05. $Idx_{34,35,38,39,42,43}$ are good examples of this behavior.

MP PL Threshold - Group D - Variable ϵ and α values Figure 71 (c) shows the simulations results of Group D in Idx_{52-71}. Using variable values for both hyperparameters lead to overall more consistent results, as seen in this group's results.

Different than it was seen in when analyzing the AP and MP metric values, there is some observable dispersion when looking at the metric's convergence threshold values, even though optimizing for the Main Path is an easier task than for All Paths. As seen in some of the other analysis, group D shows better performance overall. In the next section, the MP MOD metric thresholds will be discussed.

A.1.8.2 Main Path Mean Obstacle Distance metric threshold

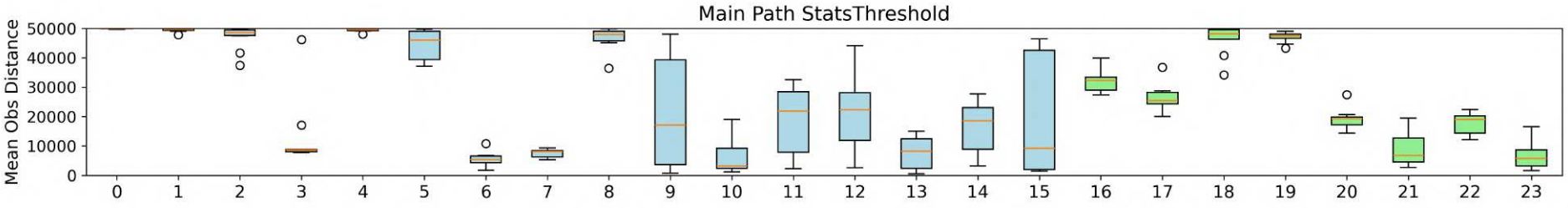
Figure 72 depicts the Mean Obstacle Distance metric threshold computed for the Main Path, where the indexes and groups are detailed in Table 14.

MP MOD Threshold - Group A - Constant ϵ and α values Figure 72 (a) shows the simulations results of Group A in Idx_{0-15}. There is a similar pattern to the results from this group on the MP PL metric threshold, but with a substantial increase in dispersion such as seen in Idx_{9-15}

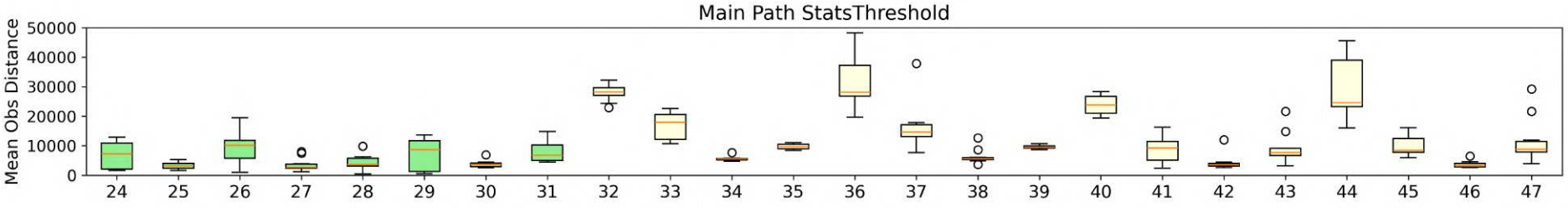
MP MOD Threshold - Group B - Constant ϵ and variable α values Figures 72 (a) and (b) shows the simulations results of Group B in Idx_{16-31}. This group showed more stable results when using ϵ lower than 0.9, as well. It is possible to see a progressively lower dispersion and lower threshold values for hyperparamenter configurations in Idx_{20-31}.

MP MOD Threshold - Group C - Variable ϵ and constant α values - Figures 72 (b) and (c) shows the simulations results of Group C in Idx_{32-51}. This group also follows a similar pattern to the MP PL metric threshold but with increased dispersion across all hyperparameter combinations.

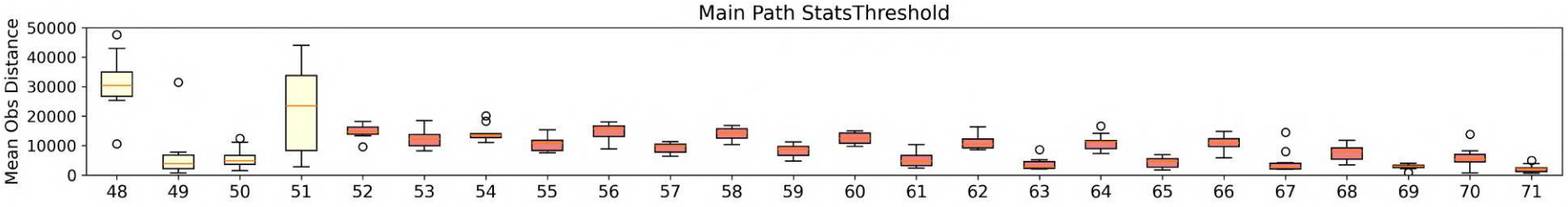
MP MOD Threshold - Group D - Variable ϵ and α values Figure 72 (c) shows the simulations results of Group D in Idx_{52-71}. Using variable values for both hyperparameters lead to overall more consistent results, as seen in this group's results.



(a) Results for Groups A and B.



(b) Results for Groups B and C.



(c) Results for Groups C and D.

Figure 72 – Box Plot for Main Path Mean Obstacle Distance metric threshold average. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively.

This metric's threshold values and dispersion followed a similar behavior as the one seen in the MP PL metric threshold overall but with an increased dispersion between the results from each simulation. This can arise from the fact that when modifying the main path, there is a much smaller shift in the Path Length, than there is in the Mean Obstacle Distance, as it is calculated using the ten closest obstacles. Next section, the MP Total Turns metric thresholds will be discussed.

A.1.8.3 All Paths Total Turns metric threshold

Figure 73 depicts the Total Turns metric threshold computed for the Main Path, where the indexes and groups are detailed in Table 14.

MP TT Threshold - Group A - Constant ϵ and α values Figure 73 (a) shows the simulations results of Group A in Idx_{0-15}. There is a similar pattern to the results from this group on the MP PL metric threshold, but slighly less dispersion as seen in Idx_{5-15}

MP TT Threshold - Group B - Constant ϵ and variable α values Figures 73 (a) and (b) shows the simulations results of Group B in Idx_{16-31}. There is an notable decrease in dispersion as well as threshold values for ϵ of 0.7 and 0.5, as seen in the previous two metrics in Idx_{20-27}, but with an increase when using $\epsilon = 0.3$ as seen in Ix_{28-31}

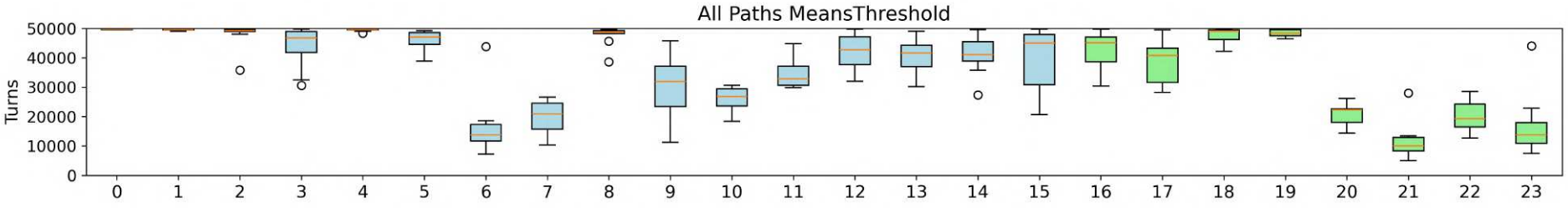
MP TT Threshold - Group C - Variable ϵ and constant α values - Figures 73 (b) and (c) shows the simulations results of Group C in Idx_{32-51}. Using variable ϵ values showed an increase in dispersion when compared to the previous two metrics.

MP TT Threshold - Group D - Variable ϵ and α values Figure 73 (c) shows the simulations results of Group D in Idx_{52-71}. Even though this group showed overall better results, in the final hyperparameter combinations.

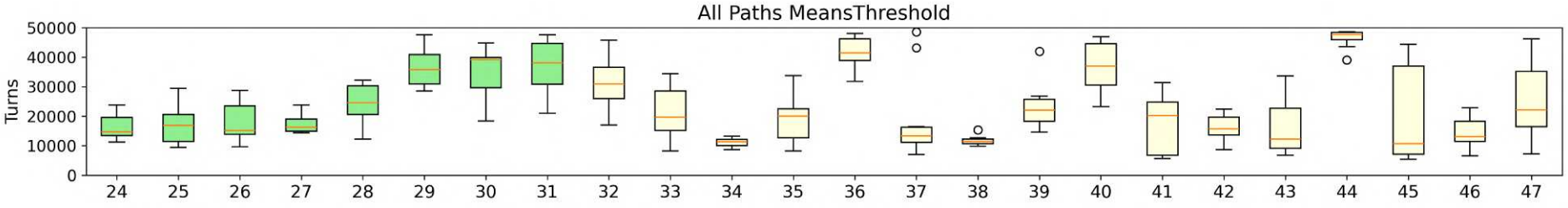
Even though the variation step size for this metric represents a higher portion of the maximum possible value found throughout the episodes, it is possible to see less dispersion overall when compared to the MOD metric. This may arise to having less possible TT configuration in the viable and efficient paths learned in the training processes.

A.2 Comparison of Epsilon and Exploring Starts Hyperparameters

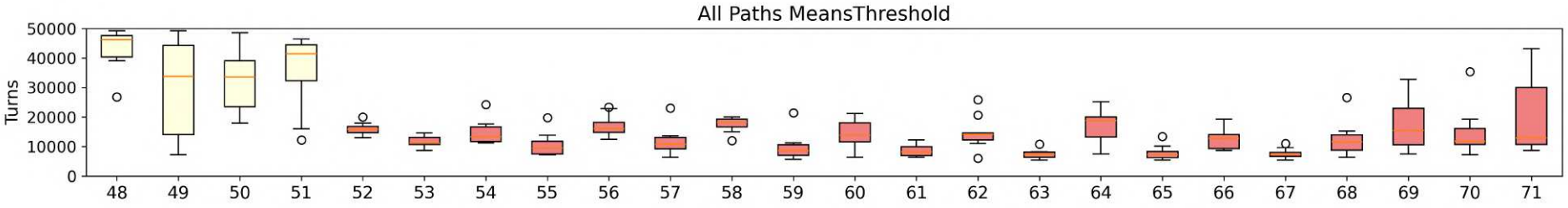
This section provides an examination of the impact of varying initial and terminal values for the hyperparameter Epsilon as well as the constant involving the Exploring Starts strategy. These parameters regulate the agent's likelihood to take random actions



(a) Results for Groups A and B.



(b) Results for Groups B and C.



(c) Results for Groups C and D.

Figure 73 – Box Plot for Main Path Total Turns metric threshold average. The colors light blue, light green, light yellow and light coral represent the groups A, B, C and D, respectively.

over the best-known action and the frequency in which the agent will start a new episode in a random new cell, respectively. Both of them deal with exploration strategies throughout training. The Epsilon value will affect the exploration factor during decision making, while the Exploring Starts constant will influence the exploration by changing the starting conditions with higher or lower frequency (lower or higher constant values). Each simulation was conducted iteratively for 10 repetitions.

For this analysis, the same map used in the previous section represented in Figure 74 was used as the learning environment, where the green cell is the origin point and the blue cell is the destination point. It is worth highlighting that during training, the agent may start in cells different from this origin point, due to the exploring starts strategy. In this Figure, the blue line represents the path taken by the agent and its length corresponds to the Path Length metric. Each time the agent takes a turn in its pathing, it adds up to the Total Turns metric, represented by orange dashed circles. Finally, to address safety concerns, the euclidean distance to the N closests obstacles for each cell of the agent's path is calculated and this is exemplified by the red dashed lines in the image with Obs_{d_i} being the distance to the i-th obstacle. For further reference, whenever the text refers to the "Main Path", it means the path that connects pre determined origin to the destination point and whenever the text refers to "All Paths", it means the group of paths originating from every cell of the map reaching the final destination point, also including the Main Path.

A.2.1 Performance Metrics and Representations

In conducting a performance analysis, it is imperative to not only determine the hyperparameters to be manipulated and the manner of their variation, but also the performance metrics for analysis and subsequent data visualization. Each of these components is described below.

- **Values of Epsilon and Exploring Starts constant** - 45 combinations involving constant values for both hyperparameters and diverse linear decay ranges for Epsilon were employed for evaluation.
  - **Constant Values** - *epsilon*: 0.9, 0.7, 0.5, 0.3 - *Exploring Starts*: 1, 10, 100, 1000, 10000;
  - **Decaying Range Upper and Lower Limits** - *epsilon*: (1, 0.1), (1,0.3), (0.8,0.1), (0.8,0.3), (0.6,0.3);
- **Data Visualization** - Three distinct plotting techniques were utilized to emphasize various aspects of the results, as described below.

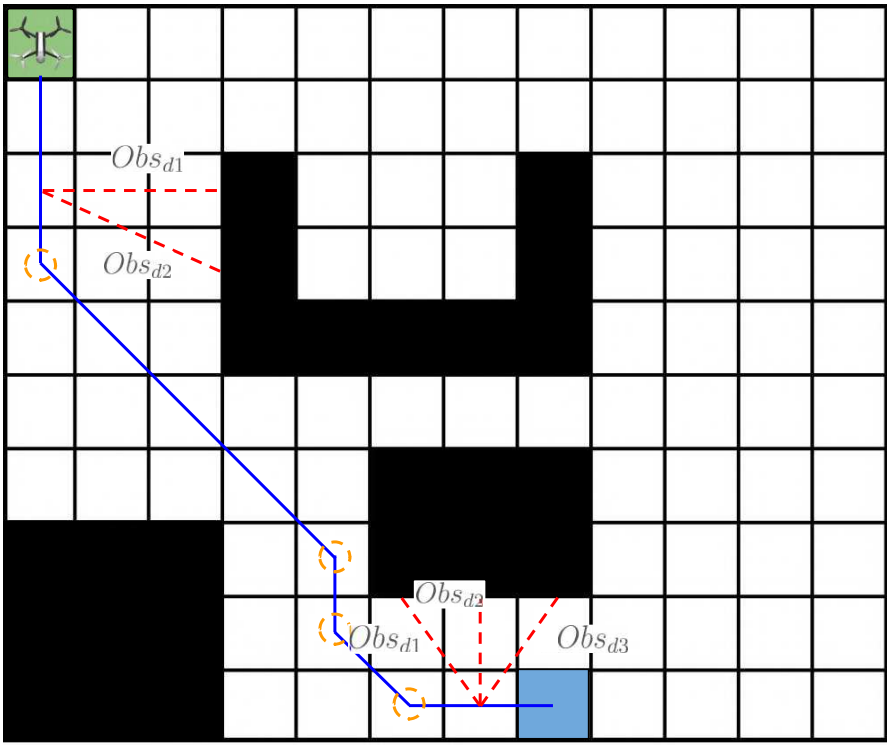


Figure 74 – Map used for each simulation in this section. Green and blue cells represent the starting and ending point respectively. Red dashed lines represent the euclidean distance to occupied cells, orange dashed circles represent each turn.

- **Parallel Coordinates Plots** - These plots enable the grouping of all simulation metrics into fewer visualizations, allowing for an overall view of simulation performances and the identification of trends and final values in hyperparameter sets. The metrics employed in these plots were the final value averaged over the 10 repeated simulations for each set. When depicting the number of episodes taken to reach the pre determined threshold, this word was abbreviated to "T".
- **Regular 3D Plots** - These plots were utilized to observe the progression of each of the Final Performance Values for each of the three priorities throughout training. This analysis will discuss aspects such as over and undershoots during training, as well as the plot form and convergence time for each metric. The graphics were categorized into four groups to aggregate similar hyperparameter sets: constant ϵ and α values, constant ϵ and variable α, variable ϵ and constant α, variable ϵ and α.
- **Box Plots** - These were employed to evaluate the numerical distribution across

the 10 repeated simulations for each set of hyperparameters. This analysis will discuss aspects such as final overall value as well as dispersion over the simulations. Given the volume of data, these results were presented across various subfigures for ease of viewing. In these plots, distribution of the final values as well as the thresholds are exhibited.

- **Performance Metrics** - Metrics related to two distinct aspects of our study were chosen to assess simulation performance: Main Path (MP), given the significance of establishing an optimal path for the agent, and All Paths Means (AP), considering that this algorithm emphasizes not only generating a viable path utilizing designated priorities from the starting point to the goal, but also producing quality paths from any location on the map. The following metrics are employed for both aspects and can be categorized into two groups, resulting in a total of 12 components, as outlined below.
  - **Final Performance Values** - These assess the values attained at the conclusion of training for each of the three priorities: Path Length, Mean Obstacle Distance, and Total Turns, which will henceforth be referred by the acronyms PL, MOD and TT. These attributes evaluate the overall final performance for each set of hyperparameters.
  - **Performance Threshold** - These evaluate how quickly the agent was able to achieve and maintain performance within a certain threshold of the training's best performance for each of the three priorities. This evaluates not the absolute value, but the speed at which the specific hyperparameter set approached a performance level close to that obtained after all episodes. The threshold range employed was $\pm 2\%$ of the final value.

Finally, all tests were done using the same $\alpha = 0.2$ and Reward Constants of -0.6, -0.1, -0.6 and 100 for the Safety, Step, Energy and Goal Rewards, respectively.

A.2.2 Parallel Coordinates Plots for All Paths

Parallel Coordinate Plots were employed as a visualization tool to comprehensively present the metrics and thresholds as well as more easily visualize trends throughout the hyperparameters configurations. In this type of plot, each line represents one simulation. In Figure 75, the three columns inside the green box represent the hyperparameters used in each simulation and the six columns inside the red box represent the evaluated metrics for each of them. As an example to indicate how to read this plot, the highlighted green line denotes initial and final ϵ values (In. Eps and Final Eps) of 0.9, along with Exploring Starts value (Exp. Strt) of 1. Traversing the line across all columns allows access to the final metrics values averaged across the 10 repetitions of each simulation. The color bar

within Figures 75 and 76 corresponds to the index of each simulation, as referenced in Table 15. Their ordering follows the permutation of the Exploring Starts value with the constant values of ϵ, then with the variable values for ϵ, which were grouped in two groups A and B respectively with the following colors: light blue and light green. For further reference, when referring to the simulation indexes, the notation Idx_n will refer to the the index n, Idx_{n-m} will refer to indexes from n to m and $Idx_{x,y,z}$ will refer to indexes x, y and z.

Final Performance Metrics In Figure 75, the metrics obtained from the simulations are depicted. It is possible to see that, regarding the MP metrics, most paths reached the 19.8 m length, and the alternative paths came with lower PL at the expense of safety with a lower MOD as well. As for AP metrics, there are more red lines on higher AP MOD values, indicating that a balancing between higher exploration in early stages and more exploitation on the later ones is important to maximize safety. Overall lower AP PL and AP Turns are found on lower indexes (green lines), which indicate that higher constant exploration rates can lead to shorter paths and energy efficient paths at the expense of safety.

Performance Thresholds Regarding the thresholds for each metric, faster convergence was observed for AP PL using higher indexes, indicating that variable ϵ values are better in this matter. When it comes to AP MOD, the hyperparameter combinations had overall low threshold limits, with some outliers on the lower index range needing more episodes to converge. As for MP MOD, there is more diversity in convergence speed when compared to AP MOD. Both AP and MP Turn thresholds are more concentrated on the mid to high episode count, which is expected given that shifting for one extra or one less Total Turns can represent a substantial percentage shift, most likely leading the Total Turns metric outside of the %2 threshold limit.

A.2.3 3D Plots for All Paths

Regular 3D plots will make way for an in-depth analysis of the evolution of each metric throughout training. Figures 77 through **??** illustrate the evolution of these metrics concerning the values of ϵ and Exploring Starts throughout the training regimen for All Paths. It is noteworthy that for certain hyperparameter configurations, the values of ϵ decay over time, with this variation serving as an indicator of episode progression. The presented values in these figures result from the averaging of metrics derived from 10 repetitions conducted for each simulation. For ease of visualization, each of the 3D plots have different color tones for each value of the Exploring Starts constant, with 1 having shades of red, 10 of green, 100 of blue, 1000 of yellow and 10 000 of purple. To mitigate

Table 15 – ϵ and Exploring Starts (Exp. Strt) Permutation Indexes. Group A has constant values for ϵ and B constant values for ϵ.

A	Index	**0**	**1**	**2**	**3**	**4**
	$\epsilon/Exp.Strt$	0.9, 1	0.9, 10	0.9, 100	0.9, 1 000	0.9, 10 000
	Index	**5**	**6**	**7**	**8**	**9**
	$\epsilon/Exp.Strt$	0.7, 1	0.7, 10	0.7, 100	0.7, 1 000	0.7, 10 000
	Index	**10**	**11**	**12**	**13**	**14**
	$\epsilon/Exp.Strt$	0.5, 1	0.5, 10	0.5, 100	0.5, 1 000	0.5, 10 000
	Index	**15**	**16**	**17**	**18**	**19**
	$\epsilon/Exp.Strt$	0.3, 1	0.3, 10	0.3, 100	0.3, 1 000	0.3, 10 000
B	Index	**20**	**21**	**22**	**23**	**24**
	$\epsilon/Exp.Strt$	[0.1, 1], 1	[0.1, 1], 10	[0.1, 1], 100	[0.1, 1], 1 000	[0.1, 1], 10 000
	Index	**25**	**26**	**27**	**28**	**29**
	$\epsilon/Exp.Strt$	[0.3, 1], 1	[0.3, 1], 10	[0.3, 1], 100	[0.3, 1], 1 000	[0.3, 1], 10 000
	Index	**30**	**31**	**32**	**33**	**34**
	$\epsilon/Exp.Strt$	[0.1, 0.8], 1	[0.1, 0.8], 10	[0.1, 0.8], 100	[0.1, 0.8], 1 000	[0.1, 0.8], 10 000
	Index	**35**	**36**	**37**	**38**	**39**
	$\epsilon/Exp.Strt$	[0.3, 0.8], 1	[0.3, 0.8], 10	[0.3, 0.8], 100	[0.3, 0.8], 1 000	[0.3, 0.8], 10 000
	Index	**40**	**41**	**42**	**43**	**44**
	$\epsilon/Exp.Strt$	[0.3, 0.6], 1	[0.3, 0.6], 10	[0.3, 0.6], 100	[0.3, 0.6], 1 000	[0.3, 0.6], 10 000

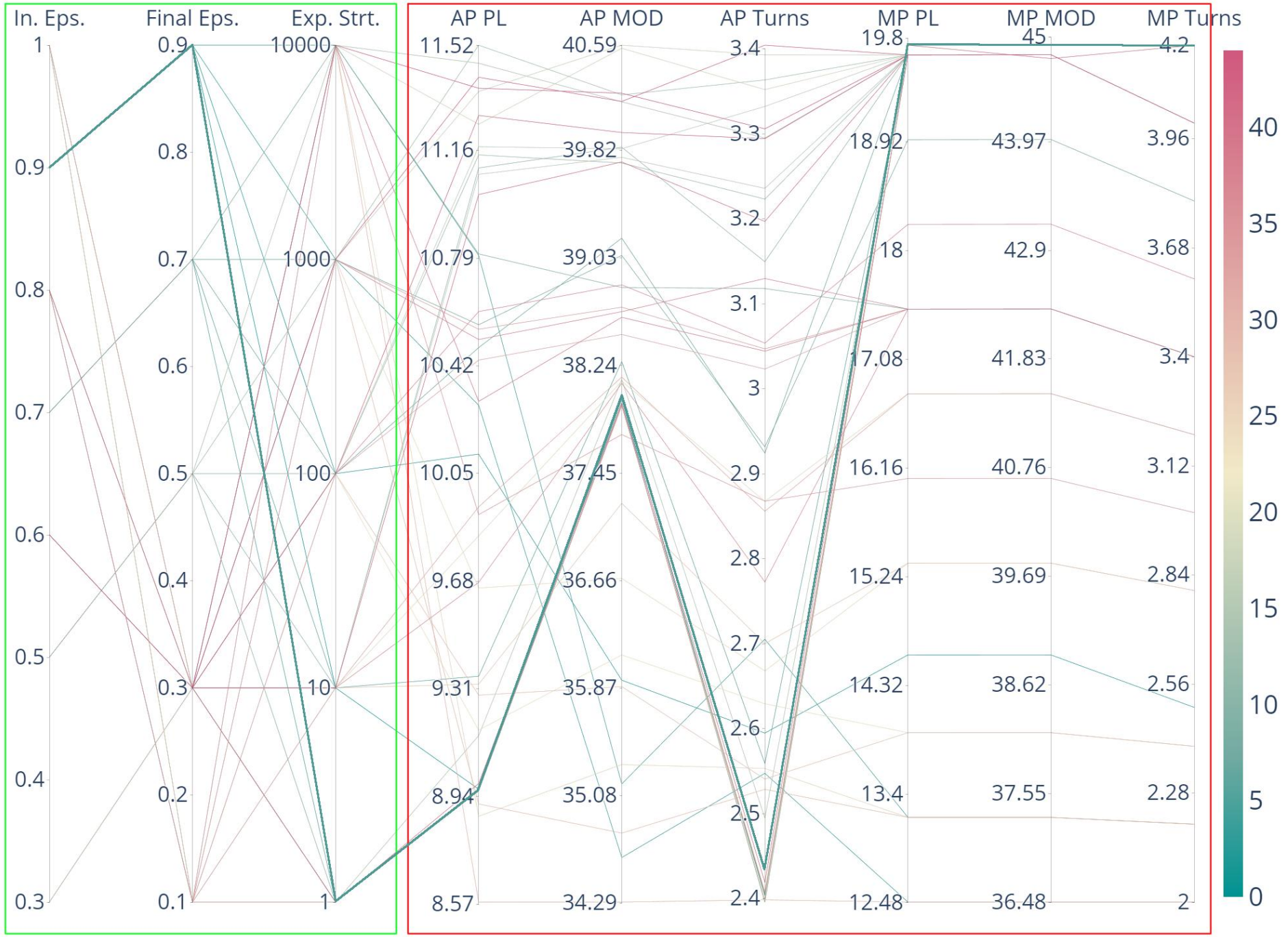


Figure 75 – Parallel Coordinates Plot for Metrics in all Simulations. Axis inside green box are the hyperparameters used, and axis inside red box are the evaluated metrics. Green highlighted line constant value of 0.9 for ϵ and 1 for Exploring Starts.

