UNIVERSITY OF
CAMBRIDGE

Utilisation-Aware Roadmap Optimisation for Multi-Agent Path Planning

Yi Lam Kwan

St Edmund's College

This dissertation is submitted on June 2021
for the degree of Master of Philosophy in Advanced Computer Science

Declaration

I, Yi Lam Kwan of St Edmund's College, being a candidate for the M.Phil in Advanced Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed: Yi Lam Kwan
Date: 16^{th} June, 2021

Acknowledgements

I would like to express my appreciation to Dr. Amanda Prorok, Dr. Zhe Liu and Mr. Jan Blumenkamp for their insight, time and support throughout this project.

Abstract

Utilisation-Aware Roadmap Optimisation for Multi-Agent Path Planning

Multi-agent system has important applications ranging from game designs [1], to crowd simulation [2] and logistic applications [3], and the quest of developing a set of collision-free paths for these robot teams has been actively researched. Furthermore, most of the solvers [4, 5, 6, 7] focus on minimising the total distance travelled. However, a sole focus on timing performance may no longer be sufficient in modern days. With limited environmental resources, such as free space, material cost for constructing roads and congestion-associated cost, yet, there is increasing demand for deploying robot fleets, which motivates us to consider the utilisation of space and road. One of the methods is to optimise the environment, in terms of the road style and the moving cost. However, in the literature [8, 9, 10], these concepts were often investigated separately in different research communities and we hypothesis that a combined analysis may also be beneficial.

Therefore, we present a novel utilisation-aware Multi-Agent Path Planning environment optimisation framework that unites the field of environment representation and roadmap optimisation. Our framework features a Voronoi-based topology roadmap, with direction and usage preference for each connection. It also includes an original graph pruning scheme, which was inspired by walkability planning research [11, 12, 13]. A Bayesian Optimisation approach was used to optimise the edge attributes and the edge pruning threshold. Our framework successfully brings 574.78% and 386.85% improvements in roadmap utilisation and space efficiency compared to the grid method, at a cost of 88.09% decrease in flowtime performance. The proposed work is 2.42 and 3.27 time better in resources utilisation than that without any direction or usage optimisations, while sacrificing 12.90% less flowtime performance. We demonstrated the potentials of topology-based roadmaps and the benefits of optimising the directions and usage of roads.

We hope that our work would provide insights and promote the development of utilisation-aware path planning frameworks, given the significance of environmental resources utilisation in the modern era of automation.

Contents

Glossary

ANN Artificial Neural Network.

BFGS Broyden–Fletcher–Goldfarb–Shanno algorithm.

BO Bayesian Optimisation.

CBS Conflict-based Search.

CPDP Centralised Planning for Distributed Plans.

CT Constraint Tree.

DPCP Distributed Planning for Centralised Plans.

DPDP Distributed Planning for Distributed Plans.

GP Gaussian Process.

L-BFGS Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm.

MAPP Multi-Agent Path Planning.

Chapter 1

Introduction

1.1 Motivations

The use of robot teams has become a common solution for various industrial challenges. In 2019, the worldwide installation of the industrial robot was 373, 000, an increase of 69% compared with the 2014 figure [14]. Multi-agent systems have shown the potential to meet the needs of mission-critical, complex distributed systems, where interdependencies and uncertainty play a singular role [15]. In particular, researchers have been actively investigating the Multi-Agent Path Planning problem (MAPP) - to design a set of conflict-free paths for a team of robots to transverse to their destination. This has gathered a wide range of research interests because MAPP has important applications in the fields of GPS [16], game designs [1], robotics [17], logistics [18] and crowd simulation [2]. For instance, robot teams have been deployed for last-mile deliveries [3], and multi-agent planning has proven to be useful in providing emergency evacuation plans for stadium [19]. These applications require efficient and reliable movements, hence the need for a high-performance MAPP framework.

Furthermore, with the advancement in collision avoidance path planning algorithms and that adopting robot teams for real-world logistic applications becomes a reality, there are new calls for researchers to investigate the utilisation of roadmaps and overall space efficiency. There are multiple incentives to do so. Firstly, transportation is an example of a multi-agent system and one of the major challenges confronting most metropolises is traffic congestion. In 2000, the "congestion bill" in the United States alone was USD 67.5 billion [20], which consists of 3.6 billion hours of delay plus 5.7 billion gallons of gas according to Texas Transportation Institute. By encouraging agents to 'spread out' when traversing in the map, it minimises the number of likely intersection points, thereby, reduces congestion and its corresponding cost. Secondly, designing utilisation-aware roadmaps would also be useful in the case of planning for evacuation and disaster response [19]. Thirdly, to take an example from the routing community, bandwidth (roads) are valuable resources.

Resources utilisation is hence an important aspect of the system design. On top of the significance of roadmap utilisation, space efficiency has been another concern. In the context of traffic navigation, roads are costly to build [21]. Furthermore, as urban areas become more congested, city designers often aim to minimise the number of roads while maximising traffic throughput. Therefore, it is in our favour to also constrain the size of the overall roadmap, such that it does not take up all the available free space and ensures that space is being utilised efficiently.

We recognise the importance of developing a utilisation-aware MAPP framework. However, challenges remain. Although research on MAPP has been flourishing in the past couple of years [4, 5, 6, 7], which enable us to develop collision-free plans for multiple robots with completeness and optimality guarantee. However, most of these MAPP literatures [4, 5, 6, 7] have focused on optimising the time efficiency of the solutions, i.e. to minimise the total distance travelled by the agents. Hence, there is a gap in the literature in terms of MAPP solutions that considered roadmap utilisation and space efficiency. Furthermore, among the handful of literature that considered environmental resources utilisation, most of them have only considered dense grid-based roadmaps [9, 22, 23, 10]. Although it is often more straightforward to parameterise the environment as a grid, the scheme is subject to memory inefficiency and a lack of flexibility in embedding other objectives into the roadmap. Hence, we would also like to look into more sparse topology-based environment representation, which would allow us to customise the moving cost of edges to achieve our utilisation objectives. The concept of optimising edges weights is not unheard of and is often investigated alongside the Travelling Salesman Problem [24]. For instances, some has considered adding directionality to the roads [9, 25], while others have considered pruning extra edges [26]. Promising results have been shown and we are interested in combining some of these ideas into our framework.

1.2 Research Objectives

Therefore, we would like to propose a novel utilisation-aware environment optimisation framework, which includes an environment parameterisation scheme, an optimiser and a MAPP solver. We would like to optimise the environment in terms of the road style, moving costs of edges and the presences of edges, such that resource utilisation could be promoted while maintaining reasonable timing performance. The primary hypothesis of the project is as follows:

Environmental resources utilisation in a multi-agent system can be improved by using topology-based environment parameterisation schemes that include directions and usages

specifications.

To validate the hypothesis, we have defined four research objectives:

1. Investigate the impact of various environment parameterisation schemes on the system performance, which would be quantified in terms of environmental resources utilisation and timing efficiency.

2. Determine an appropriate tool for the intended roadmap optimisation problem by exploring various optimisation approaches.

3. Investigate the impact of adding direction preferences to roadmaps on system performance.

4. Investigate the impact of roadmap pruning on system performance.

1.3 Contributions

Our research lies in the field of Environment Representation, Roadmap Optimisation and Utilisation-aware MAPP, it contributes in several ways to each domain.

- We presented a novel utilisation-aware environment optimisation framework for MAPP, which bridge the gaps in the literature as discussed in Section 2.4.

- We demonstrated the advantages and trade-offs of using a topology-based roadmap relative to the conventional grid-based roadmap. Our scheme successfully brings 574.78% and 386.85% improvements in roadmap utilisation and space efficiency compared to the grid method, at a cost of 88.09% decrease in flowtime performance.

- We show the gain of adding direction and usage preferences to edges, the utilisation and space efficiency attained is 2.42 and 3.27 times better than that of using a basic Voronoi roadmap while sacrificing 12.90% less flowtime performance.

- We propose the use of Gaussian Process (GP) with Bayesian Optimisation (BO) techniques for roadmap optimisation, as an alternative to using Artificial Neural Network (ANN) with numerical optimisers, where BO successfully doubled the performance of ANN with only one-forty of the samples.

- We test our scheme under environments of various shapes and obstacles density. We show that promoting roadmap utilisation helps resolve congestion in narrow openings environment.