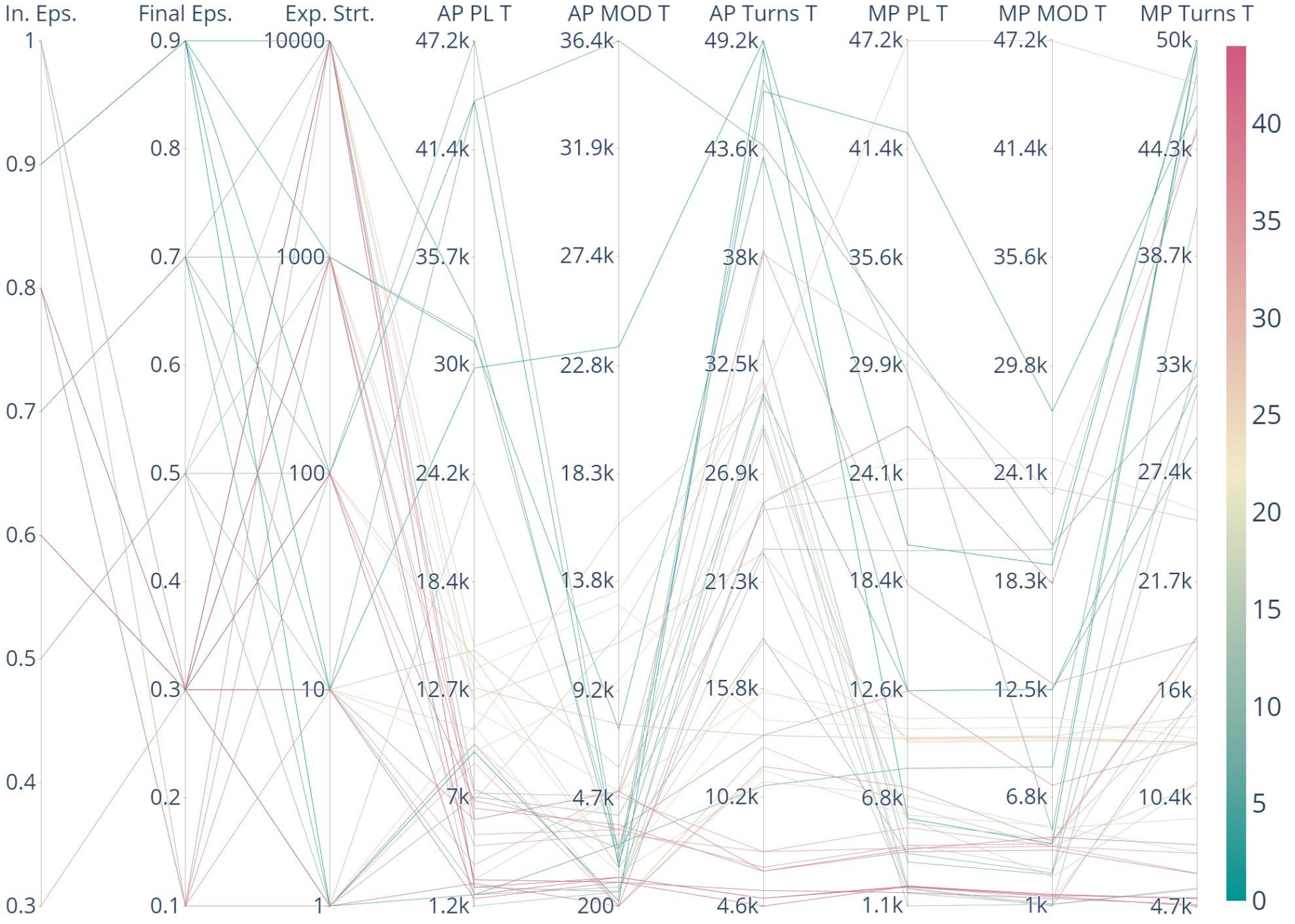
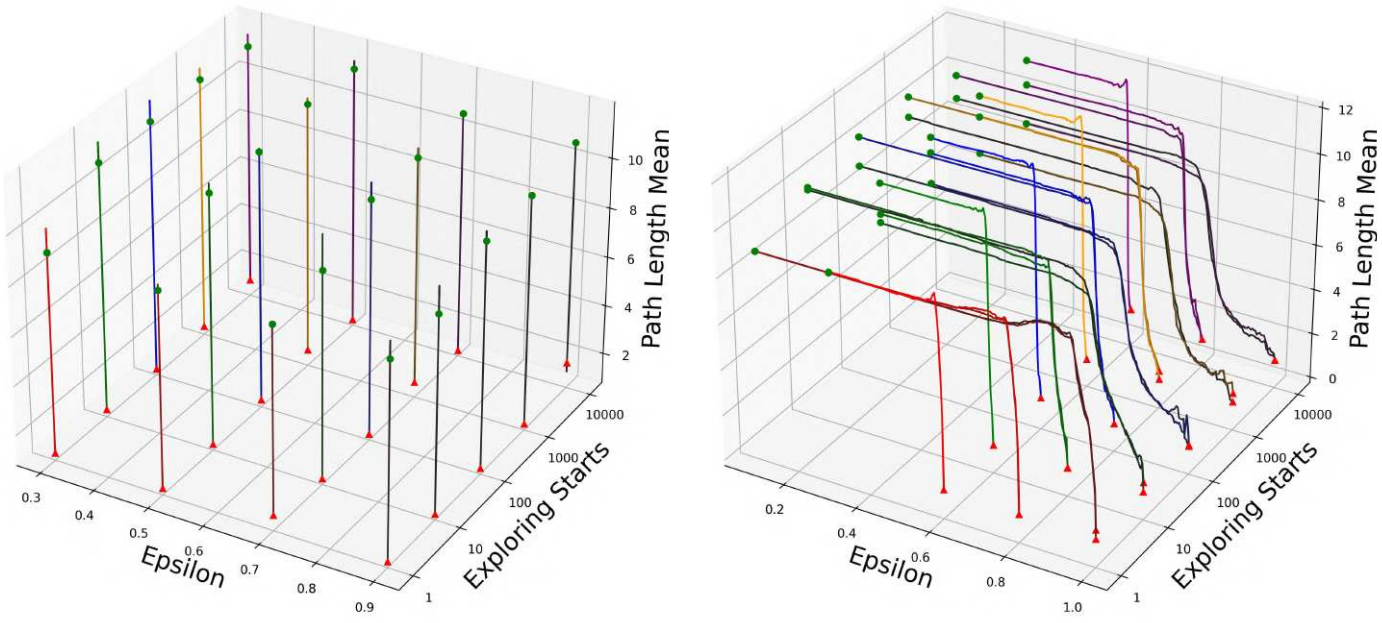


Figure 76 – Parallel Coordinates Plot for Metrics in all Simulations. Axis inside green box are the hyperparameters used, and axis inside red box are the evaluated metrics.



(a) Constant Epsilon values

(b) Variable Epsilon values.

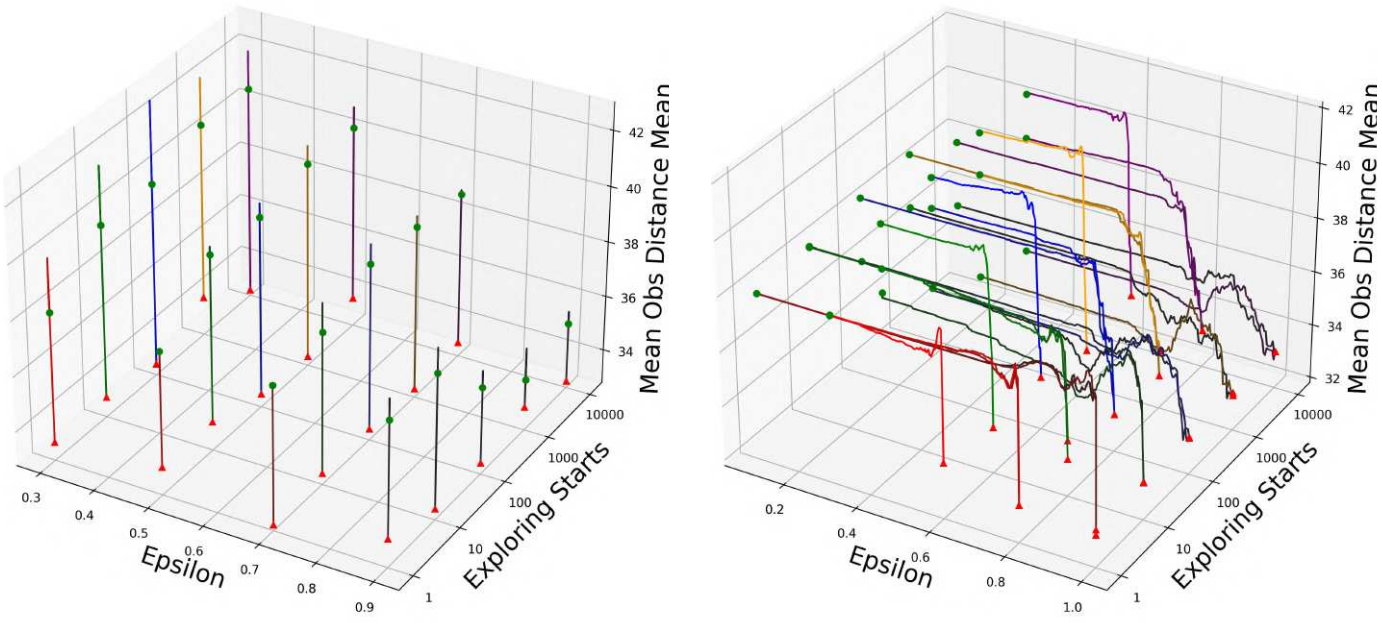
Figure 77 – All Paths Path Length average throughout training. Sub figures (a), (b) show the results for constant and variable ϵ values, respectively. Starting value is represented by a red triangle and final value by a green circle.

the inherent noise in the values, we applied a moving average window with a span of 100 episodes.

A.2.3.1 All Paths Path Length Metric Analysis

Figure 77 depicts the mean Path Length metric computed for paths associated with every cell in the map, where sub figures (a) and (b) show the results for constant ϵ variable ϵ values, respectively with Exploring Starts value remaining constant in all scenarios.

AP PL - Constant ϵ values - Figure 77 (a) shows the variation throughout training for constant ϵ values for the PL metric regarding AP. Given that the passing of time was represented by the decay of either of the hyper parameters, this graph will only provide the final value (red triangle) and final one (green circle). For the same ϵ value, it is observed that Exploring Starts of 10 and 100 tend to result in higher overshoots. This phenomenon can be attributed to the increased variety in the states encountered by the agent during training. However, this variety is not as extensive as with an Exploring Starts value of 1, which allows the agent to quickly find optimal solutions and thus reduces overshoot. In contrast, higher Exploring Starts may lack sufficient variability to produce significant overshoots. Regarding constant ϵ values, lower values lead to more frequent and pronounced overshoots, suggesting that lower constant exploration rates hinder the agent's ability to effectively optimize its



(a) Constant Epsilon values

(b) Variable Epsilon values.

Figure 78 – All Paths Mean Obstacle Distance average throughout training. Sub figures (a), (b) show the results for constant and variable ϵ values, respectively. Starting value is represented by a red triangle and final value by a green circle.

findings.

AP PL - Variable ϵ values Figure 77 (b) shows the variation throughout training for variable ϵ values for the PL metric regarding AP. For higher values of Exploring Starts there is a longer build-up stage in early episodes, specially for higher starting ϵ values. Which suggests that with lower variability in the starting locations, the agent takes a longer time to find feasible paths from every cell. Higher overshoots are present for lower values of starting ϵ, which can be attributed to the agent having lower exploration rates, leading to longer time to optimize the paths found.

Overall, for the AP PL metric, it was observed that the extra exploration layer provided by the Exploring Starts metrics can substantially improve the metric evolution and even help prevent overshoots in some combination. In the next section it will be discussed the MOD metrics for AP.

A.2.3.2 All Paths Mean Obstacle Distance Metric Analysis

Figure 78 depicts the Mean Obstacle Distance metric computed for paths associated with every cell in the map, where sub figures (a) and (b) show the results for constant ϵ variable ϵ values, respectively with Exploring Starts value remaining constant in all scenarios.

AP PL - Constant ϵ values - Figure 78 (a) shows the variation throughout training for constant ϵ values for the MOD metric regarding AP. Given that the passing of time was represented by the decay of either of the hyper parameters, this graph will only provide the final value (red triangle) and final one (green circle). Higher ϵ values led to less safe paths, specially when using higher Exploring Starts values, given the lack of of exploitation required to refine a more complex priority such as MOD. For the lower end of ϵ values there is a higher overshoot presence, but overall safer paths, suggesting the lower exploration rates will take longer to optimize, but ended up with better final scores.

AP PL - Variable ϵ values Figure 78 (b) shows the variation throughout training for variable ϵ values for the MOD metric regarding AP. Higher initial ϵ values led to more disturbance in the metrics evolution throughout training, specially when paired with higher Exploring Starts Values, which indicates that the added exploration layer provided by the lower Exploring Starts values can lead to a more direct learning curve. When it comes to lower initial ϵ starting values, even though there is a low overshoot, the final results where overall better, leading to safer paths.

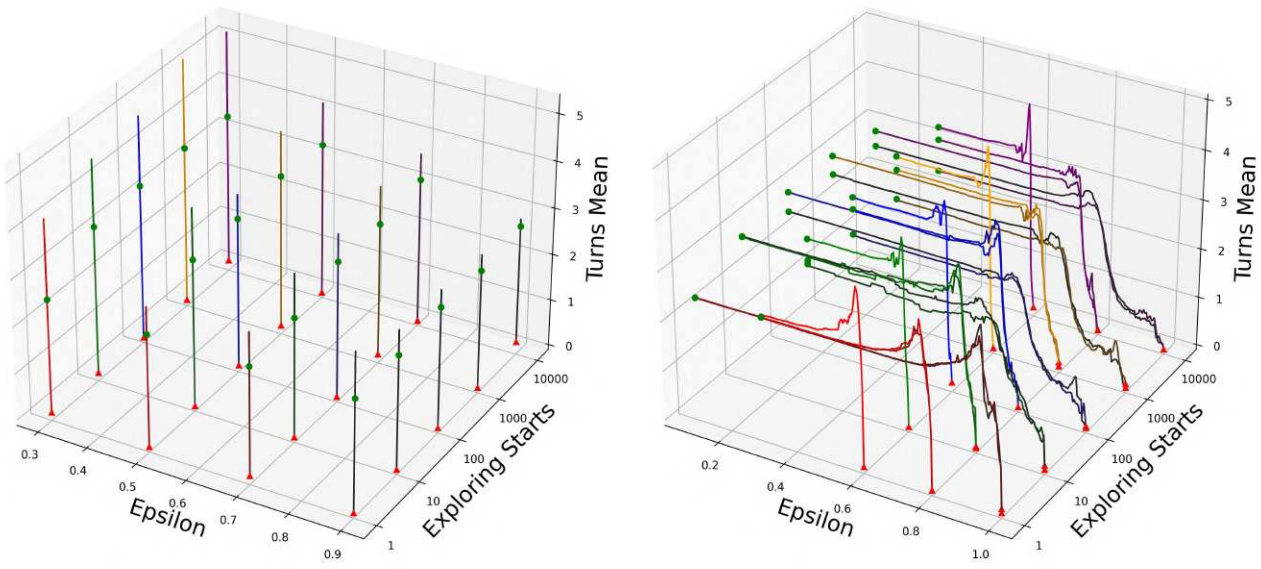
In the context of the AP MOD metric, it is clear that it is a more complex priority to optimize, therefore relying mostly on exploration is not enough to achieve the best results. A more careful balance between exploitation and exploitation is required for optimal results, as seen in the effects of varying ϵ and Exploring Starts values. In the next section the TT metric for AP will be dicussed

A.2.3.3 All Paths Total Turns Metric Analysis

Figure 79 depicts the Total Turns metric computed for paths associated with every cell in the map, where sub figures (a) and (b) show the results for constant ϵ variable ϵ values, respectively with Exploring Starts value remaining constant in all scenarios.

AP PL - Constant ϵ values - Figure 79 (a) shows the variation throughout training for constant ϵ values for the TT metric regarding AP. Given that the passing of time was represented by the decay of either of the hyper parameters, this graph will only provide the final value (red triangle) and final one (green circle). In a similar pattern as seen in the AP MOD metrics, it is possible to see increased TT values the lower the ϵ value, with overall more overshoots then seen in the previous the AP MOD metric.

AP PL - Variable ϵ values Figure 79 (b) shows the variation throughout training for variable ϵ values for the TT metric regarding AP. Higher Exploring Starts values lead to longer build up and settling time periods, as well as lower overshoots in most



(a) Constant Epsilon values

(b) Variable Epsilon values.

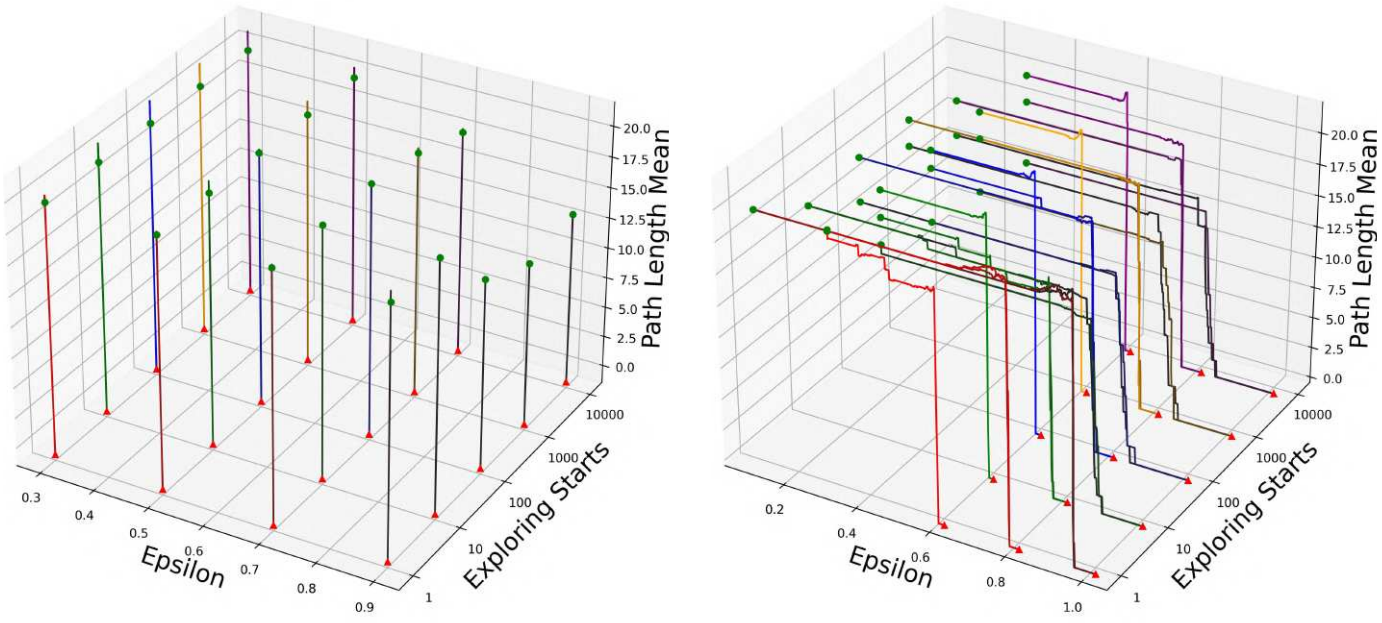
Figure 79 – All Paths Total Turns average throughout training. Sub figures (a), (b) show the results for constant and variable ϵ values, respectively. Starting value is represented by a red triangle and final value by a green circle.

cases. This suggests that the extra exploration layer of lower Exploring Starts values can lead to finding paths from all cells easier, but takes longer to optimize then and settle the overshoot in this metric.

Overall the AP TT metric had very similar behavior to the AP MOD, with a more noisy evolution when compared to the AP PL metric, as well as longer build up periods when using higher Exploring Starts values, but higher settling time as well when the fine tuning stage of training is reached.. The similarities are expected given that MOD and TT are more complex priorities than PL, with clearly more intricacies. In the next section all the metrics for the MP will be discussed.

A.2.4 3D Plots for Main Path

Figures 80 through 82 illustrate the evolution of these metrics concerning the values of ϵ and Exploring Starts throughout the training regimen for the Main Path. It is noteworthy that for certain hyperparameter configurations, the values of ϵ decay over time, with this variation serving as an indicator of episode progression. The presented values in these figures result from the averaging of metrics derived from 10 repetitions conducted for each simulation. For ease of visualization, each of the 3D plots have different color tones for each value of the Exploring Starts constant, with 1 having shades of red, 10 of green, 100 of blue, 1000 of yellow and 10 000 of purple. In subfigures (b), it is clear that the values initiate at zero and progressively increase. This behavior stems from the



(a) Constant Epsilon values

(b) Variable Epsilon values.