1.4 Thesis Outline

The thesis is structured as follows:

- **Chapter 2** reviews the related work on path planning and environment optimisation.
- **Chapter 3** discusses the relevant background concepts, such as conventional environment parameterisation techniques in the robotics community and various optimisation approach.
- **Chapter 4** introduces the problem formulation, the experiment set-up, as well as the implementation details of the proposed scheme.
- **Chapter 5** investigates the performance of various environment parameterisation scheme under different environment and obstacle density. On top of that, we also evaluate whether the submodules of the proposed framework are behaving as expected, hence, we include a series of experiments on optimisers and the local cost function.
- **Chapter 6** summarises the research, provides a critical analysis of the proposed framework and presents directions for future work.

Chapter 2

Related Work

In this chapter, we will first give an overview of the MAPP literature and in general how the utilisation problem has been approached. Afterwards, we will review related work for Environment Optimisation, which includes choices of road styles and various roadmap optimisation techniques. Finally, we will also reiterate the research gap and discuss how our framework attempts to fill it.

2.1 Multi-Agent Path Planner

MAPP usually takes two main approaches: centralised approaches that utilised a global planner to compute all the path simultaneously; and decentralised approach that compute paths individually. These can be further divided into three main categories: Distributed Planning for Centralised Plans (DPCP), which usually involves agents communicating their objectives with one another [5, 6]; Distributed Planning for Distributed Plans (DPDP), where agents come up with their plans and handle collisions when they encountered them; Centralised Planning for Distributed Plans (CPDP), where there exists a centralised planner that generates path plans for each agent.

An example of DPCP would be D. D. Corkill research on hierarchical planning [5], where each agent generates their representation of the problem and plans based on that. A global plan is then produced by combining the sub-plans. The advancement in sensor technology realised inter-vehicle communication [6], hence, further enabled these DPCP methods. Moreover, when agents' state-graphs are unsynchronised or when the agents have incompatible goals, these methods cannot function properly. The complex planning process and communication overhead also render DPCP method less suited for our application.

Secondly, a common example of DPDP methods is a sampling-based method. For instance, S. Karaman and E. Frazzoli (2011) [27] proposed to use Rapidly exploring Random Trees (RRT). They first sampled a random position and connected it to the nearest existing tree node given that it satisfied the collision check. The tree would tend

to expand to unexplored areas due to random sampling and would stop once the goal location is found. Moreover, sampling-based methods often yield sub-optimal solutions and lose completeness.

Finally, CPDP has been one of the most popular approaches. They are often graph-search-based methods, which help us guarantee completeness. An example would be Conflict-based Search (CBS) proposed by G. Sharon et al. (2014) [7]. It stores path conflicts between individual agents and the cost of resolving them in a tree structure, hence, a minimum cost solution could be found by performing a simple tree search. Although it is generally more computational expensive than that of DPDP methods, it provides optimality guarantees, which will work well on our small dataset. Furthermore, the graph-search-based method also allows us to change the system objectives easily, as there are a lot of flexibilities in customising the graph. This property is another pull factor for adopting these types of planners in our project.

We also notice that the well-known path planners listed above [4, 7] mostly focus on optimising the timing efficiency of the solutions. However, road and space resources could be scarce and costly as mentioned in Section 1.1, hence, it is of paramount importance to consider environmental resources utilisation as well. In the next section, we will provide an in-depth discussion on path planning frameworks that considered utilisation.

2.2 Utilisation-Aware Framework

Utilisation can generally be improved by adjusting agents' paths based on dynamic road situation, or by adopting a pre-optimised roadmap that was initialised with previous congestion data or by changing the environment representation framework.

Gaining information related to real-time road condition may assist drivers to choose alternate routes to avoid traffic jam. Taking insights from our daily life, the congestion problem could be approached with a communication-based method. To give an example, P. Desai et al. (2013) [6] proposed to resolve congestion through inter-vehicular communication. Agents exchanged information about their initial planned routes, availability of alternate route(s) and their compliance history with their neighbours. Based on this information, agents reached cooperative route-allocation decisions and paths were alerted to maximise utility. However, such methods are often associated with communication overhead, as well as the risk of coming across non-equipped vehicles. The heightened system complexity also deterred us from considering this approach.

For the roadmap optimisation approach, there are various common heuristics for setting up the roadmap based on historical congestion data. For instance, most research has opted for gradient methods to optimise the weight of the edges in the roadmap [10, 23, 9]. This effectively impacts the path selection process as agents would select the lowest cost edges

when transversing in the roadmap. The data-driven approach requires less computation power for individual agents and the overall system complexity is also lower. This motivated us to consider roadmap optimisation in our framework, it also gave us some inspirations on how to formulate the weights in our roadmap.