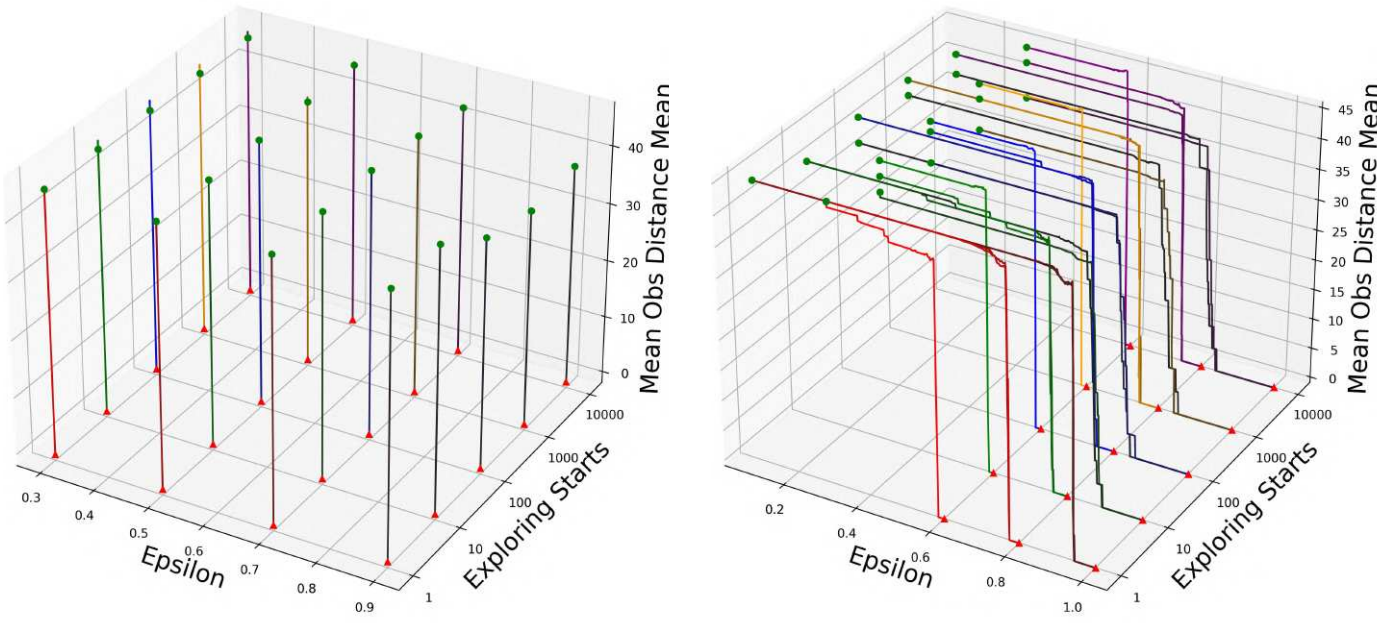
Figure 80 – Main Path Path Length average throughout training. Sub figures (a), (b) show the results for constant and variable ϵ values, respectively. Starting value is represented by a red triangle and final value by a green circle.

number of episodes required to obtain a viable path originating from the starting cell. The longer the length of this zero value phase in the training, the longer it took to find the first feasible path since during the initial stages of training, there is no clear path from the starting cell to the goal. To mitigate the inherent noise in the values, we applied a moving average window with a span of 100 episodes.

A.2.4.1 Main Path Path Length Metric Analysis

Figure 80 depicts the mean Path Length metric computed for the Main Path, where sub figures (a) and (b) show the results for constant ϵ variable ϵ values, respectively with Exploring Starts value remaining constant in all scenarios.

MP PL - Constant ϵ values - Figure 80 (a) shows the variation throughout training for constant ϵ values for the PL metric regarding MP. Given that the passing of time was represented by the decay of either of the hyper parameters, this graph will only provide the final value (red triangle) and final one (green circle). For the same ϵ value, it is observed that Exploring Starts of 10 and 100 tend to result in higher overshoots. Higher ϵ did not pair well with lower Exploring Starts, with the best results being obtained with less frequency on changing the starting cell, suggesting that doubling down on high exploration does not yield good results for this metric. Low intensity overshoots were present mostly on lower ϵ values.



(a) Constant Epsilon values

(b) Variable Epsilon values.

Figure 81 – Main Path Mean Obstacle Distance average throughout training. Sub figures (a), (b) show the results for constant and variable ϵ values, respectively. Starting value is represented by a red triangle and final value by a green circle.

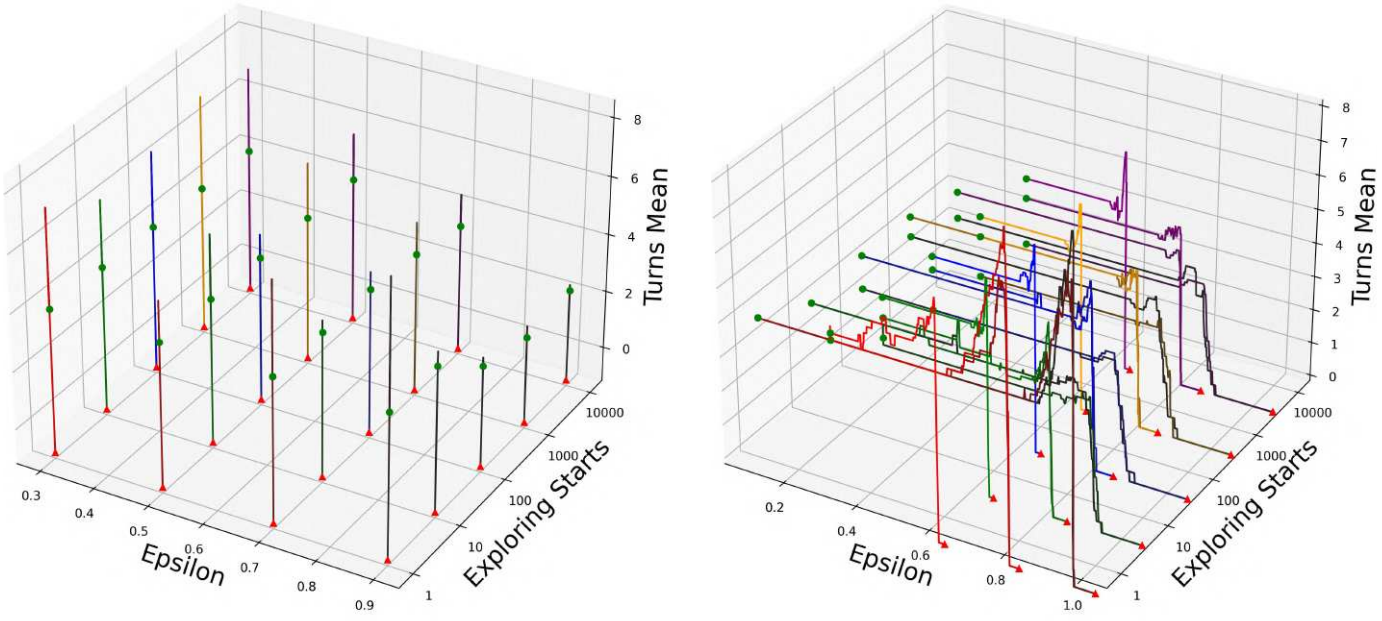
MP PL - Variable ϵ values Figure 80 (b) shows the variation throughout training for variable ϵ values for the PL metric regarding MP. The higher the starting ϵ value, the longer it took to find feasible paths, as seen in the length of the zero value period in this image. Lower starting ϵ values showed a step shape in some scenarios with also low Exploring Starts values, suggesting that the fast transition from exploration focused learning to exploitation focus, allied with high variability of starting positions can lead to local minima.

Overall, for the MP PL metric, it is possible to notice a trend that increased exploration rates from both hyperparameters can have diminishing returns, given its a simpler task to optimize the Main Path, when compared to optimizing paths from all cells in the grid. In the next section, the MP MOD metric will be discussed.

A.2.4.2 Main Path Mean Obstacle Distance Metric Analysis

Figure 81 depicts the Mean Obstacle Distance metric computed for the Main Path, where sub figures (a) and (b) show the results for constant ϵ variable ϵ values, respectively with Exploring Starts value remaining constant in all scenarios.

MP MOD - Constant ϵ values - Figure 81 (a) shows the variation throughout training for constant ϵ values for the MOD metric regarding MP. Given that the passing of time



(a) Constant Epsilon values

(b) Variable Epsilon values.

Figure 82 – Main Path Total Turns average throughout training. Sub figures (a), (b) show the results for constant and variable ϵ values, respectively. Starting value is represented by a red triangle and final value by a green circle.

was represented by the decay of either of the hyper parameters, this graph will only provide the final value (red triangle) and final one (green circle). Similar pattern as seen in MP PL with less present overshoots.

MP MOD - Variable ϵ values Figure 81 (b) shows the variation throughout training for variable ϵ values for the MOD metric regarding MP. The is a very similar behavior to what was seen in the MP PL metric, with the local minima being present in some configurations using lower starting ϵ combined with lower Exploring Starts values. Overall smaller overshoots were present, which can be explained due to the fact that this metric is calculated over the mean of the closest obstacles, attenuating the shift in its value.

It is possible to see that this metric has a similar pattern to the MP PL metric, with slightly less overshoots and the few ones that can be seen, were presented with less intensity. In the next section the MP TT metric will be discussed.

A.2.4.3 Main Path Total Turns Metric Analysis

Figure 82 depicts the Total Turns metric computed for the Main Path, where sub figures (a) and (b) show the results for constant ϵ variable ϵ values, respectively with Exploring Starts value remaining constant in all scenarios.

MP TT - Constant ϵ values - Figure 82 (a) shows the variation throughout training for constant ϵ values for the MOD metric regarding MP. Given that the passing of time was represented by the decay of either of the hyper parameters, this graph will only provide the final value (red triangle) and final one (green circle). Similar pattern as seen in MP PL with less present overshoots. The overshoots in this metric are substantially higher and more present then on the previous two. Also there is a higher difference between the best and worse result in this scenario. The best results were obtained using Exploring Starts values of 10 to 10 000 for most ϵ values.

MP TT - Variable ϵ values Figure 82 (b) shows the variation throughout training for variable ϵ values for the TT metric regarding MP. The metric's evolution for variable ϵ values shows a patter with higher oscillations when compared to the previous ones and more intense overshoots. Specially when it comes to lower Exploring Starts values, suggesting that the agent required more exploitation in order to better refine its path.

The Total Turns metric exhibits significantly more variation compared to the previous two metrics, likely due to the larger step size relative to the maximum value the metric can achieve. When comparing MP with AP results, we observe that convergence is generally faster and smoother for the main path, as anticipated. However, both PL and MOD metrics encounter local minima, a challenge not present in the AP metrics. In the next section, Box Plots will be displayed and discussed involving all metrics as well as the threshold values required to converge to its final value.

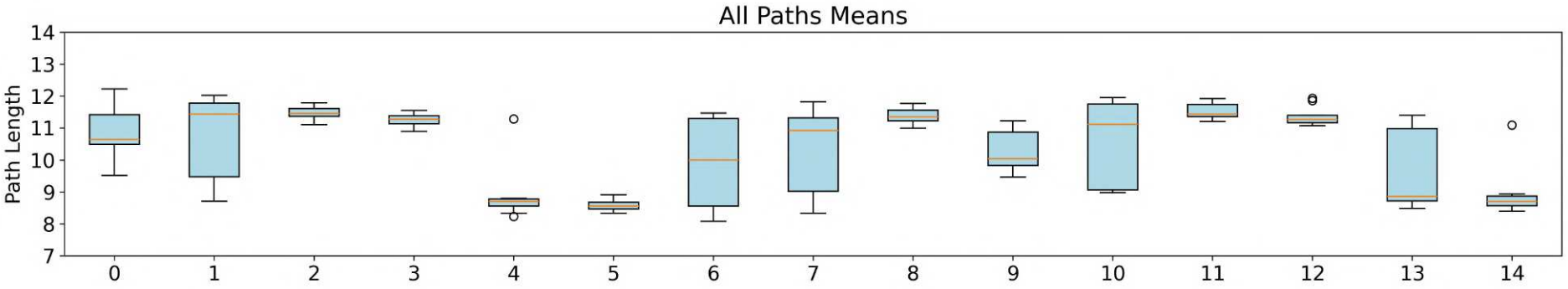
A.2.5 Box Plots for All Paths Metrics

Box plots were used in order to have insights about the metrics distribution over the 10 repeated simulations. The final value regarding AP for each metric as well as the number of episodes to reach the pre determined performance threshold for each of them are shown in Figures 83-85. The indexes in the x axis in the plots are numerical values that represent the combination of ϵ and Exploring Start values used for that simulation. These indexes can be found in Table 15.

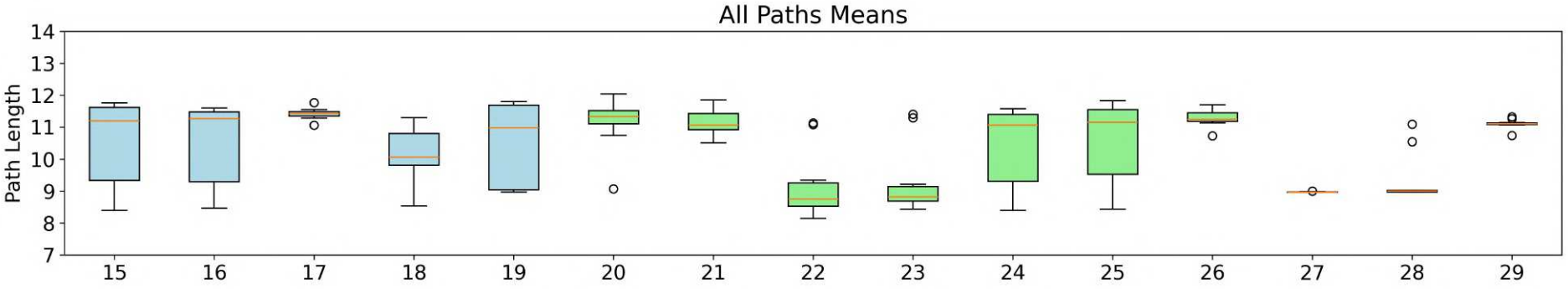
A.2.5.1 All Paths Path Length metric

Figure 83 depicts the Path Length metric computed for paths associated with every cell in the map, where the indexes and groups are detailed in Table 15.

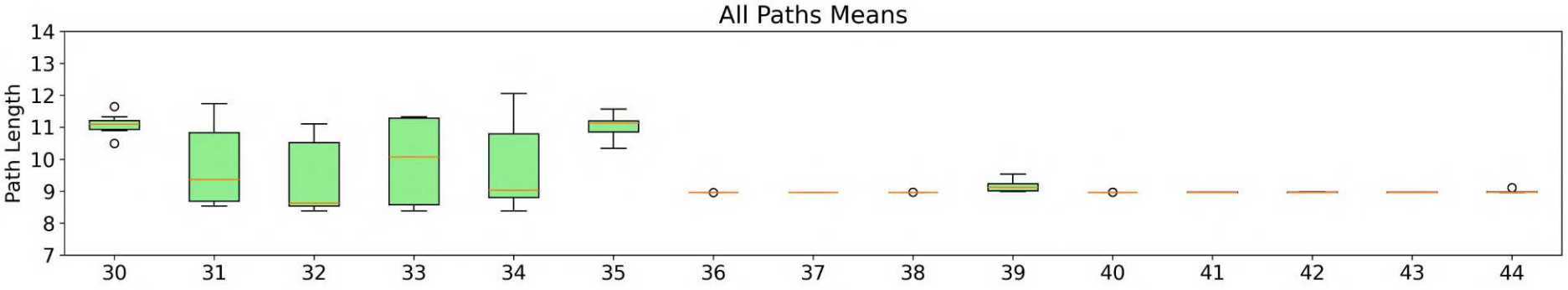
AP PL - Group A - Constant ϵ values Figure 83 (a) and (b) shows the simulations results of Group A in Idx_{0-19}. For $\epsilon = 0.9$ ($Idx_{0,5}$), there is more dispersion for lower Exploring Starts values, suggesting that with such high constant exploring ratio,



(a) Results for Group A.



(b) Results for Groups A and B.



(c) Results for Group B.

Figure 83 – Box Plot for All Paths Path Length average metric. The colors light blue and light green represent the groups A and B, respectively.

there is a greater need to start from the same cell in order to achieve consistent results. In contrast, for $\epsilon = 0.3$ ($Idx15 - 19$, there was high dispersion for all Exploring Starts values, except for 100 (Idx_17), suggesting that with such low exploration value, the variation degree on the starting cell is a sensitive parameter.

AP PL - Group B - variable ϵ values Figures 83 (b) and (c) shows the simulations results of Group B in Idx_{20-45}. The highest dispersion was found for $\epsilon \in [0.1, 08]$ (Idx_{30-34}), with the exception of Exploring Starts value of 1. When compared to the results of Idx_{35-44} which have the final value of ϵ as 0.3, it is possible to infer that, in order to achieve consistent results, some exploration on the later stages of training is required.

For the AP PL metric, Group A exhibited greater variability compared to Group B, suggesting that constant exploration rates may not be optimal for this metric. The most consistent results were observed in Idx_{35-44}, where the final ϵ values were 0.3, with initial values set to 0.8 and 0.6. This indicates that the AP PL metric is less responsive to the highest initial exploration rate of 1, and that maintaining some level of exploration in the later stages is necessary for achieving consistent results. In the next section the AP MOD metric will be discussed.

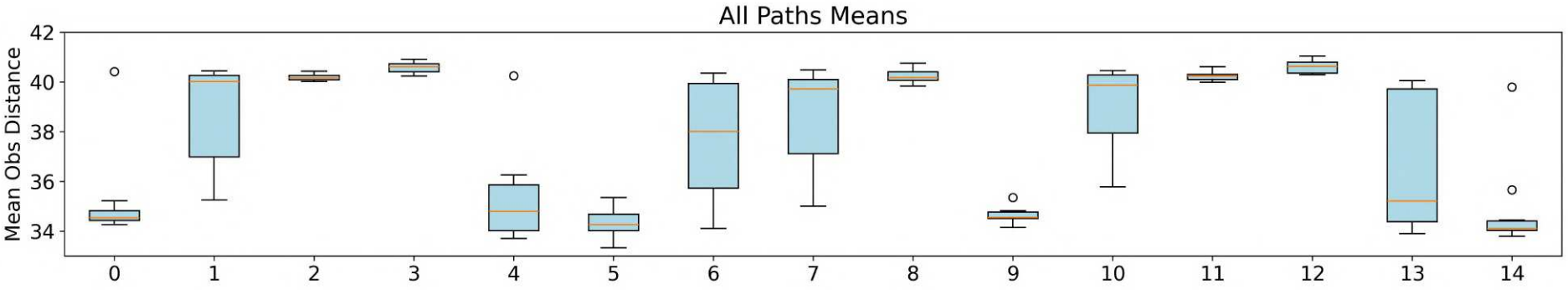
A.2.5.2 All Paths Mean Obstacle Distance metric

Figure 84 depicts the Mean Obstacle Distance metric computed for paths associated with every cell in the map, where the indexes and groups are detailed in Table 15.

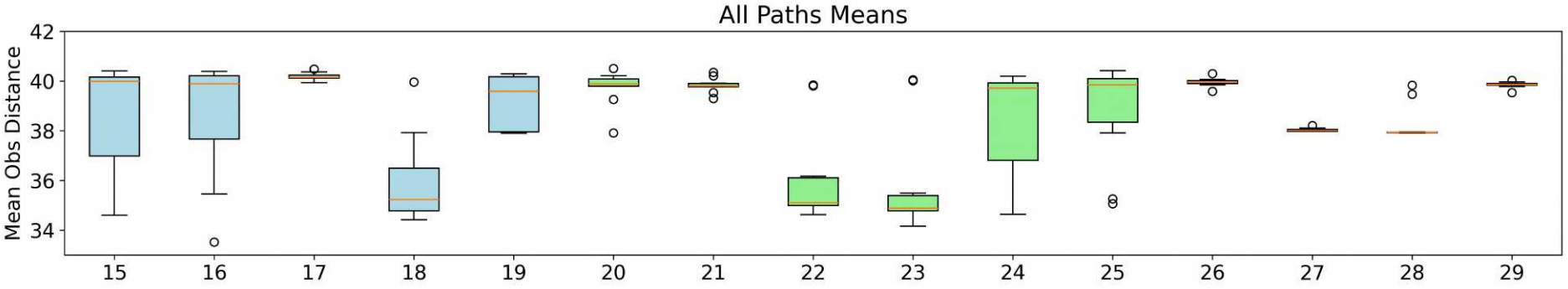
AP PL - Group A - Constant ϵ values Figure 84 (a) and (b) shows the simulations results of Group A within the Idx_{0-19} range. In this group, the dispersion in results increased as the ϵ value decreased. Notably, when $\epsilon = 0.3$, more hyperparameter combinations exhibited significant variation in outcomes compared to other ϵ values.

AP PL - Group B - variable ϵ values Figures 84 (b) and (c) shows the simulations results of Group B in Idx_{20-45}. A notable pattern emerges within this group, where the hyperparameter combinations with low variance predominantly feature a final ϵ value of 0.3 (Idx_{25-30} and Idx_{35-44}). This strongly suggests that maintaining a higher exploration rate during the later stages of training enhances consistency.

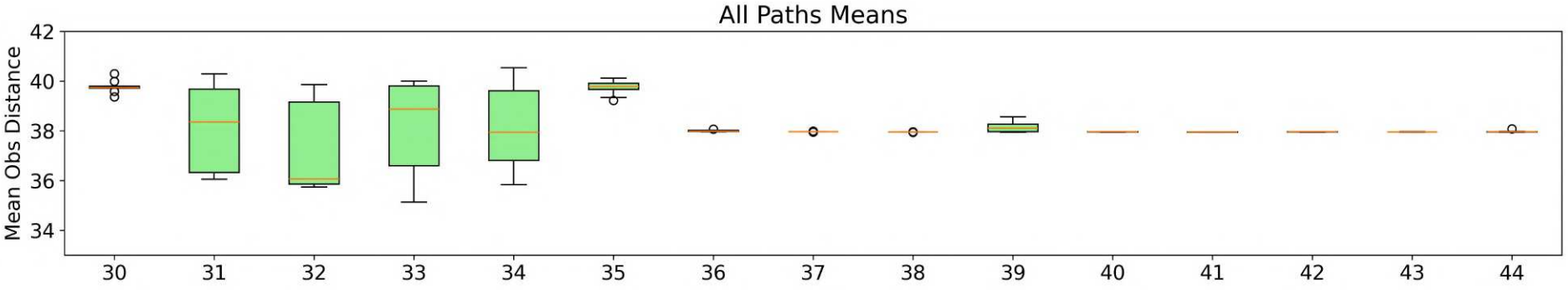
As seen in the previous metric, more consistent results are obtained with variable ϵ as opposed to constant ones, highlighting the relevance of a good exploration/exploitation balance in order to achieve better results. In the next section, the AP TT metric will be discussed.



(a) Results for Group A.



(b) Results for Groups A and B.



(c) Results for Group B.

Figure 84 – Box Plot for All Paths Mean Obstacle Distance average metric. The colors light blue and light green represent the groups A and B, respectively.

A.2.5.3 All Paths Total Turns Distance metric

Figure 85 depicts the Total Turns metric computed for paths associated with every cell in the map, where the indexes and groups are detailed in Table 15.

AP PL - Group A - Constant ϵ values Figure 85 (a) and (b) shows the simulations results of Group A within the Idx_{0-19} range. Compared to the other two metrics, PL and MOD, TT exhibited a wider dispersion across various hyperparameter combinations, albeit with less intensity. This may be due to the reduced number of viable path configurations, which results in greater variability in TT compared to the other metrics

AP PL - Group B - variable ϵ values Figures 85 (b) and (c) shows the simulations results of Group B in Idx_{20-45}. Following the trend observed in other metrics, the most consistent results were achieved using intervals where the final ϵ value was set to 0.3 (specifically in Idx_{25-30} and Idx_{35-44}). In contrast, the other combinations exhibited greater overall dispersion compared to the simulations in Group A.

As with the previous metrics, the AP TT metric demonstrates that variable ϵ values lead to more consistent outcomes compared to constant ones. Specifically, maintaining a final ϵ value of 0.3 appears to minimize variability, reinforcing the importance of a well-balanced exploration/exploitation strategy for achieving stable performance. In the next section, the MP metric will be discussed, starting with PL.

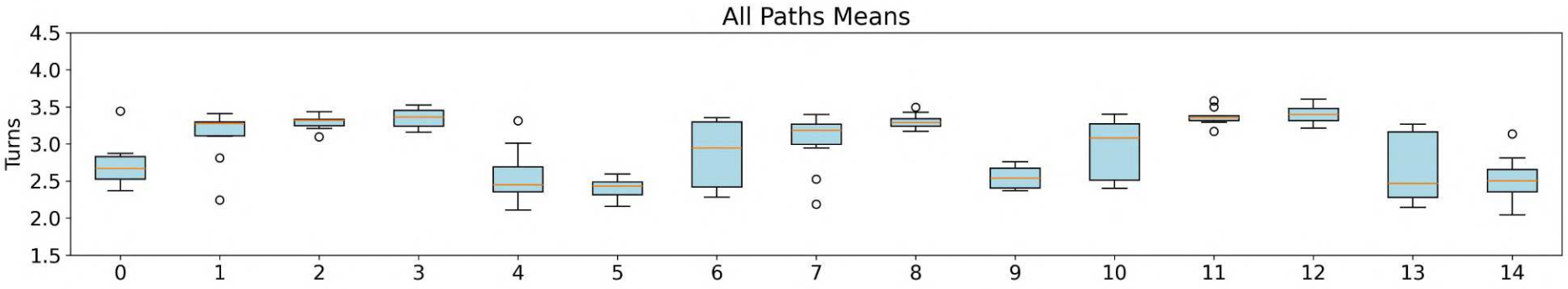
A.2.6 Box Plots for Main Path Metrics

Box plots were used in order to have insights about the metrics distribution over the 10 repeated simulations. The final value regarding MP for each metric as well as the number of episodes to reach the pre determined performance threshold for each of them are shown in Figures 86-88. The indexes in the x axis in the plots are numerical values that represent the combination of ϵ and Exploring Start values used for that simulation. These indexes can be found in Table 15.

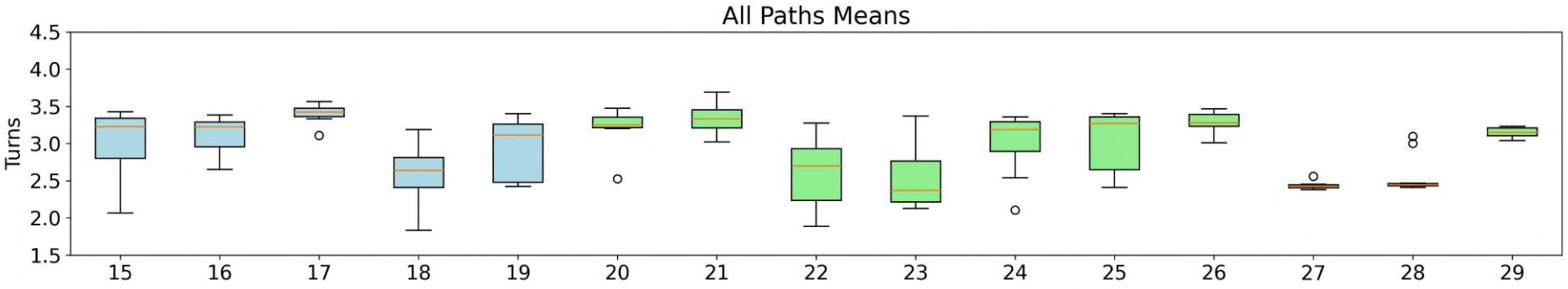
A.2.6.1 Main Path Path Length metric

Figure 86 depicts the Path Length metric computed for the Main Path, where the indexes and groups are detailed in Table 15.

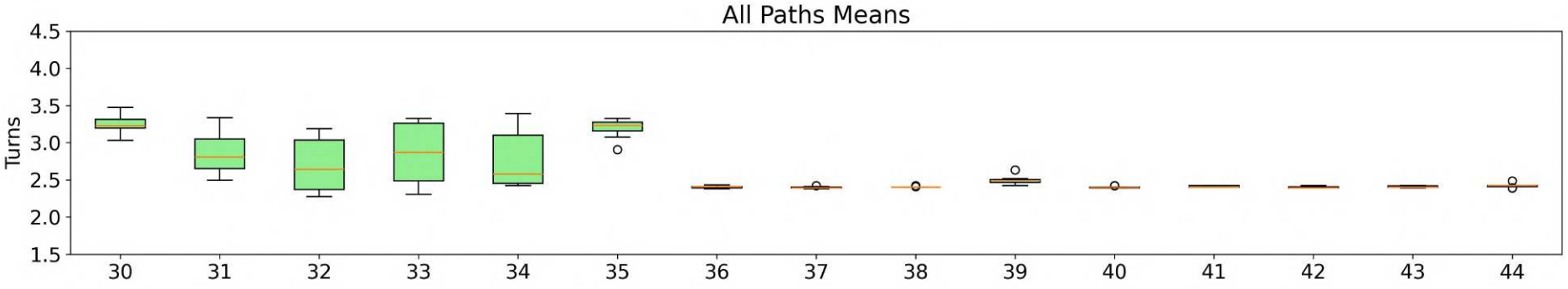
MP PL - Group A - Constant ϵ values Figure 86 (a) and (b) shows the simulations results of Group A in Idx_{0-19}. Accounting for all simulations, there is only two MP PL values found, either 12, or 20 meters, with the longest one been the safest, in which the agent will take the route around the U obstacle, instead of reaching the



(a) Results for Group A.

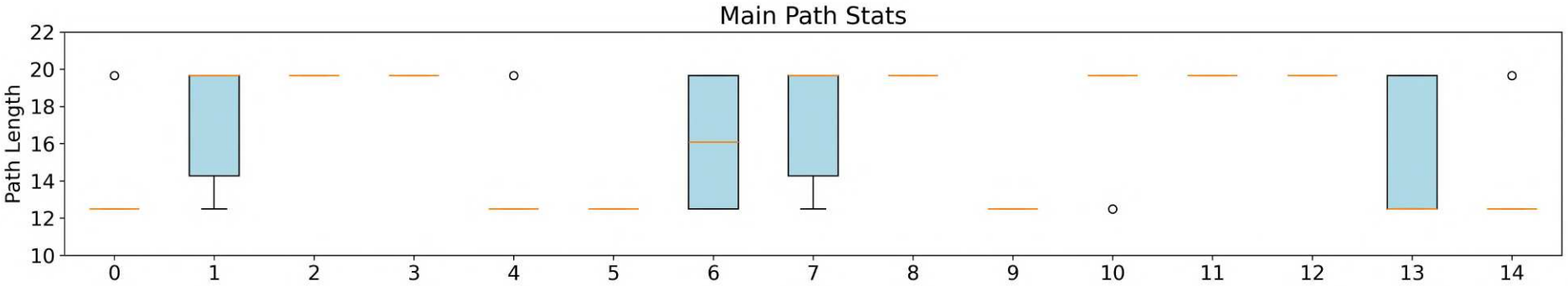


(b) Results for Groups A and B.

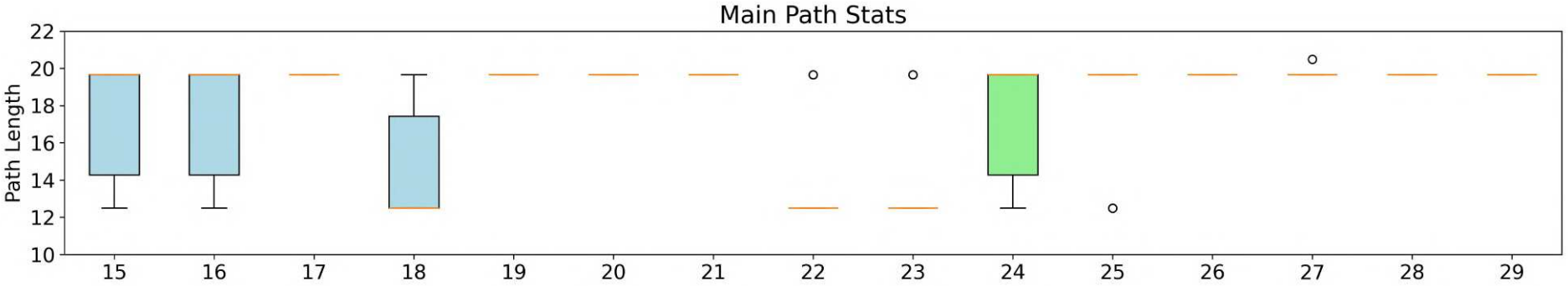


(c) Results for Group B.

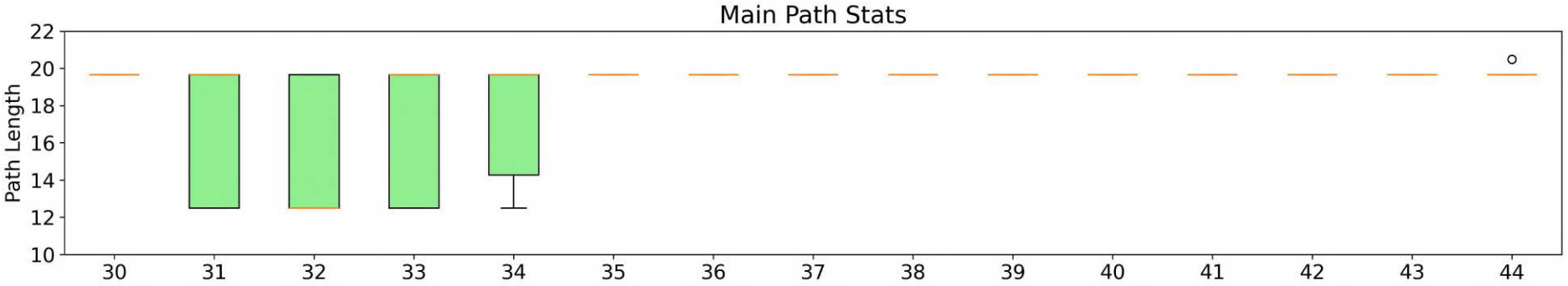
Figure 85 – Box Plot for All Paths Total Turns average metric. The colors light blue and light green represent the groups A and B, respectively.



(a) Results for Group A.



(b) Results for Groups A and B.



(c) Results for Group B.

Figure 86 – Box Plot for Main Path Path Length average metric. The colors light blue and light green represent the groups A and B, respectively.

goal navigating between the three obstacles as seen in Figure 74. Given that the rewards used for the simulations were intended to prioritize the three metrics, the ϵ value that most consistently found the safest route was 0.5 (Idx_{10-14}).

MP PL - Group B - variable ϵ values Figures 86 (b) and (c) shows the simulations results of Group B in Idx_{20-45}. In this Group, the safest route was found much more consistently than on the previous one, with the combinations that most struggled for this used $\epsilon \in [0.1, 0.8]$ (Idx_{30-34}).

The MP PL results emphasize the significance of a well-calibrated exploration-exploitation balance. In Group A, where constant ϵ values were used, the agent consistently identified two distinct path lengths, with the safest route being more frequently selected when $\epsilon = 0.5$. In contrast, Group B, with variable ϵ values, showed greater consistency in finding the safest route, particularly when avoiding the ϵ range of [0.1, 0.8]. These findings highlight that dynamically adjusting ϵ during training can lead to more reliable and safer navigation strategies. The next section will discuss the MP MOD metric.

A.2.6.2 Main Path Mean Obstacle Distance metric

Figure 87 depicts the Mean Obstacle Distance metric computed for the Main Path, where the indexes and groups are detailed in Table 15.

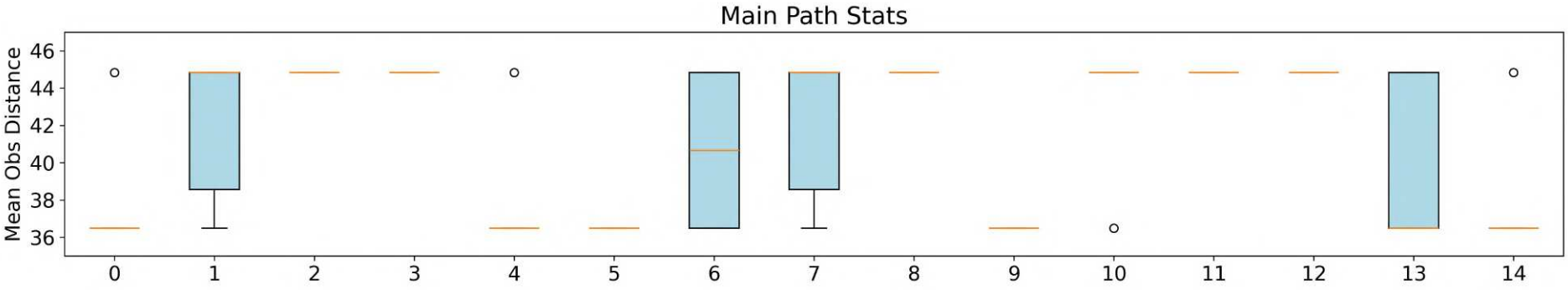
MP MOD - Group A - Constant ϵ values Figure 87 (a) and (b) shows the simulations results of Group A in Idx_{0-19}. The same pattern found on the previous metric is seen here.

MP MOD - Group B - variable ϵ values Figures 87 (b) and (c) shows the simulations results of Group B in Idx_{20-45}. The same pattern found on the previous metric is seen here, with a slight variation on Idx_{27} suggesting minor modifications to the safe route found by the agent.

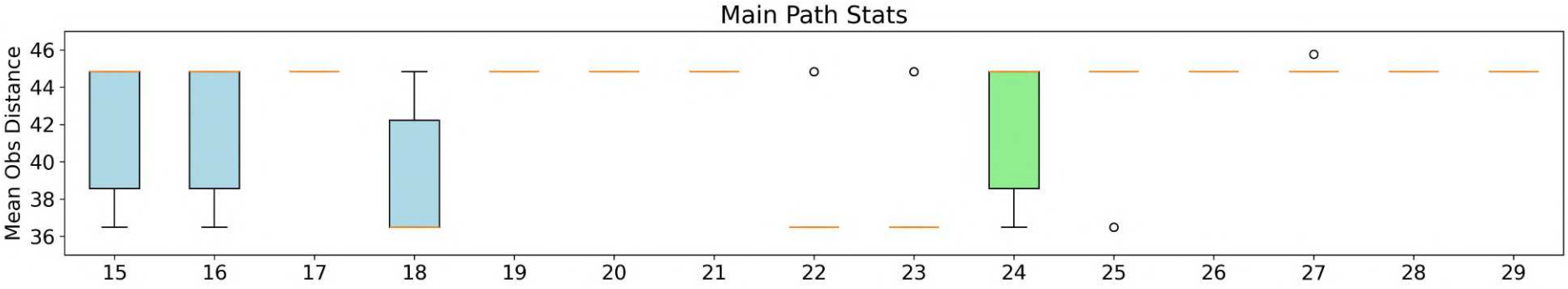
Since this metric exhibits greater variability compared to the PL metric (e.g., a path can have the same PL but a different MOD), its values showed slight variations but remained largely concentrated around two specific routes.Next section the MP TT metric will be discussed.

A.2.6.3 Main Path Total Turns metric

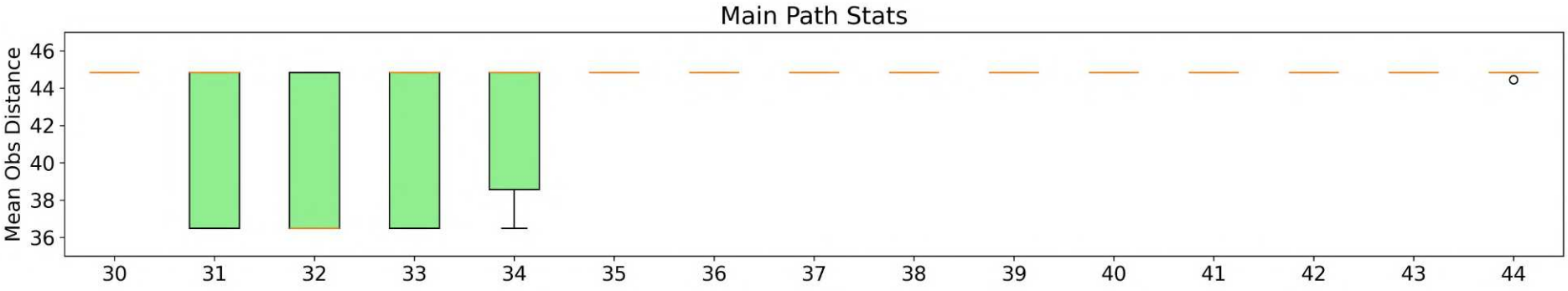
Figure 88 depicts the Total Turns metric computed for the Main Path, where the indexes and groups are detailed in Table 15.



(a) Results for Group A.

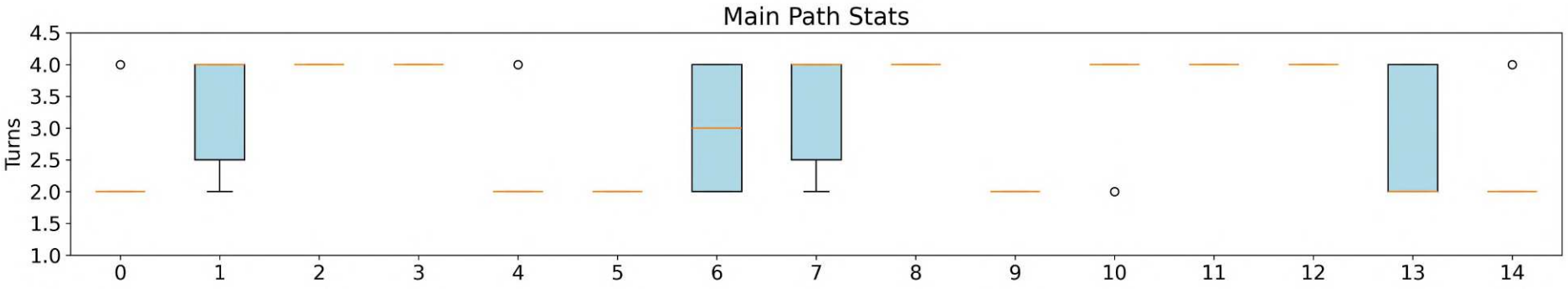


(b) Results for Groups A and B.

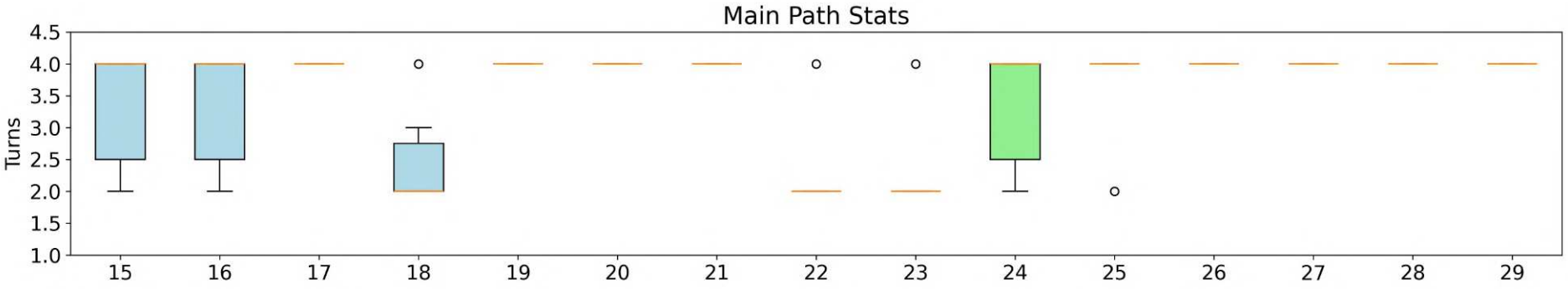


(c) Results for Group B.

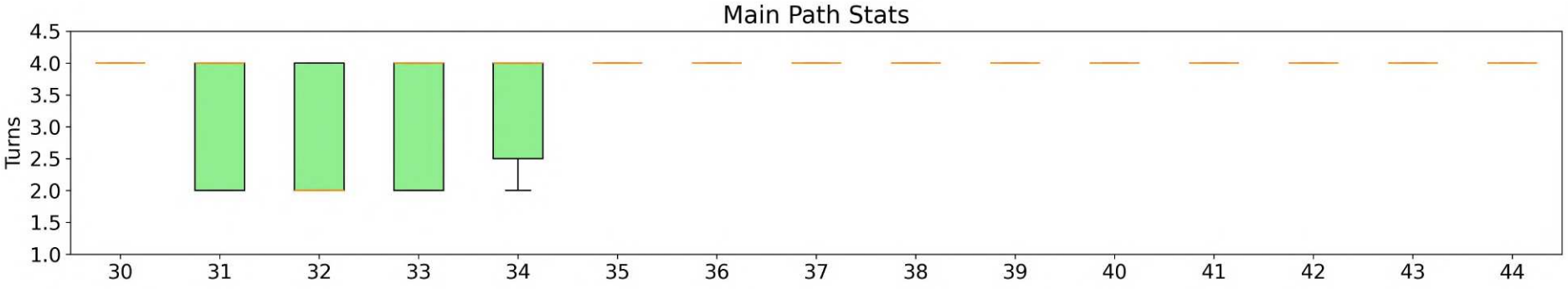
Figure 87 – Box Plot for Main Path Mean Obstacle Distance average metric. The colors light blue and light green represent the groups A and B, respectively.



(a) Results for Group A.



(b) Results for Groups A and B.



(c) Results for Group B.

Figure 88 – Box Plot for Main Path Total Turns average metric. The colors light blue and light green represent the groups A and B, respectively.

MP TT - Group A - Constant ϵ values Figure 87 (a) and (b) shows the simulations results of Group A in Idx_{0-19}. The same pattern found on the previous metrics is seen here.

MP TT - Group B - variable ϵ values Figures 87 (b) and (c) shows the simulations results of Group B in Idx_{20-45}. The same pattern found on the previous metrics is seen here.

The polarized results presented in this and the previous sections suggest that the agent effectively learned the two optimal paths: one that navigates between the three obstacles, offering a faster but less safe route, and another that moves around the obstacles, which is longer but safer. The following sections will explore the convergence speed of the metrics by examining the episode threshold required for the agent to reach within 2% of the final result.

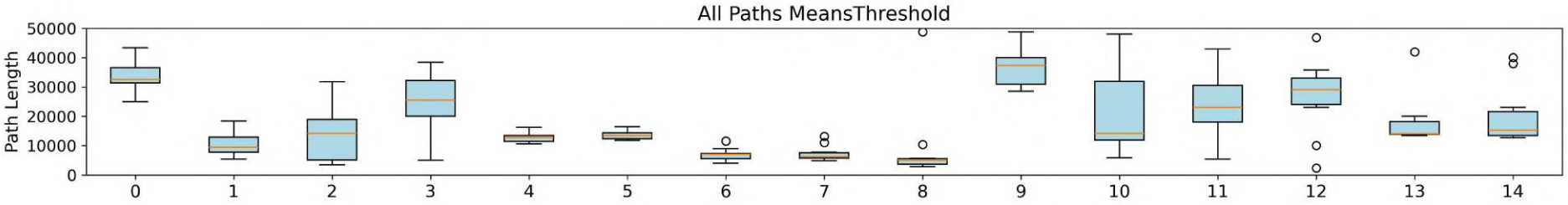
A.2.7 Box Plots for All Paths Metrics Thresholds

Box plots were used in order to have insights about the metric thresholds distribution over the 10 repeated simulations. These thresholds represent how fast each simulation reached and stayed in a region of ±2% of the best metric value. The number of episodes to reach the pre determined performance threshold for each of them are shown in Figures 89-91. The indexes in the x axis in the plots are numerical values that represent the combination of ϵ and α values used for that simulation. These indexes can be found in Table 15. They were grouped in two groups, A and B respectively, which will be plotted, in this order, with the following colors: light blue and light green akin to how it was displayed in the previous section.

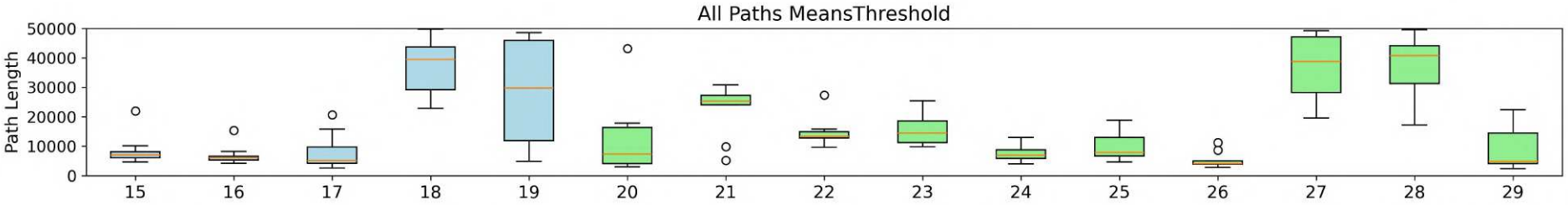
A.2.7.1 All Paths Path Length metric threshold

Figure 89 depicts the Path Length metric threshold computed for paths originating from every cell, where the indexes and groups are detailed in Table 15.

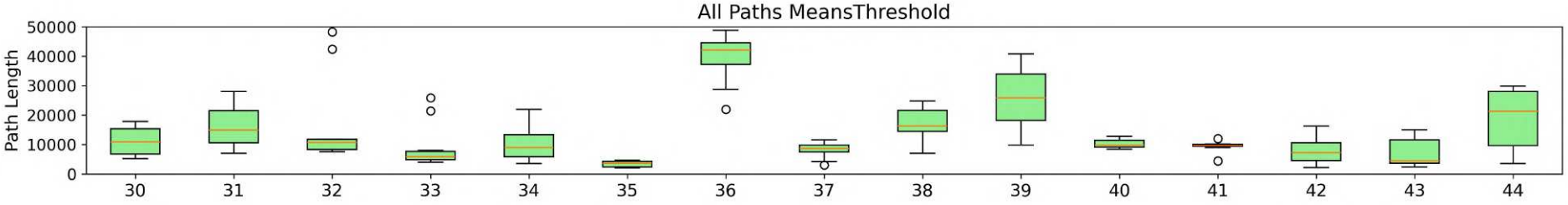
AP PL Threshold - Group A - Constant ϵ values Figure 89 (a) and (b) shows the simulations results of Group A in Idx_{0-19}. When $\epsilon = 0.7$ (Idx_{5-9}), the agent exhibited the quickest and most consistent convergence, suggesting that this constant value offers a more effective exploration/exploitation balance. In contrast, Idx_{15-17}, with a lower ϵ value of 0.3, also showed rapid and consistent convergence. This can likely be attributed to the low exploration rate combined with the Exploring Starts value, which limited the agent's exposure to variation, causing it to exploit its narrow knowledge base and reach its final value quickly.



(a) Results for Group A.



(b) Results for Groups A and B.



(c) Results for Group B.

Figure 89 – Box Plot for All Paths Path Length metric threshold average. The colors light blue and light green represent the groups A and B, respectively.

AP PL Threshold - Group B - variable ϵ values Figures 89 (b) and (c) shows the simulations results of Group B in Idx_{20-45}. This group demonstrated faster and more consistent convergence compared to the previous one. The best results were achieved in Idx_{20-24} and Idx_{30-34}, both of which had a final ϵ value of 0.1. This suggests that increased exploitation during the later stages of training can enhance convergence rates

The AP PL Threshold results demonstrated greater sensitivity to variations in hyperparameter combinations compared to the final AP PL metrics. This suggests that fine-tuning these parameters, which directly influence the balance between exploration and exploitation during different training stages, is crucial for achieving faster and more consistent convergence. In the next section the threshold values for the AP MOD metric will be discussed.

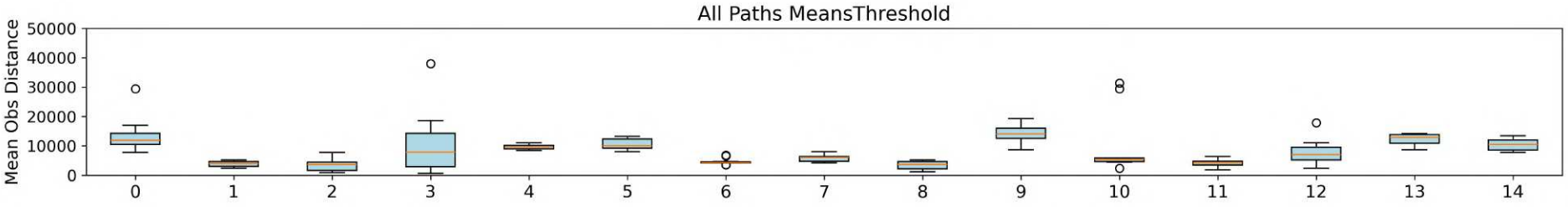
A.2.7.2 All Paths Mean Obstacle Distance metric threshold

Figure 90 depicts the Mean Obstacle Distance metric threshold computed for paths originating from every cell, where the indexes and groups are detailed in Table 15.

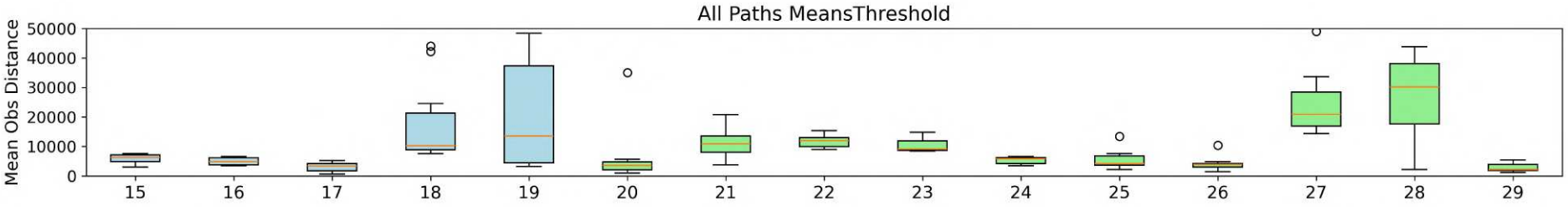
AP MOD Threshold - Group A - Constant ϵ values Figure 90 (a) and (b) shows the simulations results of Group A in Idx_{0-19}.Overall, the threshold values for this group are generally low, with minimal dispersion. However, a significant exception is observed in Idx_{19}, which utilizes $\epsilon = 0.3$ and 10,000 exploring starts. This suggests that such a low level of exploration hinders the agent's ability to improve in this metric.

AP MOD Threshold - Group B - variable ϵ values Figures 90 (b) and (c) shows the simulations results of Group B in Idx_{20-45}. Variable ϵ values, characterized by overall lower threshold levels and reduced dispersion, also demonstrate their viability in this context. However, combinations $Idx_{27,28}$ ($\epsilon \in [0.3, 1]$ with Exploring Starts of 100 and 1000, respectively) exhibit a significantly poorer performance. This suggests that such variation in the starting cell may be detrimental to this particular metric.

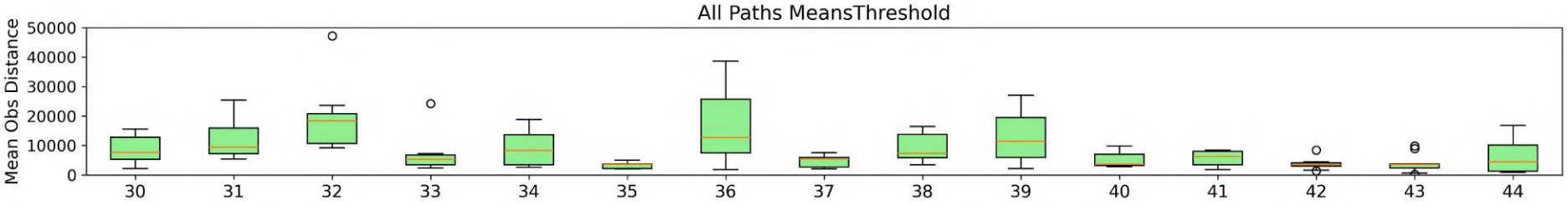
Overall, the AP MOD metric exhibited consistently lower values and less dispersion across most hyperparameter combinations compared to the AP PL metric. This difference may be attributed to the AP MOD metric's ability to make finer adjustments relative to its maximum possible value, thereby enabling it to remain within the 2% range while still achieving incremental improvements. In the next section the AP TT metric thresholds will be discussed.



(a) Results for Group A.



(b) Results for Groups A and B.



(c) Results for Group B.

Figure 90 – Box Plot for All Paths Mean Obstacle Distance metric threshold average. The colors light blue and light green represent the groups A and B, respectively.

A.2.7.3 All Paths Total Turns metric threshold

Figure 91 depicts the Total Turns metric threshold computed for paths originating from every cell, where the indexes and groups are detailed in Table 15.

AP TT Threshold - Group A - Constant ϵ values Figure 91 (a) and (b) shows the simulations results of Group A in Idx_{0-19}.The lowest threshold values and most consistent results were observed with $\epsilon = 0.7$ (Idx_{5-9}). Other combinations failed to achieve the same level of consistency. This indicates that for constant exploration rates, an ϵ value of 0.7 serves as a flexible middle ground, performing comparably well across different Exploring Starts values.

AP TT Threshold - Group B - variable ϵ values Figures 91 (b) and (c) shows the simulations results of Group B in Idx_{20-45}. Best results for this Group are seen using $\epsilon \in [0.3, 0.6]$ (Idx_{40-44}), suggesting that, for variable values of ϵ, highers exploration rates throughout training can lead to less consistent results.

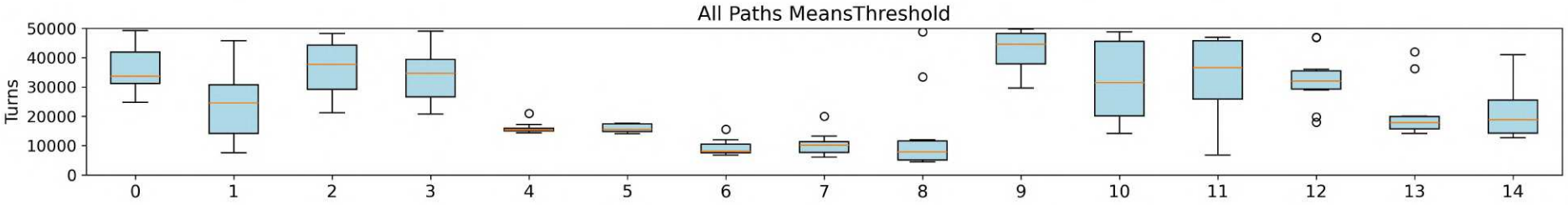
Among the three metrics, AP TT exhibited both a higher average threshold level and greater variability in the results. This increased dispersion can be attributed to the significant fluctuations in AP TT when compared to its maximum value. For example, if a path originally includes four turns and is reduced to three, this represents a 25% variation. Such a change demands considerably more stability to maintain convergence within the 2% threshold limit for this metric.In the next section, the MP metrics thresholds will be discussed.

A.2.8 Box Plots for Main Path Metrics Thresholds

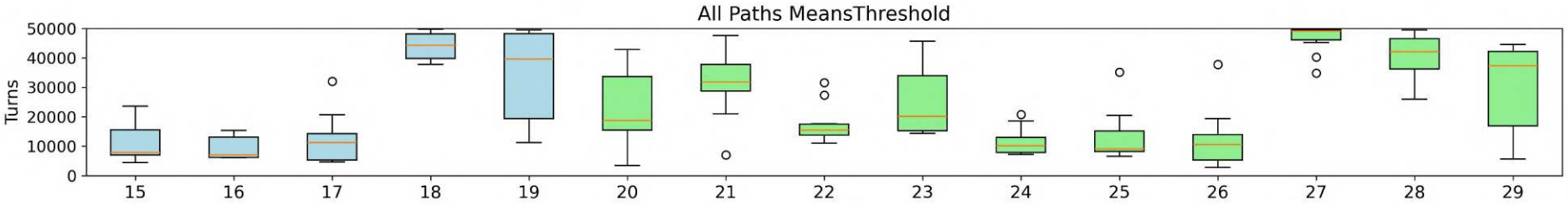
Box plots were used in order to have insights about the metric thresholds distribution over the 10 repeated simulations. These thresholds represent how fast each simulation reached and stayed in a region of $\pm 2\%$ of the best metric value. The number of episodes to reach the pre determined performance threshold for each of them are shown in Figures 92-94. The indexes in the x axis in the plots are numerical values that represent the combination of ϵ and α values used for that simulation. These indexes can be found in Table 15. They were grouped in two groups, A and B respectively, which will be plotted, in this order, with the following colors: light blue and light green akin to how it was displayed in the previous section.

A.2.8.1 Main Path Path Length metric threshold

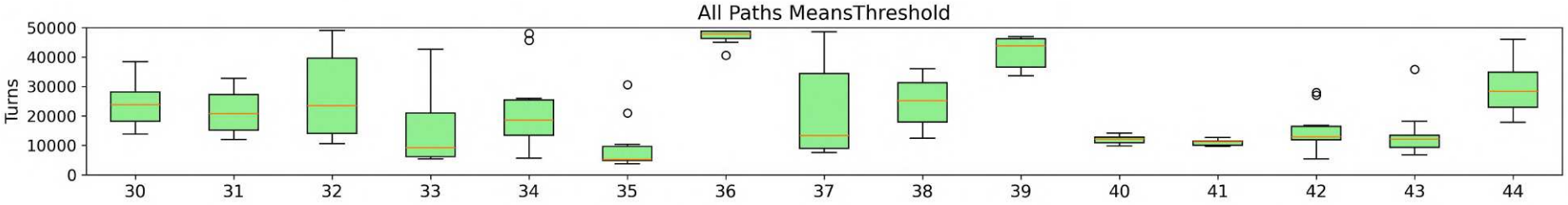
Figure 92 depicts the Path Length metric threshold computed for the Main Path, where the indexes and groups are detailed in Table 15.



(a) Results for Group A.

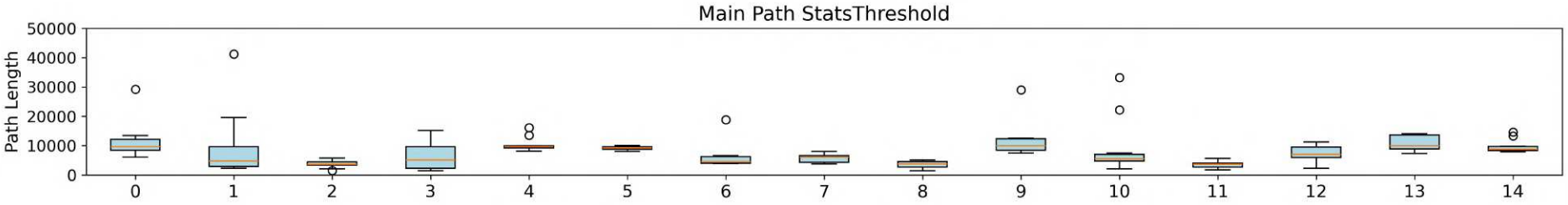


(b) Results for Groups A and B.

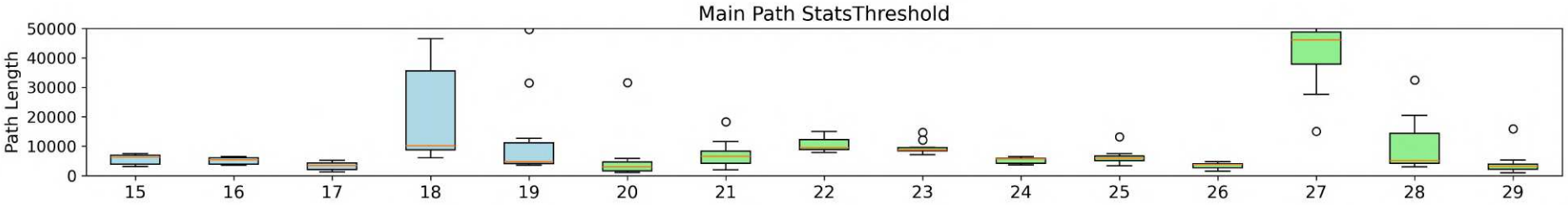


(c) Results for Group B.

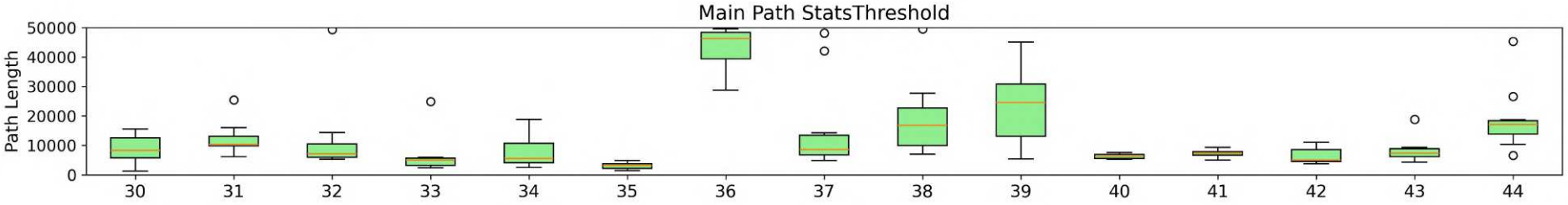
Figure 91 – Box Plot for All Paths Total Turns metric threshold average. The colors light blue and light green represent the groups A and B, respectively.



(a) Results for Group A.



(b) Results for Groups A and B.



(c) Results for Group B.

Figure 92 – Box Plot for Main Path average Path Length metric threshold. The colors light blue and light green represent the groups A and B, respectively.

MP PL Threshold - Group A - Constant ϵ values Figure 92 (a) and (b) shows the simulations results of Group A in Idx_{0-19}. Most simulations exhibited consistently low dispersion and threshold values, with the notable exception of Idx_{18}, which corresponds to $\epsilon = 0.3$ and 1000 Exploring Starts. This configuration resulted in significantly reduced exploration rates during training, thereby increasing the variability in the results.

MP PL Threshold - Group B - variable ϵ values Figures 92 (b) and (c) shows the simulations results of Group B in Idx_{20-45}. Compared to the AP metrics, this group continues to exhibit low threshold values and significant dispersion. The most pronounced declines are observed with $\epsilon \in [0.3, 0.8]$ for Exploring Starts greater than 1 (Idx_{36-40}). These results suggest that the additional exploration layer, involving frequent changes in the starting cell, significantly enhances this metric.

As expected of a simpler task, when compared to optimizing multiple paths at once, the threshold values as well as the dispersion in the results were much lower in the MP PL metric, with some few exceptions. In the next section, the MP MOD metric threshold will be discussed.

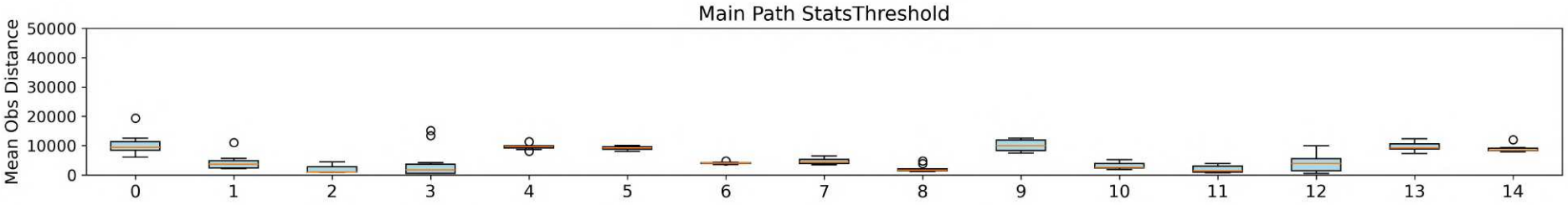
A.2.8.2 Main Path Mean Obstacle Distance metric threshold

Figure 93 depicts the Mean Obstacle Distance metric threshold computed for the Main Path, where the indexes and groups are detailed in Table 15.

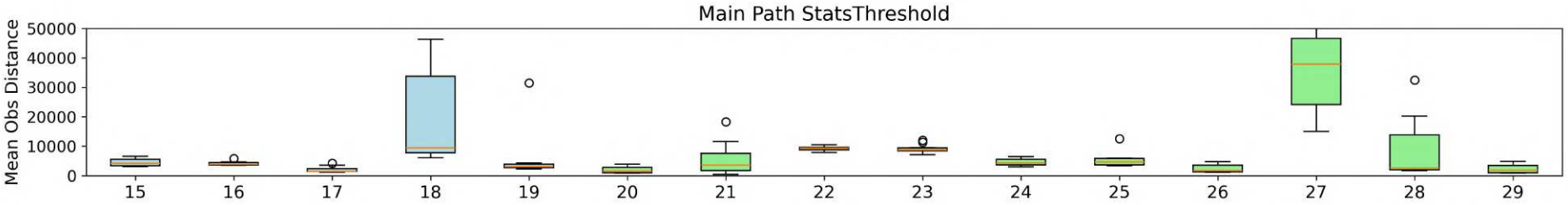
MP MOD Threshold - Group A - Constant ϵ values Figure 93 (a) and (b) shows the simulations results of Group A in Idx_{0-19}. As seen in the MP PL metric, there is little dispersion as well as lower threshold levels, except for Idx_{18}. It is possible to see that the dispersion levels were even lower in this metric, which suggests that it has more room for fine tuning while still staying in the 2% region.

MP MOD Threshold - Group B - variable ϵ values Figures 93 (b) and (c) shows the simulations results of Group B in Idx_{20-45}. This metric exhibits similar behavior to the previous one, as expected since both relate to the same path, but it shows slightly lower levels of dispersion

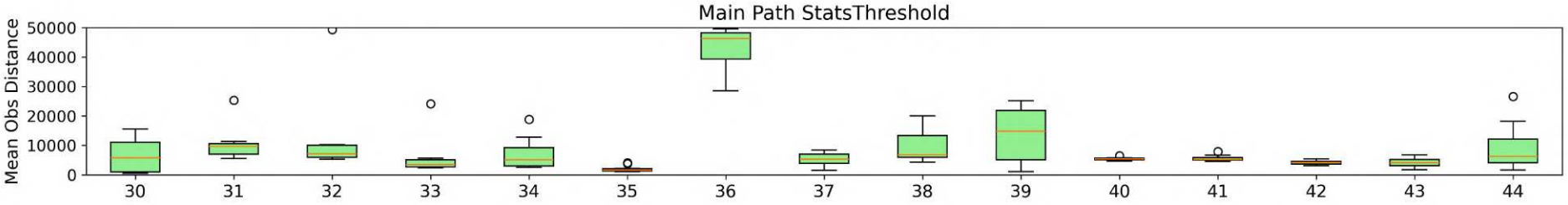
Overall the same trend of having lower dispersion as well threshold levels is seen in this metric, with the exception of few combinations. The next section will discuss the MP TT threshold metric.



(a) Results for Group A.



(b) Results for Groups A and B.



(c) Results for Group B.

Figure 93 – Box Plot for Main Path Mean Obstacle Distance metric threshold average. The colors light blue and light green represent the groups A and B, respectively.

A.2.8.3 Main Path Total Turns metric threshold

Figure 94 depicts the Total Turns metric threshold computed for the Main Path, where the indexes and groups are detailed in Table 15.

MP TT Threshold - Group A - Constant ϵ values Figure 94 (a) and (b) shows the simulations results of Group A in Idx_{0-19}. The same pattern found on metrics is seen here.

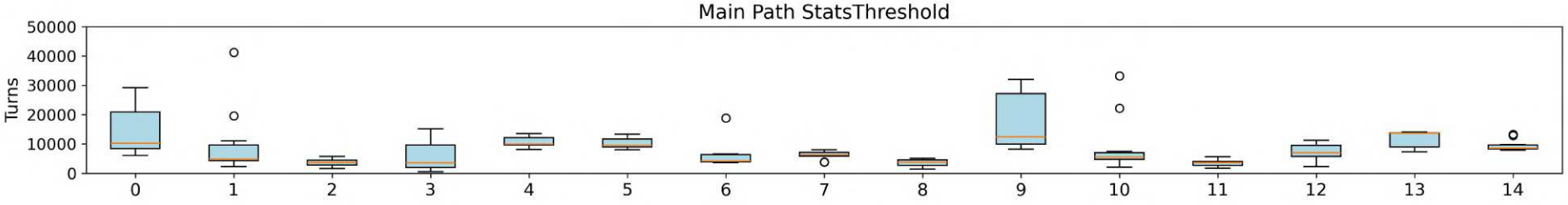
MP TT Threshold - Group B - variable ϵ values Figures 94 (b) and (c) shows the simulations results of Group B in Idx_{20-45}. The same pattern found on previous metrics is seen here.

The MP metrics exhibited largely consistent behavior, with only minor variations in threshold levels and dispersion. This consistency was maintained across a wide range of hyperparameter combinations, including the most extreme values for each parameter. In contrast, the AP metrics did not show this level of consistency, suggesting that the default number of episodes is typically sufficient to achieve convergence with ease for the main path. This can be attributed to the relative simplicity of optimizing the main path compared to the more complex task of optimizing paths from every cell in the map.

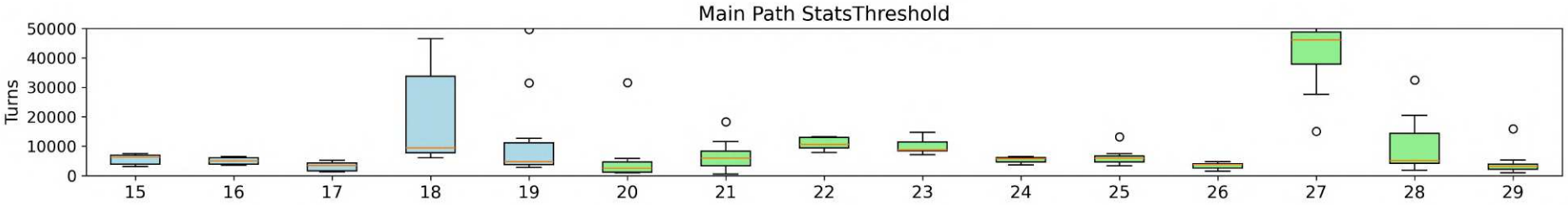
A.3 Comparison of Reward Constant Hyperparameters

This section provides an examination of the impact of varying reward constants for each of the priorities. These parameters regulate how the agent's actions will be encouraged or discouraged in the context of each of the main optimization constraints such as path length, safety and energy consumption. Each simulation was conducted iteratively for 10 repetitions.

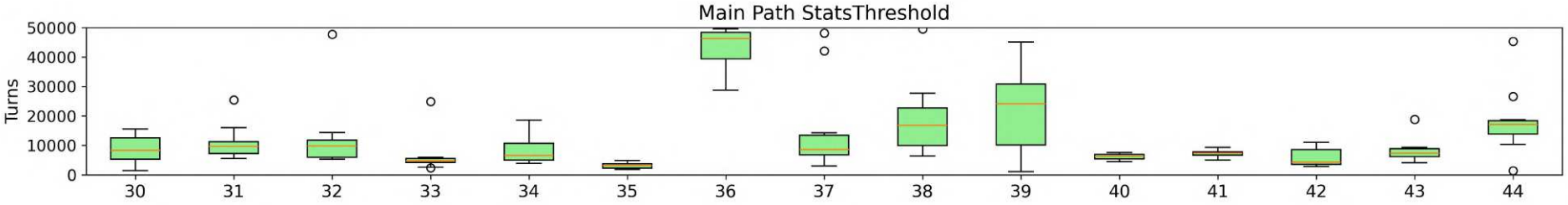
For this analysis, simulations were done in the map seen in 95, where the green cell is the origin point and the blue cell is the destination point. It is worth highlighting that during training, the agent may start in cells different from this origin point, due to the exploring starts strategy. In these Figures, the blue line represents the path taken by the agent and its length corresponds to the Path Length metric. Each time the agent takes a turn in its pathing, it adds up to the Total Turns metric, represented by orange dashed circles. Finally, to address safety concerns, the euclidean distance to the N closests obstacles for each cell of the agent's path is calculated and this is exemplified by the red dashed lines in the image with Obs_{d_i} being the distance to the i-th obstacle. For further reference, whenever the text refers to the "Main Path", it means the path that connects pre determined origin to the destination point and whenever the text refers to "All Paths",



(a) Results for Group A.



(b) Results for Groups A and B.



(c) Results for Group B.

Figure 94 – Box Plot for Main Path Total Turns metric threshold average. The colors light blue and light green represent the groups A and B, respectively.

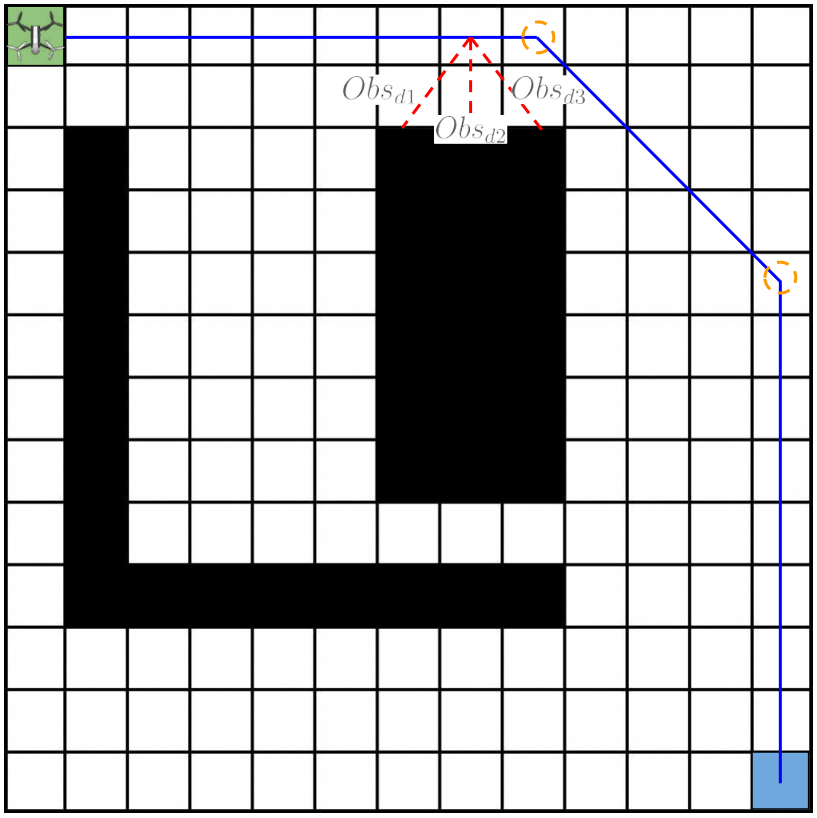


Figure 95 – Map used for each simulation in this section. Green and blue cells represent the starting and ending point respectively. Red dashed lines represent the euclidean distance to occupied cells, orange dashed circles represent each turn.

it means the group of paths originating from every cell of the map reaching the final destination point, also including the Main Path.

A.3.1 Performance Metrics and Representations

This section has the intent of showcasing the impact of varying the reward constants for each of the priorities over two different maps. In these simulations, only the final performance is analyzed given the nature of the hyperparameters that are being modified as well as the sheer amount of testing done that would lead to a cumbersome analysis with little discussion depth of the metrics evolution over the episodes or the convergence speed of each test Each simulation was conducted iteratively for 10 repetitions.

In conducting a performance analysis, it is imperative to not only determine the hyperparameters to be manipulated and the manner of their variation, but also the performance metrics for analysis and subsequent data visualization. Each of these components is described below.

- **Values of each priority Reward Constant** - 125 combinations for each of the two maps were used for the simulations, with the values for each constant ranging from -0.1 to -0.8, as seen bellow:

  - **Safety Constant**: -0.1, -0.3 , -0.5, -0.6, -0.8
  - **Step Constant**: -0.1, -0.3 , -0.5, -0.6, -0.8
  - **Energy Constant**: -0.1, -0.3 , -0.5, -0.6, -0.8

- **Performance Metrics** - Metrics related to two distinct aspects of our study were chosen to assess simulation performance: Main Path, given the significance of establishing an optimal path for the agent, and All Paths Means, considering that this algorithm emphasizes not only generating a viable path utilizing designated priorities from the starting point to the goal, but also producing quality paths from any location on the map. The following metrics are employed for both aspects and can be categorized into two groups, resulting in a total of 12 components, as outlined below.

  - **Final Performance Values** - These assess the values attained at the conclusion of training for each of the three priorities: Path Length, Mean Obstacle Distance, and Total Turns. These attributes evaluate the overall final performance for each set of hyperparameters.
  - **Performance Threshold** - These evaluate how quickly the agent was able to achieve and maintain performance within a certain threshold of the training's best performance for each of the three priorities. This evaluates not the absolute value, but the speed at which the specific hyperparameter set approached a performance level close to that obtained after all episodes. The threshold range employed was $\pm 2\%$ of the final value.

- **Data Visualization** - In this analysis only the final metrics values are taken in account. With that in mind, the most suited visualization tool is described bellow.

  - **Parallel Coordinates Plots** - These plots enable the grouping of all simulation metrics into fewer visualizations, allowing for an overall view of simulation performances and the identification of trends in hyperparameter sets. The metrics employed in these plots were the final value averaged over the 10 repeated simulations for each set.

Regarding the indexes, the permutations the constants regarding Safety (K_s), Path Length (K_d) and Energy Consumption (K_t) were varied as seen in Table 16.

A.3.2 Parallel Coordinates Plots

Parallel Coordinate Plots were employed as a visualization tool to comprehensively present the metrics and thresholds as well as more easily visualize trends throughout the hyperparameters configurations. In this type of plot, each line represents one simulation.

Index	K_s	K_d	K_t
0	-0.1	-0.1	-0.1
1	-0.1	-0.1	-0.3
2	-0.1	-0.1	-0.5
3	-0.1	-0.1	-0.6
4	-0.1	-0.1	-0.8
5	-0.1	-0.3	-0.1
6	-0.1	-0.3	-0.3
7	-0.1	-0.3	-0.5
8	-0.1	-0.3	-0.6
9	-0.1	-0.3	-0.8
10	-0.1	-0.5	-0.1
11	-0.1	-0.5	-0.3
12	-0.1	-0.5	-0.5
⋮	⋮	⋮	⋮
124	-0.8	-0.8	-0.8

Table 16 – Combinations of K_s, K_d, and K_t with assigned indices.

Starting with the map represented in Figure 95, we have the following discussion:

Final Performance Metrics In Figure 96, the metrics obtained from the simulations are depicted. It is possible to see that there are no clear patterns when it comes to the AP metrics with the exception of the AP TT, where green indexes showed worse results. When it comes to the MP metrics, there was lesser variability of results with the most polarizing metric being MP PL, with most results concentrated in the upper quarter of the column and the best results being with green indexes, indicating that with lower K_s, there is more room to find shorter paths, which is expected. MP TT also showed a great range os possible results with the lowest being two turns, and the highest 4.61.

Performance Thresholds Figure 97 illustrates the threshold (T) results of the simulations. Compared to the previous figure, these results exhibit a more streamlined result as the variations in the reward constants does not seem to create such mixed results as in the previous analysis, with the exception to the AP MODT, which had more varying results. The other metrics showed a more stable variation when changing indexes, which is well highlighted in the MP PL T and MP MOD T columns with progressively lower thresholds with more orange and red lines, indicating that higher reward constants may lead to quicker convergence. The MP Turns metric showed higher threshold due to struggling to stay within the 2% threshold limit due to higher variance in each update.

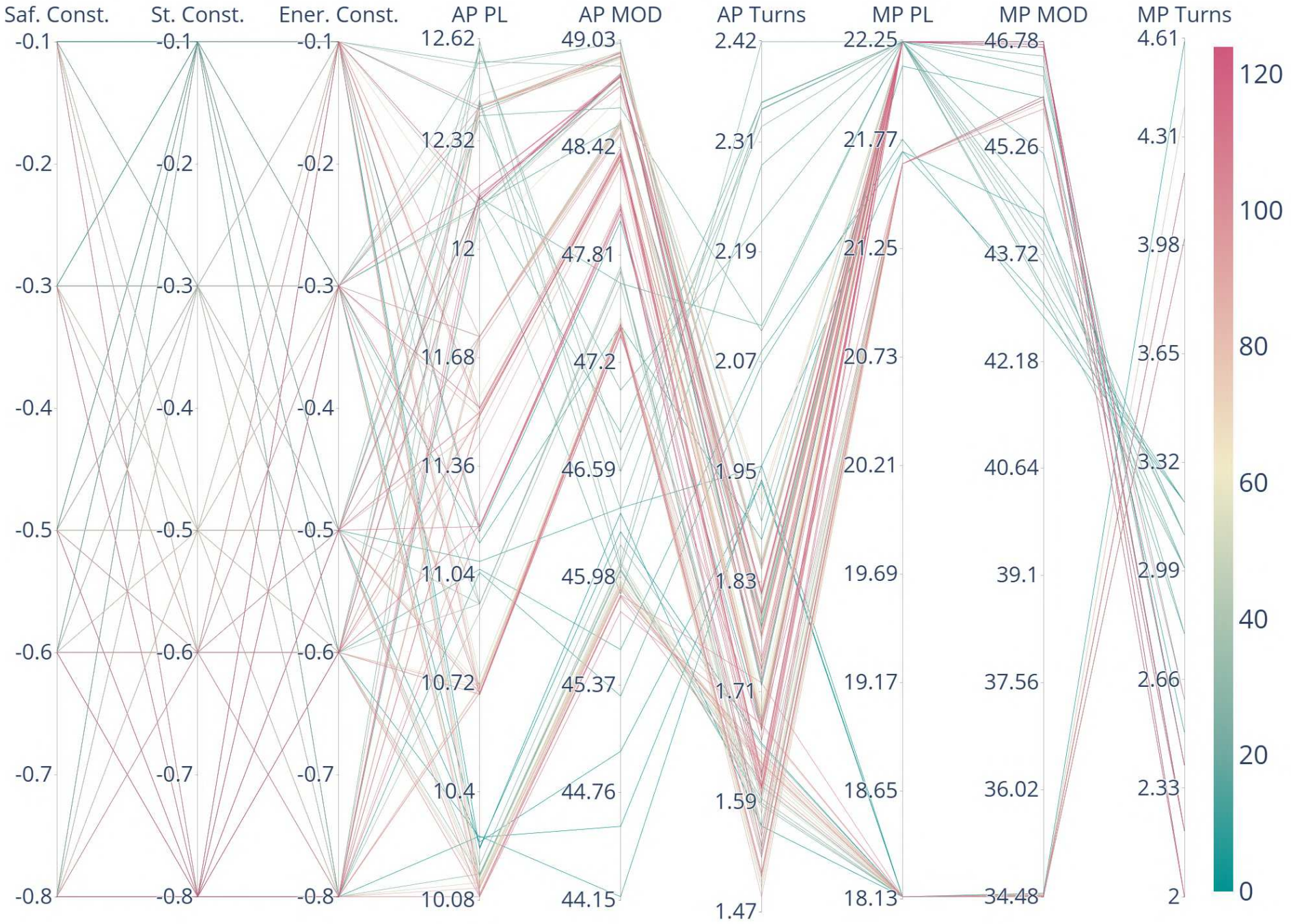


Figure 96 – Parallel Coordinates Plot for Metrics in all Simulations for Map 1. Saf. Const stands for K_s, St. Const stands for K_d and Ener. Const stands for K_t.

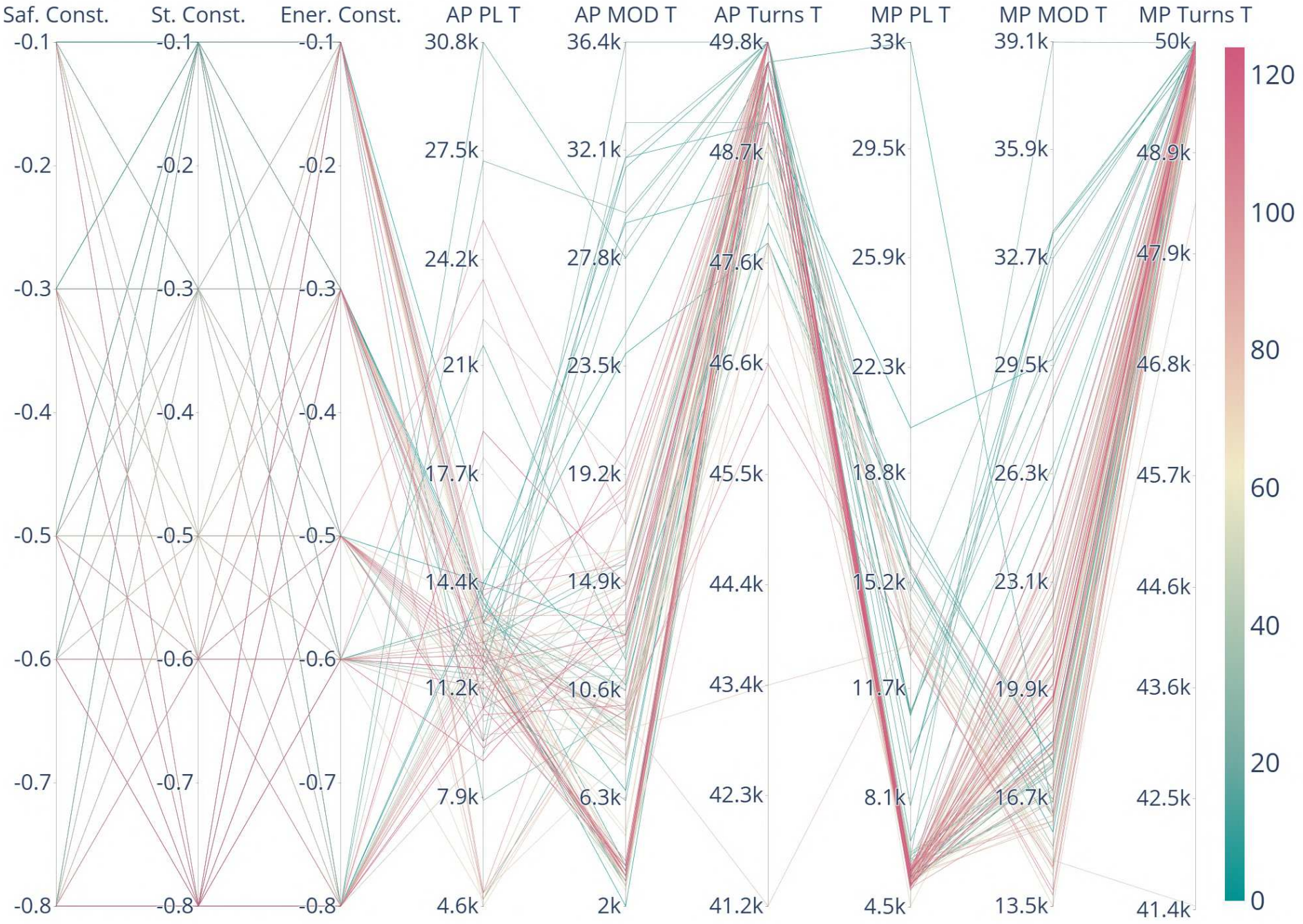


Figure 97 – Parallel Coordinates Plot for Threshold in all Simulations for Map 1. Saf. Const stands for K_s, St. Const stands for K_d and Ener. Const stands for K_t.

A.4 Q-Table Initialization

This analysis examines how different strategies for initializing the Q-Table can provide prior knowledge or guidance to the agent, thereby enabling more informed decisions, particularly in the early stages of training. Two approaches were explored: the first involves initializing the state-action pairs based on the shortest path calculated using Dijkstra's algorithm, while the second approach initializes the Q-Table such that the optimal action for each state is the one that directs the agent towards the goal. In both approaches, only one action per state had its Q-value modified. Furthermore, two different strategies were employed for setting these Q-values: a soft initialization with a Q-value of 5, and a more decisive initialization with a Q-value of 50.

For this analysis, two maps were used and they are represented in Figures 98 and 99, where the green cell is the origin point and the blue cell is the destination point. It is worth highlighting that during training, the agent may start in cells different from this origin point, due to the exploring starts strategy. In this Figures, the blue line represents the path taken by the agent and its length corresponds to the Path Length metric. Each time the agent takes a turn in its pathing, it adds up to the Total Turns metric, represented by orange dashed circles. Finally, to address safety concerns, the euclidean distance to the N closest obstacles for each cell of the agent's path is calculated and this is exemplified by the red dashed lines in the image with Obs_{d_i} being the distance to the i-th obstacle. For further reference, whenever the text refers to the "Main Path", it means the path that connects pre determined origin to the destination point and whenever the text refers to "All Paths", it means the group of paths originating from every cell of the map reaching the final destination point, also including the Main Path.

In conducting this performance analysis, it is imperative to not only to determine how the Q-values are being initiated, but also the performance metrics for analysis and subsequent data visualization. Each of these components is described below.

- **Initialization Methods** - The Q-table initially starts with a Q-value of 0 for all state-action pairs. To incorporate prior knowledge into the agent's learning process, two initialization strategies were employed:
  - **Dijkstra Approach** - In this method, Dijkstra's algorithm was applied to the map, and each state was initialized with the action suggested by the algorithm, starting with a higher Q-value.
  - **Heuristic Approach** - This method uses a simple heuristic to adjust Q-values. For each state, the action that most closely aligns with the goal had its Q-value increased.

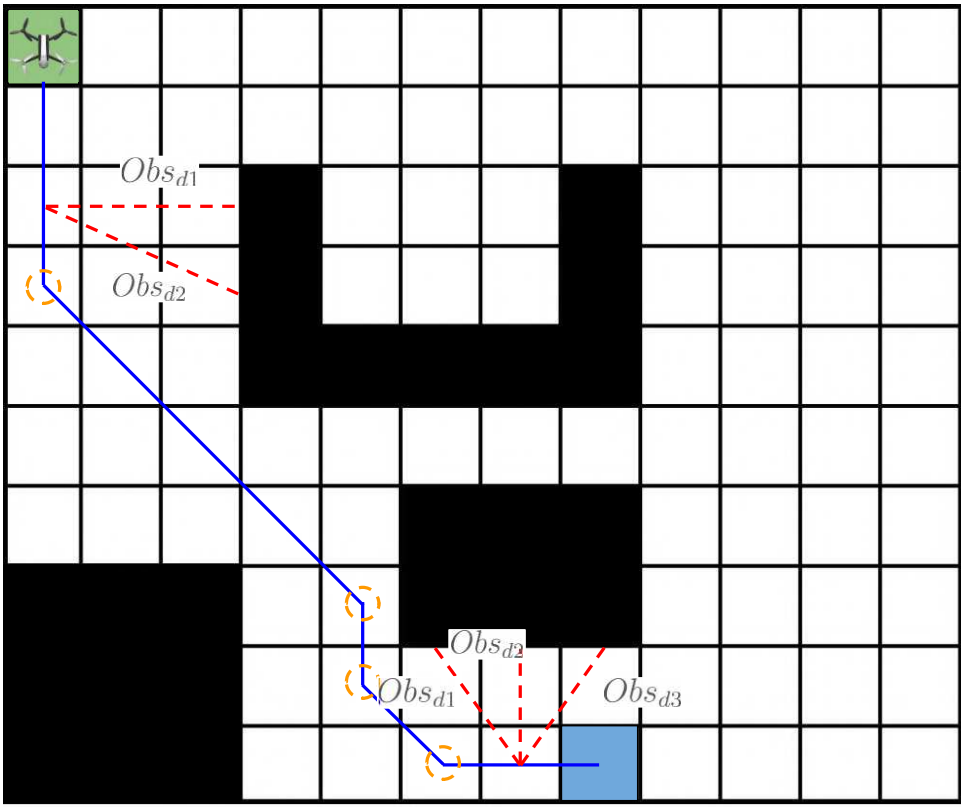


Figure 98 – Map 1 used in this section's simulations. Green and blue cells represent the starting and ending point respectively. Red dashed lines represent the euclidean distance to occupied cells, orange dashed circles represent each turn.

- **Q-Table Initial Values** - To evaluate the impact of Q-table initialization, two Q-values were tested: 5 and 50. Given that the goal reward is set to 100, these values represent a soft initialization and a more assertive initialization, respectively.
- **Data Visualization** - Two distinct techniques were used to visualize the results, each highlighting different aspects of the analysis:
  - **Final Results Table** - This table presents and discusses the average final results across all repetitions for each method, enabling a direct comparison of overall performance.
  - **2D Plots of Metric Evolution** - These plots depict the evolution of metrics over the episodes, providing a deeper understanding of how the initializations affect the agent's learning throughout training.

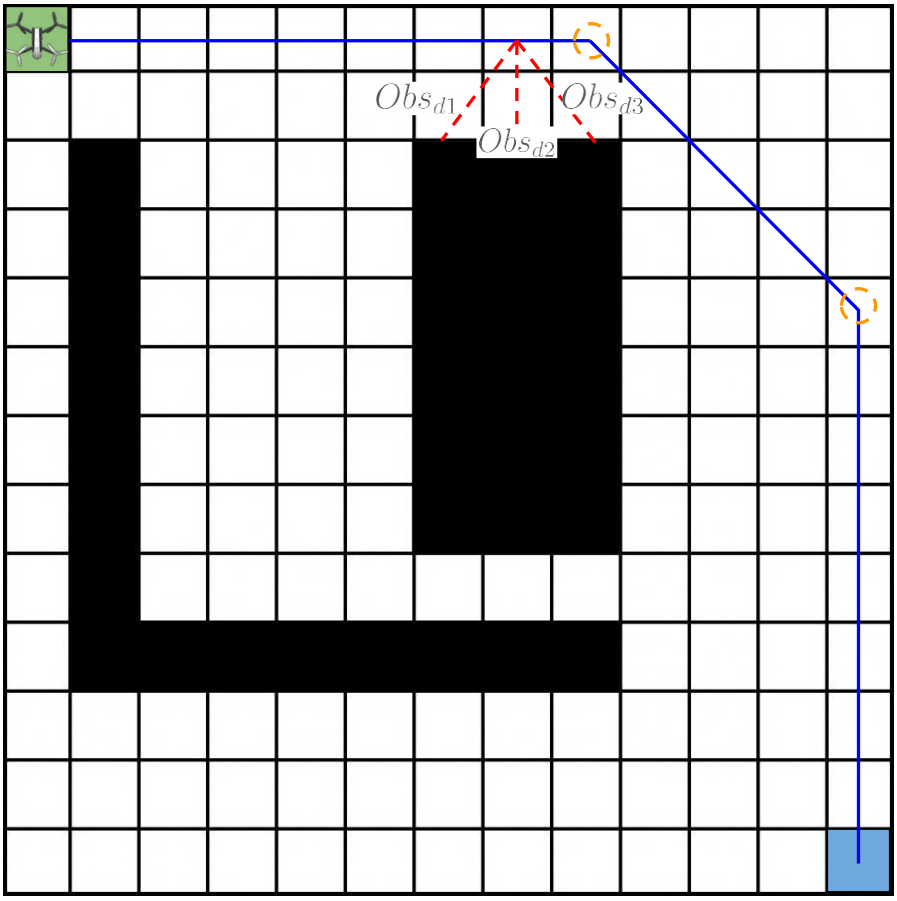


Figure 99 – Map 2 used in this section's simulations. Green and blue cells represent the starting and ending point respectively. Red dashed lines represent the euclidean distance to occupied cells, orange dashed circles represent each turn.

- **Performance Metrics** - Two categories of metrics were selected to assess the simulation performance: Main Path (MP) metrics, which emphasize the importance of establishing an optimal path for the agent, and All Paths Means (AP) metrics, which account for the algorithm's ability to generate quality paths from any point on the map. The following metrics were used to evaluate both aspects, categorized into two groups, resulting in a total of six components, as outlined below.
  - **Final Performance Values** - These metrics assess the final values achieved at the conclusion of training for three key priorities: Path Length (PL), Mean Obstacle Distance (MOD), and Total Turns (TT). These attributes are used to evaluate the overall performance of each set of hyperparameters.

Finally, all tests were done using the same Reward Constants of -0.6, -0.1, -0.6 and 100 for the Safety, Step, Energy and Goal Rewards, respectively, as well as $\epsilon \in [0.1, 1]$, $\alpha = 0.2$ and Exploring Starts frequency of 1.

A.4.1 2D Plots for All Paths Metrics

Regular 2D plots will serve as the foundation for analyzing the evolution of each metric throughout training. Each figure discussed will present the results for one of the three metrics, highlighting the outcomes for each initialization value and the two maps used. The values shown in these figures are averages derived from 10 repetitions conducted for each simulation. To reduce the inherent noise in the data, a moving average window with a span of 100 episodes was applied.

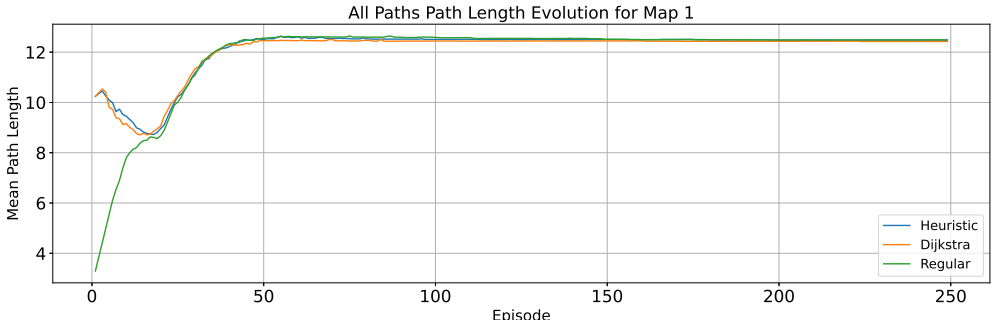
A.4.1.1 All Paths Path Length metric

Figure 100 depicts the Path Length metric computed for paths associated with every cell in the map, for each of the Q-value initialization values as well as comparing the two initialization methods with the regular approach.

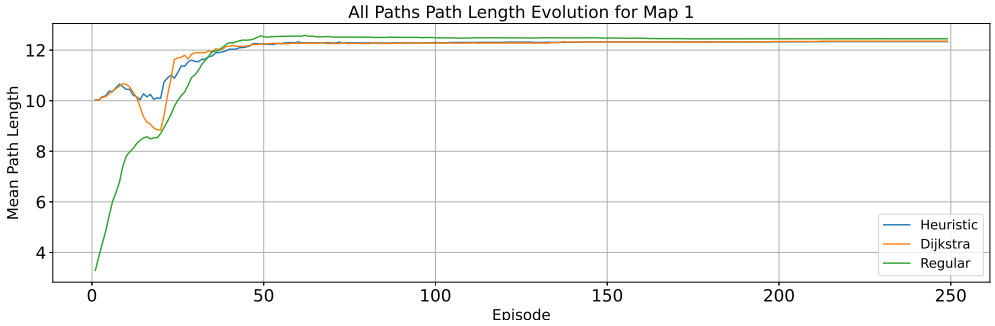
AP PL - Map 1 Figure 100 (a) and (b) presents the simulation results for both initial Q-values on Map 1, comparing the two initialization methods against the regular approach. For the soft initialization, the behavior of both methods is notably similar. Initially, both methods exhibit a higher average path length (PL), as the agent starts with a less optimal strategy. As training progresses, the agent learns to reach the goal from closer cells, leading to a reduction in the metric's value. Eventually, the regular approach also shows a similar improvement in PL, aligning with the results of the initialization methods. In contrast, with an initialization value of 50, the Heuristic approach shows a less significant reduction in the metric compared to the Dijkstra method. However, both initialization methods reach values close to the final PL much faster than the regular approach.

AP PL - Map 2 Figure 100 (c) and (d) presents the simulation results for both initial Q-values on Map 2, comparing the two initialization methods with the regular approach. Similar to the observations on the previous map, both initialization methods display comparable behavior, characterized by a 'valley' in performance with the soft initialization. On this map, however, it is evident that with an initialization value of 50, both approaches demonstrate a reduced overshoot compared to the regular strategy, indicating a more stable performance.

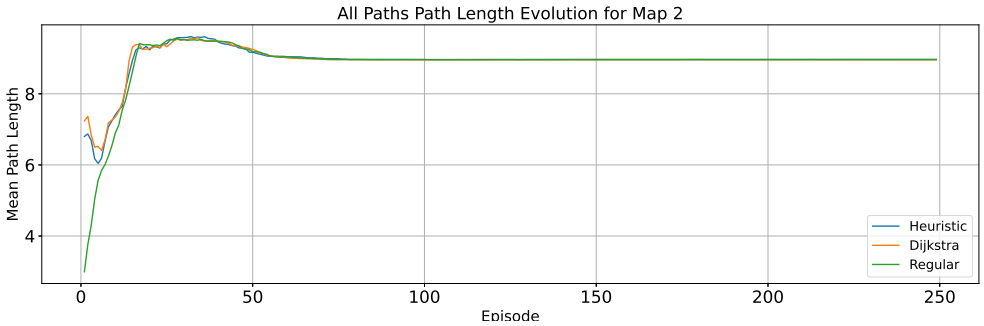
Overall, both approaches exhibited similar behavior across both maps. However, the more assertive initialization led to faster convergence towards the final values, indicating a more positive impact on the agent's learning for this metric. Next section the performance on the AP MOD metric will be discussed.



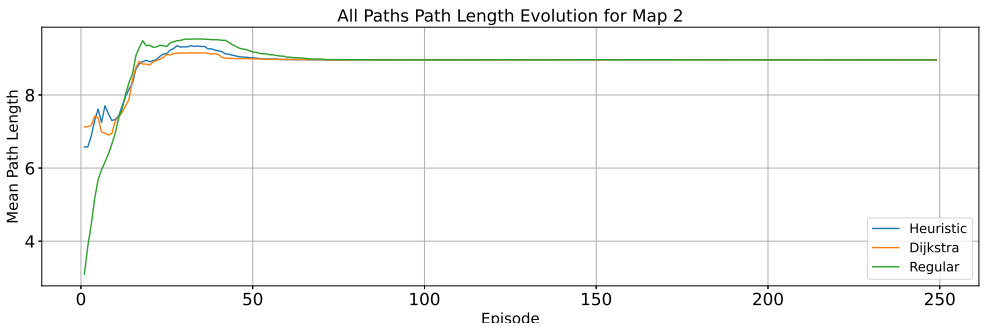
(a) All Paths Path Length results for Map 1 using 5 as initialization Q-value



(b) All Paths Path Length results for Map 1 using 50 as initialization Q-value



(c) All Paths Path Length results for Map 2 using 5 as initialization Q-value



(d) All Paths Path Length results for Map 2 using 50 as initialization Q-value

Figure 100 – All Paths Path Length average throughout training for each Map, Initialization Q-value and Method.

A.4.1.2 All Paths Mean Obstacle Distance metric

Figure 101 depicts the Mean Obstacle Distance metric computed for paths associated with every cell in the map, for each of the Q-value initialization values as well as comparing the two initialization methods with the regular approach.

AP MOD - Map 1 Figure 101 (a) and (b) present the simulation results for both initial Q-values on Map 1, comparing the two initialization methods against the regular approach. For an initial Q-value of 5, both initialization methods exhibit similar progression in the early stages, although Dijkstra's approach starts with a slightly lower MOD than the alternative method. In contrast, with an initial Q-value of 50, the orange line quickly surpasses the blue line in the early stages, indicating that a more assertive initialization facilitates more effective learning in this scenario.

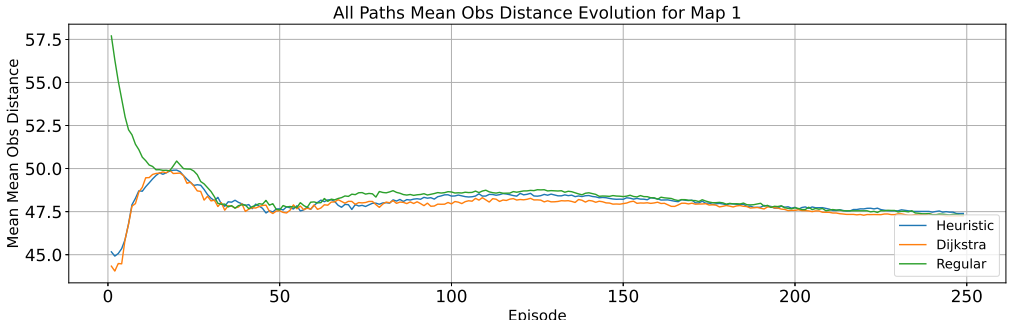
AP MOD - Map 2 Figure 101 (c) and (d) present the simulation results for both initial Q-values on Map 2, comparing the two initialization methods with the regular approach. On this map, all three plots exhibit similar progress, with the heuristic approach showing slightly better performance in the early stages when the initial Q-value is set to 5. However, overall, the evolution of all approaches remains comparable. This outcome may be attributed to Map 2 being a simpler environment compared to Map 1.

Different of what was seen in the previous metric, where initial Q-value of 50 had the most distinguished behavior between the two initialization approaches, there was not a clear overall best initial value for both maps. Next section the performance on the AP TT metric will be discussed.

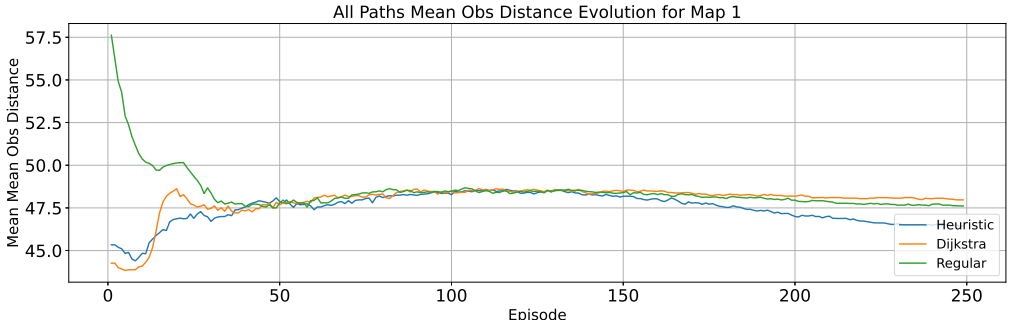
A.4.1.3 All Paths Total Turns metric

Figure 102 depicts the Total Turns metric computed for paths associated with every cell in the map, for each of the Q-value initialization values as well as comparing the two initialization methods with the regular approach.

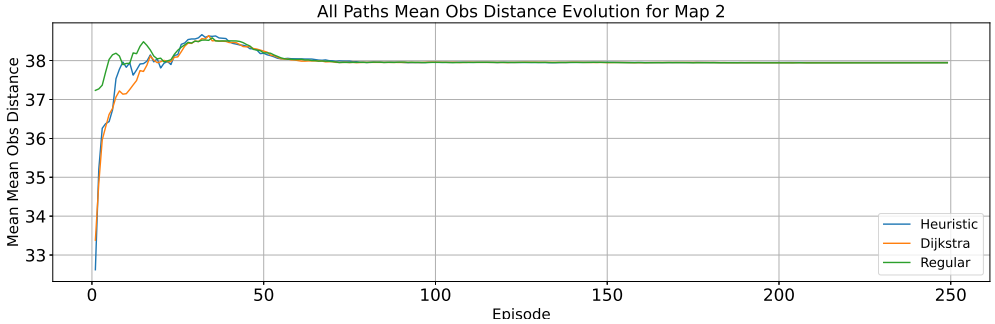
AP TT - Map 1 Figure 101 (a) and (b) present the simulation results for both initial Q-values on Map 1, comparing the two initialization methods against the regular approach. In this map, the heuristic approach slightly outperforms Dijkstra's algorithm when the initial Q-value is 5, but shows a more significant advantage when the initial value is set to 50. However, with a more assertive Q-table initialization, although the heuristic approach initially outperforms Dijkstra's algorithm, Dijkstra's eventually surpasses it by the end of training.



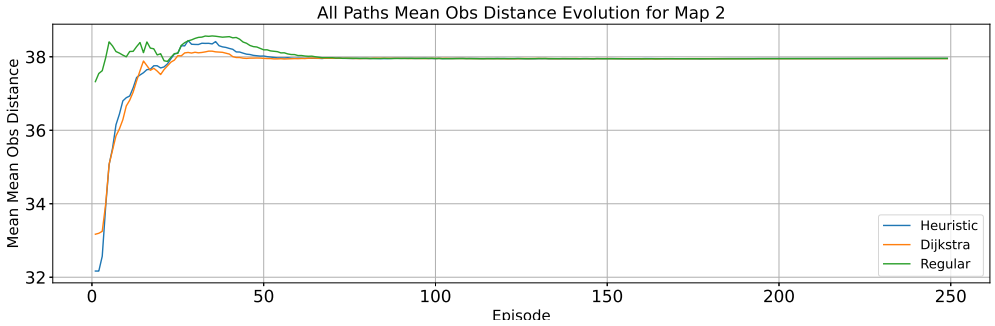
(a) All Paths Mean Obstacle Distance results for Map 1 using 5 as initialization Q-value



(b) All Paths Mean Obstacle Distance results for Map 1 using 50 as initialization Q-value

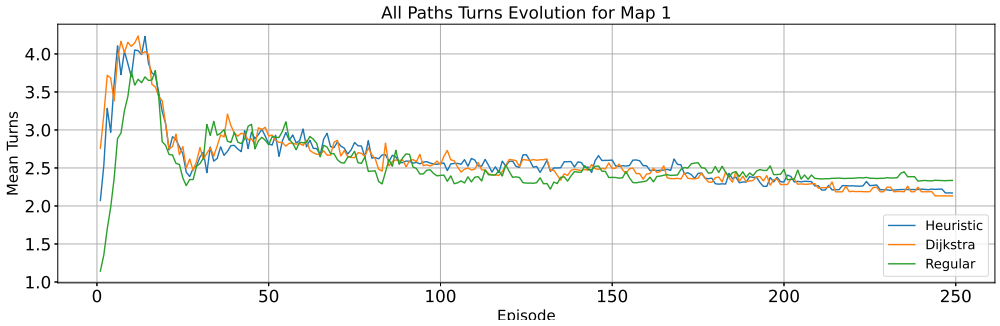


(c) All Paths Mean Obstacle Distance results for Map 2 using 5 as initialization Q-value

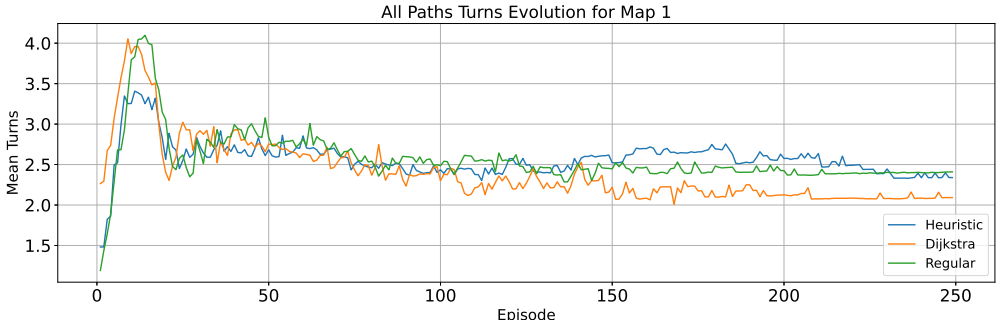


(d) All Paths Mean Obstacle Distance results for Map 2 using 50 as initialization Q-value

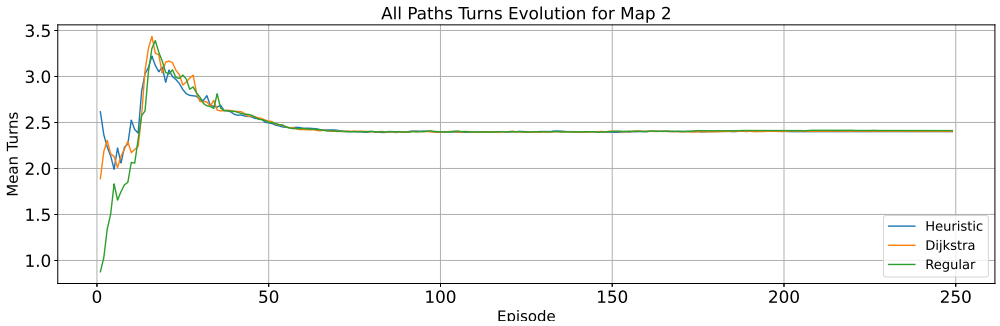
Figure 101 – All Paths Mean Obstacle Distance average throughout training for each Map, Initialization Q-value and Method.



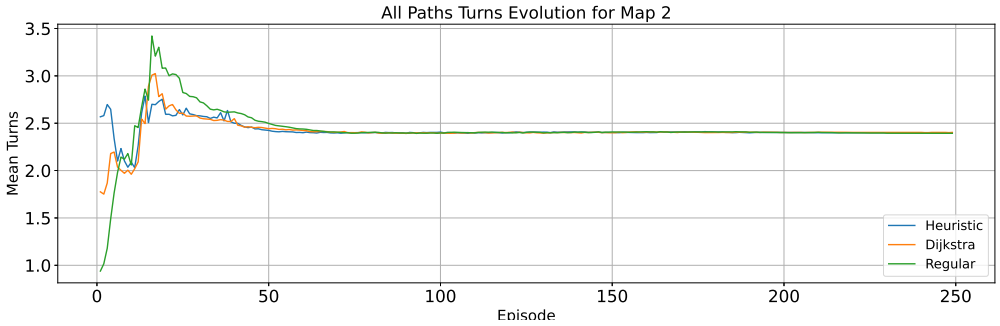
(a) All Paths Total Turns results for Map 1 using 5 as initialization Q-value



(b) All Paths Total Turns results for Map 1 using 50 as initialization Q-value



(c) All Paths Total Turns results for Map 2 using 5 as initialization Q-value



(d) All Paths Total Turns results for Map 2 using 50 as initialization Q-value

Figure 102 – All Paths Total Turns average throughout training for each Map, Initialization Q-value and Method.

AP TT - Map 2 Figure 101 (c) and (d) present the simulation results for both initial Q-values on Map 2, comparing the two initialization methods with the regular approach. This map exhibits a smoother evolution compared to Map 1, likely due to its simpler environment. With the soft initialization of the Q-table, both the Heuristic and Dijkstra approaches performed similarly to the regular method. However, when using an initial Q-value of 50, both initialization methods demonstrated significantly less overshoot than the regular approach.

This metric had more noisy evolution for Map 1, when compared to the other metrics, which can be associated to the variation step of this metric being larger when compared to the maximum value it has. The more incisive initialization of the Q-table led to more apparent differences compared to the soft initialization. Next section the Path Length metric for the Main Path will be discussed.

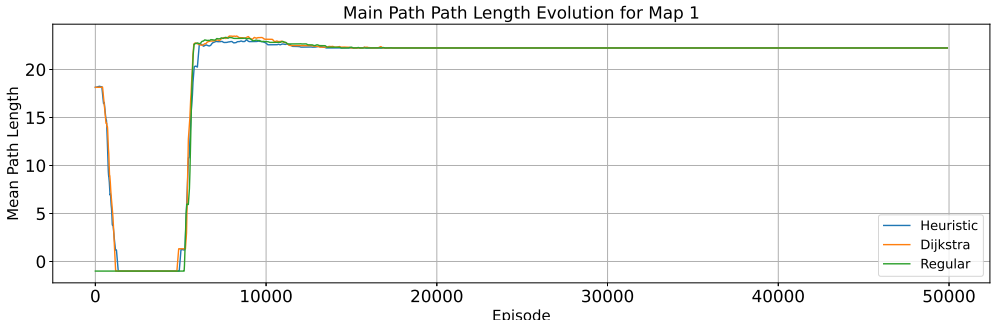
A.4.2 2D Plots for Main Path Metrics

Regular 2D plots will serve as the foundation for analyzing the evolution of each metric throughout training. Each figure discussed will present the results for one of the three metrics, highlighting the outcomes for each initialization value and the two maps used. The values shown in these figures are averages derived from 10 repetitions conducted for each simulation. To reduce the inherent noise in the data, a moving average window with a span of 100 episodes was applied.

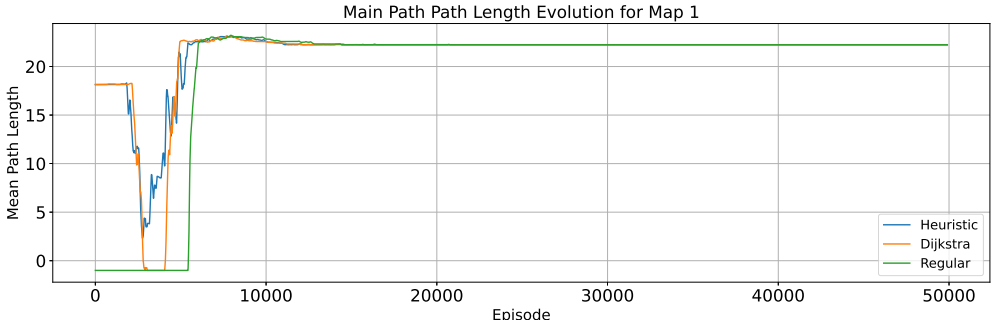
A.4.2.1 Main Path Path Length metric

Figure 103 depicts the Path Length metric computed for the Main Path, for each of the Q-value initialization values as well as comparing the two initialization methods with the regular approach.

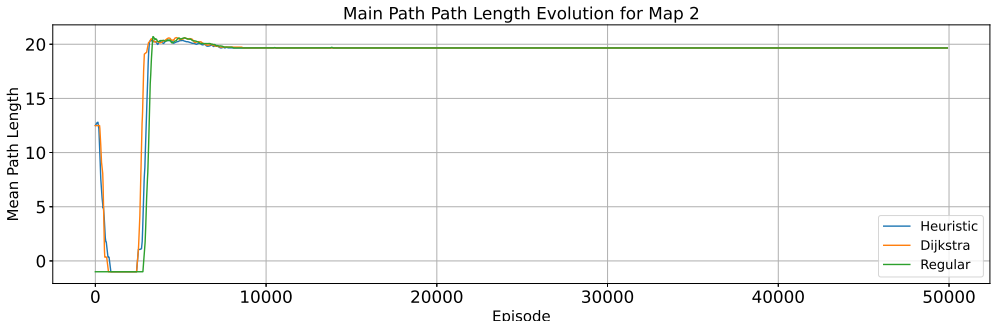
MP PL - Map 1 Figure 103 (a) and (b) presents the simulation results for both initial Q-values on Map 1, comparing the two initialization methods against the regular approach. Both initialization methods show that the agent is initially capable of reaching the destination, but for some episodes, the agent loses track of the goal. This relearning phase, where the agent must once again figure out how to reach the destination, is prolonged when using an initial Q-value of 5. With a more assertive initialization, the heuristic approach recovers from the performance drop more quickly than Dijkstra's algorithm. However, both methods converge to values closer to the final performance level before the regular method. This pattern of strong early performance followed by temporary failure to reach the destination is primarily driven by the higher exploration rates during the initial episodes of training.



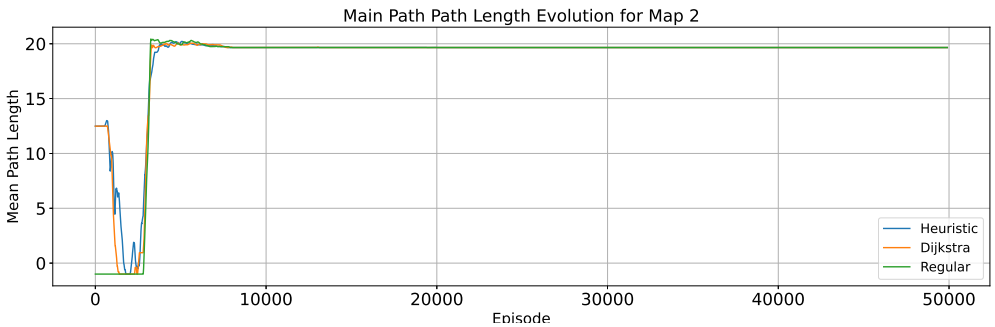
(a) Main Path Path Length results for Map 1 using 5 as initialization Q-value



(b) Main Path Path Length results for Map 1 using 50 as initialization Q-value



(c) Main Path Path Length results for Map 2 using 5 as initialization Q-value



(d) Main Path Path Length results for Map 2 using 50 as initialization Q-value

Figure 103 – Main Path Path Length average throughout training for each Map, Initialization Q-value and Method.

MP PL - Map 2 Figure 103 (c) and (d) presents the simulation results for both initial Q-values on Map 2, comparing the two initialization methods with the regular approach. A similar pattern to that observed in Map 1 emerges, where both initialization methods occasionally fail to reach the destination in some episodes but eventually succeed in finding a feasible path. However, in this map, the performance gap between the initial Q-values of 5 and 50 is less pronounced compared to Map 1.

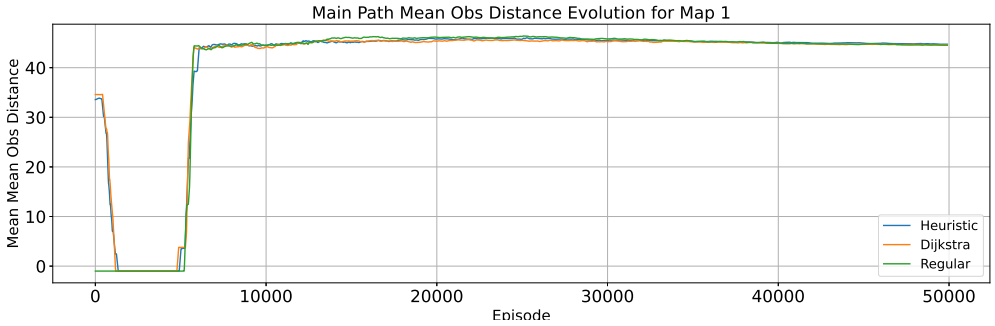
In both maps is possible to see that the initialization methods require an relearning phase, where the initial knowledge lead to better results than the updated one after some episodes due to the exploration done in the early stages of training. This phenomena is more easy to see analyzing metrics of the Main Path, given the data focuses in one path only, but certainly it is also present in All Paths metrics as well. In the next section the Main Path Mean Obstacle Distance metric is discussed.

A.4.2.2 Main Path Mean Obstacle Distance metric

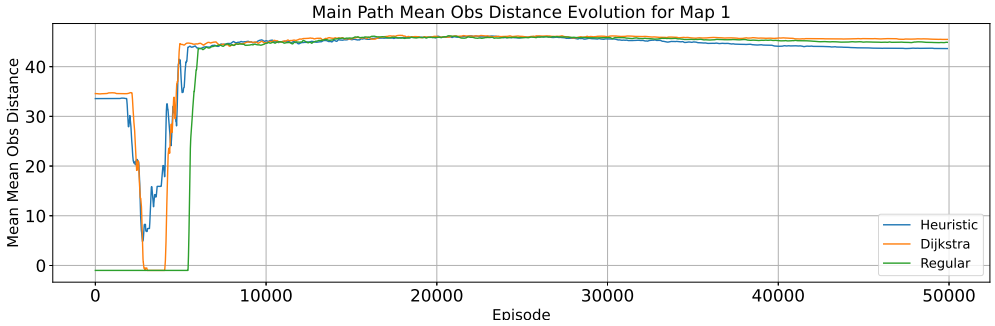
Figure 104 depicts the Mean Obstacle Distance metric computed for the Main Path, for each of the Q-value initialization values as well as comparing the two initialization methods with the regular approach.

MP MOD - Map 1 Figure 104 (a) and (b) presents the simulation results for both initial Q-values on Map 1, comparing the two initialization methods against the regular approach. With an initial Q-value of 5, both initialization methods showed comparable evolution and overall performance, with similar periods lacking feasible paths. However, when a more assertive initialization was applied, the heuristic method maintained its performance more effectively than Dijkstra's, which experienced a more noticeable drop. Although the heuristic approach proved slightly more resilient, its final results were marginally lower than the other methods, suggesting that this assertive initialization did not provide a clear benefit in terms of safety.

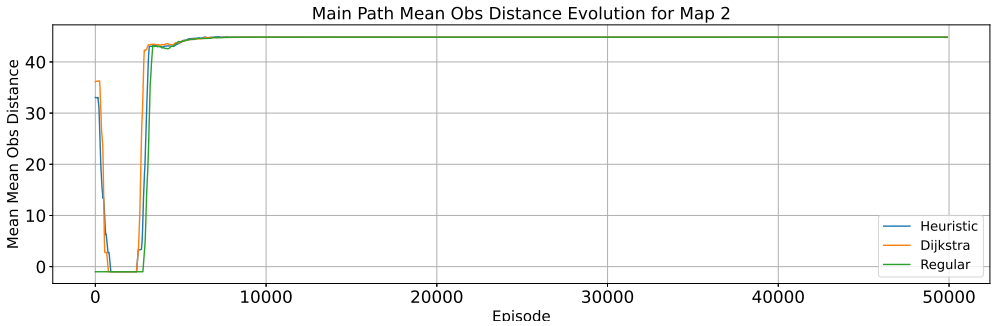
MP MOD - Map 2 Figure 104 (c) and (d) presents the simulation results for both initial Q-values on Map 2, comparing the two initialization methods with the regular approach. In a similar pattern to Map 1, both initialization methods with an initial Q-value of 5 exhibited comparable performance during the early stages. However, by the end, all strategies converged to similar results for this metric. When using an initial Q-value of 50, the heuristic approach faced a less performance hindering, although the effect was less pronounced than on Map 1. Ultimately, the final values were equivalent across all approaches. This outcome may be attributed to the simpler structure of Map 2 compared to Map 1.



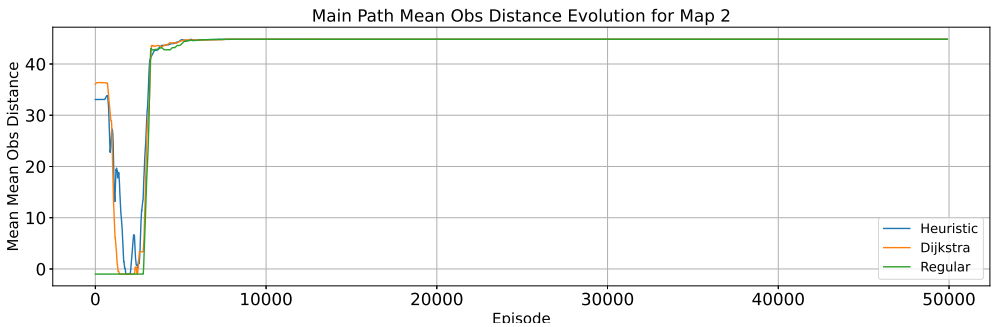
(a) Main Path Mean Obstacle Distance results for Map 1 using 5 as initialization Q-value



(b) Main Path Mean Obstacle Distance results for Map 1 using 50 as initialization Q-value



(c) Main Path Mean Obstacle Distance results for Map 2 using 5 as initialization Q-value



(d) Main Path Mean Obstacle Distance results for Map 2 using 50 as initialization Q-value

Figure 104 – Main Path Mean Obstacle Distance average throughout training for each Map, Initialization Q-value and Method.

In general, the final results across the different strategies showed only minor variations. However, using Dijkstra initialization with an initial Q-value of 50 on Map 1 yielded slightly better performance. Although this strategy initially lagged behind during early training stages, it demonstrated a notable capacity to recover and ultimately performed comparably to the other approaches. For Map 2, the final metric values were similar across all strategies, but the heuristic approach displayed a marginal advantage in the early stages when initialized with a Q-value of 50. This suggests that while initialization choices can influence early training dynamics, their impact were not as impactful regarding the final value of this metric.

A.4.2.3 Main Path Total Turns metric

Figure 105 depicts the Total Turns metric computed for the Main Path, for each of the Q-value initialization values as well as comparing the two initialization methods with the regular approach.

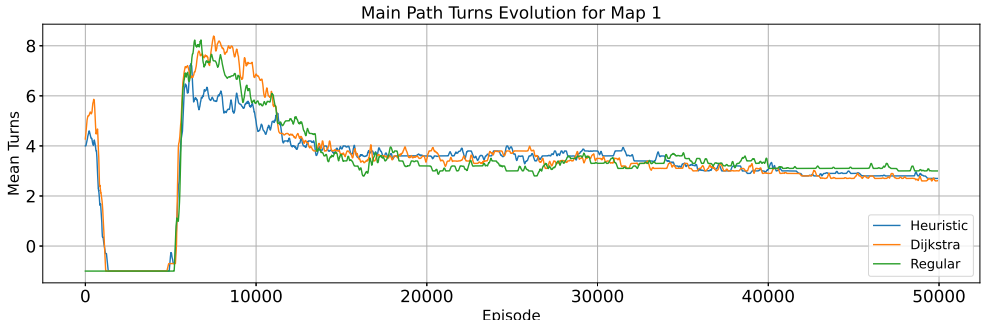
MP TT - Map 1 Figure 105 (a) and (b) presents the simulation results for both initial Q-values on Map 1, comparing the two initialization methods against the regular approach. For this map, both initialization methods outperformed the regular approach in the final results, with a larger gap using the more assertive initialization. With an initial Q-value of 5, the Dijkstra initialization shows a higher overshoot, while with an initial Q-value of 50, all three approaches perform similarly in terms of overshoot.

MP TT - Map 2 Figure 105 (c) and (d) presents the simulation results for both initial Q-values on Map 2, comparing the two initialization methods with the regular approach. Given the simplicity of this map, both initialization values provide similar final results for all three approaches, with a higher overshot for the regular strategy when using initial Q-values of 50.

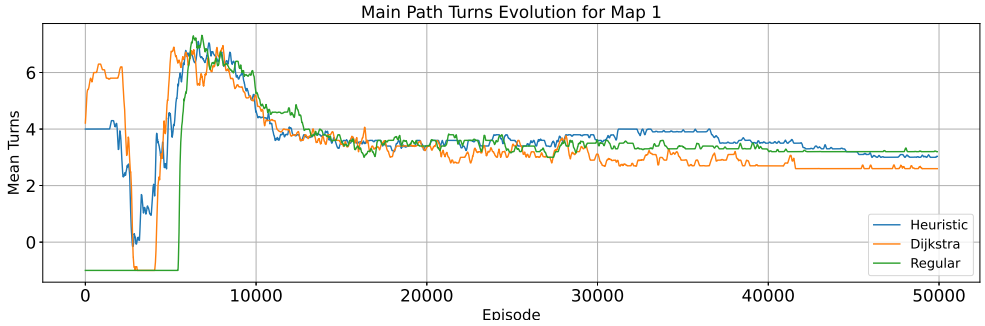
Overall the Dijkstra method outperformed the other two approaches in Map 1, with a higher difference when using initial Q-values of 50. It is possible to notice a more oscillating behavior in this metric, when compared to the previous ones, due to the high step size when changing the Total Turns. The heuristic approach recovered itself quicker in Map 1 with initial Q-Value of 5, but showed similar behavior to the other strategies when using initial Q-value of 50. In the next section the overall final results will be showed and discussed for all Maps, initial Q-values and approaches.

A.4.3 Final Result Metrics

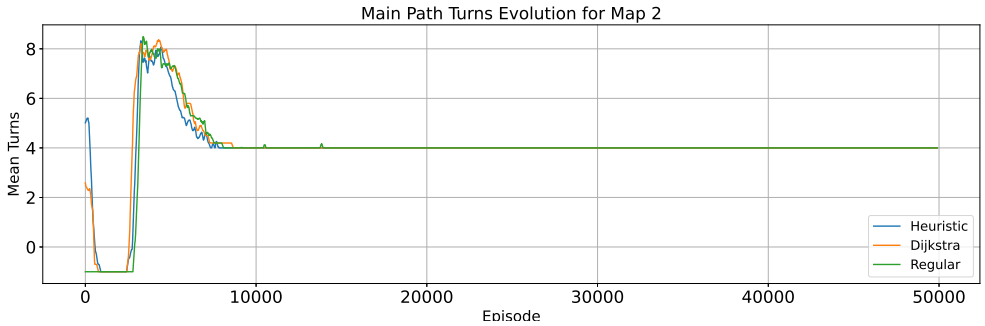
In this section, Tables 17 and 18 summarize the three main metrics for each Map and approach used, as well as the threshold for each of them, which means how many



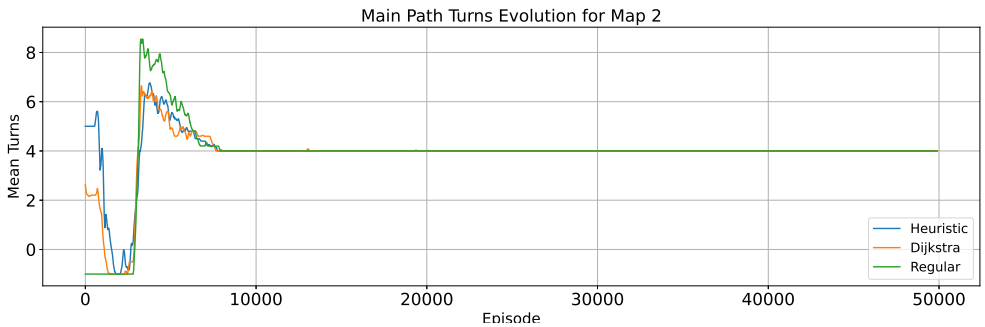
(a) Main Path Total Turns results for Map 1 using 5 as initialization Q-value



(b) Main Path Total Turns results for Map 1 using 50 as initialization Q-value



(c) Main Path Total Turns results for Map 2 using 5 as initialization Q-value



(d) Main Path Total Turns results for Map 2 using 50 as initialization Q-value

Figure 105 – Main Path Total Turns average throughout training for each Map, Initialization Q-value and Method.

episodes it took to achieve and stay in a ±2% range in regards to the best value for that metric. The best performance for each context is highlighed for ease of visualization.

Table 17 – Final Result Metrics by Map and Approach for Initial Q-Value of 5.

Metric	**Map 1**			**Map 2**		
	Heuristic	**Dijkstra**	**Regular**	**Heuristic**	**Dijkstra**	**Regular**
All Paths						
Mean Path Length	**12.34**	12.35	12.45	**8.96**	**8.96**	**8.96**
Mean Obstacle Distance	46.46	**47.97**	47.61	37.95	37.95	**37.96**
Mean Total Turns	2.34	**2.09**	2.41	2.40	2.40	**2.39**
Path Length Threshold	8600	**7800**	12000	8400	**7400**	10600
Obstacle Distance Threshold	37600	**3000**	27000	3600	**2600**	4600
Total Turns Threshold	49600	49200	**39800**	**9200**	9800	11800
Main Path						
Mean Path Length	**22.24**	**22.24**	**22.24**	**19.66**	**19.66**	**19.66**
Mean Obstacle Distance	43.65	**45.48**	44.86	**44.83**	**44.83**	**44.83**
Mean Total Turns	3.00	**2.60**	3.20	**4.00**	**4.00**	**4.00**
Path Length Threshold	9971	**8977**	10683	**5634**	6452	6240
Obstacle Distance Threshold	39048	**8716**	32227	4583	**4325**	5084
Total Turns Threshold	**4994**	49027	49837	20036	13110	**7902**

All Paths - Map 1 - For the All Paths metrics on Map 1 with an initial Q-value of 5, the heuristic approach showed the best performance for the MP metric, closely followed by Dijkstra's initialization. For the other metrics, Dijkstra's outperformed the rest, with a more significant margin on the TT metric. The slightly shorter paths observed with the heuristic initialization compared to Dijkstra's can be attributed to the early training phase, where both had very low success rates and learn to prioritize safety and energy efficiency as well. Regarding convergence, Dijkstra's initialization substantially outperformed the others in the MOD metric and achieved faster convergence for the PL metric. However, for the TT threshold, both heuristic approaches were outperformed by the conventional method.

All Paths - Map 2 - For the All Paths metrics on Map 2 with an initial Q-value of 5, all approaches showed similar results across all metrics. The main difference was that the Dijkstra initialization led to faster convergence in the PL and MOD metrics but lagged behind the heuristic approach in the TT metric. Both outperformed the regular approach in convergence speed.

Main Path - Map 1 - For the Main Path metrics on Map 1 with an initial Q-value of 5, all three approaches achieved the shortest possible path. However, Dijkstra's initialization performed best in the other two metrics. Regarding the threshold for

each metric, Dijkstra outperformed both approaches in the PL and MOD thresholds, with a significant margin in the latter. Meanwhile, the heuristic approach stood out in the TT threshold metric, substantially surpassing the other two strategies.

Main Path - Map 2 - For the Main Path metrics on Map 2 with an initial Q-value of 5, all approaches achieved the same path with the heuristic, dijkstra's and regular approach being the best in convergence for the PL, MOD and TT metrics respectively.

These results show that the initialization methods generally performed as well as or better than the regular approach. The biggest differences appear in the threshold metrics, where the initialized methods showed even greater performance gains over the regular approach. Between the two, Dijkstra's method outperformed the heuristic approach most of the time. The smallest differences were observed in Map 2, which is expected given its simplicity compared to Map 1.

Table 18 – Final Result Metrics by Map and Approach for Initial Q-Value of 50.

Metric	**Map 1**			**Map 2**		
	Heuristic	**Dijkstra**	**Regular**	**Heuristic**	**Dijkstra**	**Regular**
All Paths						
Mean Path Length	**12.36**	12.37	12.45	**8.96**	**8.96**	**8.96**
Mean Obstacle Distance	47.02	47.42	**47.61**	**37.95**	37.96	37.96
Mean Total Turns	**2.21**	**2.21**	2.41	2.40	2.40	**2.39**
Path Length Threshold	11600	**9000**	12000	**8400**	**8400**	10600
Obstacle Distance Threshold	35600	28000	**27000**	3800	**2400**	4600
Total Turns Threshold	48800	49000	**39800**	**10400**	12600	11800
Main Path						
Mean Path Length	**22.24**	**22.24**	**22.24**	**19.66**	**19.66**	**19.66**
Mean Obstacle Distance	44.33	44.80	**44.86**	**44.83**	**44.83**	**44.83**
Mean Total Turns	**2.80**	2.90	3.20	**4.00**	**4.00**	**4.00**
Path Length Threshold	**9009**	9395	10683	**5942**	6532	6240
Obstacle Distance Threshold	37589	34715	**32227**	4541	**4314**	5084
Total Turns Threshold	49980	49993	**49837**	13477	16916	**7902**

All Paths - Map 1 - For the All Paths metrics on Map 1 with an initial Q-value of 50, both initialization methods outperformed the regular approach for MP and TT metrics as well as in the PL threshold with a higher performance offset in the TT threshold with the regular approach achieving stable results 9000 episodes before the heuristic initialization.

All Paths - Map 2 - For the All Paths metrics on Map 2 with an initial Q-value of 50, overall there is a similar performance across the AP metrics, with the initialization

methods achieving quicker convergences when comparing the thresholds for each metric.

Main Path - Map 1 - For the Main Path metrics on Map 1 with an initial Q-value of 50, there are similar results across the MP metrics, with the TT metric being the one with the highest difference, having the heuristic initialization with better performance. In the context of convergence, its possible to see that the regular approach out performed the other two in the MOD and TT thresholds.

Main Path - Map 2 - For the Main Path metrics on Map 2 with an initial Q-value of 50, all approaches achieved the same main path, with the heuristic initialization, dijkstra's initialization and the regular approach converging faster for the PL, MOD and TT metrics, respectively.

When comparing the results with an initial Q-value of 5, it's clear that the initialization methods lose some efficiency. While they still tend to outperform the regular approach, a higher initial Q-value comes with drawbacks. This suggests that strong guidance in the early stages of training may not be as beneficial as a softer initialization, which, despite some drawbacks, leads to better convergence speed.