Moreover, researchers also attempted to generate a framework that lies in the middle of the spectrum. Analysing dynamic road situation for path selection relies on current information, whereas the roadmap based approach requires a lot of data. Therefore, C. Street et al. (2020) [22] proposed a utilisation-aware framework that plans based on the information of already routed path. However, sequential planning models are highly sensitive to the order of the tasks. Currently, it will not be a critical issue because each roadmap generated is only designed for a fixed set of locations. Moreover, when we consider generalising our roadmap generation framework for any locations pairs with learning-based methods, we will not be able to generate a permutation invariance dataset for a problem that is supposed to be of such nature. Hence, a sequential planner was not being considered in our work.

Last but not least, some literature has also attempted to improve utilisation by changing the road style. A. Kleiner et al. (2011) [25] proposed to use medial-axes to form a skeleton roadmap and optimised for flowtime and Utilisation by changing the edges' traverse cost. It influenced us investigating the impact of environment representation and whether it can provide performance gain in terms of roadmap utilisation and space efficiency while maintaining flowtime by changing the representation of the environment.

Overall, environment optimisation (road style and roadmap optimisation) seems to be a promising approach for constructing a utilisation-aware MAPP framework and we will look into details of it in the next section.

2.3 Environment Optimisation

Our review on environment optimisation research could be categorised in terms of road style and roadmap optimisation.

2.3.1 Road Style

The first step in roadmap generation is to transform the traversable space into a discretised representation. Then, environment representation techniques can be categorised into cell decomposition, sampling-based and skeletonisation methods.

One of the most popular approaches is to use Approximate Cell Decomposition to form 4/8-connected regular grids, also known as pixel grid. Most MAPP literature has utilised this method [4, 5, 6, 7, 9]. In this project, we have also used it as our baseline. However, this approach generates a large amount of cells, which hinders memory efficiency

and significantly increases the computational load for complex path planner. Another improvement for approximate cell decomposition is by using quad-tree decomposition [28, 29]. It is a recursive algorithm, where cells are being subdivided until each cell lies either completely in free space or completely in the obstacle region. However, its efficiency would decline significantly as the workspace's dimension increases [30].

Alternatively, sampling-based approach is another popular method for roadmap generation. For instance, C. Henkel and M. Toussaint (2020) [9] proposed to generate an irregular grid by randomly sample N vertices from the free sample and construct the edges with Delaunay Triangulation. Furthermore, there are also scenarios where sampling methods could be combined with probabilistic roadmap planner to plan for paths while constructing roadmap. RRT [27] is an example of that. Moreover, it has been shown that its performance declined when the scenes involved narrow passages and openings. This is because uniform sampling encourages the sampler to explored large open regions and there would be fewer samples from tight passages. Although the low complexity is an attractive factor, its uniform sampling property discourages us from using it because tight passages would be one of the primary datasets for us to test our utilisation-aware framework. We envision that a our framework could bring more performance gain in these crowded environments relative to a less-congested environment.

The above methods tend to generate a dense roadmap that only allows 1 robot to traverse at the edge at the same time, moreover, there are other representations that may allow us to represent intersection and space more efficiently. Therefore, topology representations that retain the salient topology of the traversable space have also been considered. The visibility graph proposed by Lozano-Pérez and Wesley (1979) [31] is the classical method for this problem, where the vertices of the obstacles are being used as graph nodes, and edges are formed by joining mutually visible nodes. Moreover, it usually takes $O(n^2)$ computation time to generate and the skeleton would consist of $O(n^2)$ edges, where n corresponds to the number of features [32]. A less computational-intensive version of that would be Reduced Visibility Graph because it only retains convex vertices for the graph construction [33]. However, the main disadvantage of the visibility graph is that it clips trajectories to the edge of the obstacles, which is undesirable in most cases and our applications. In particular, our environments involve a lot of narrow passages and concave regions, and clearance is preferred.

Another candidate for the skeletonisation approach would be the near-site Voronoi diagram. The diagram is formed by joining the set of points that are equidistant from two or more object features. In the original research of L. Dirichlet [34], the features were a set of points sites. Over the years, researchers have also considered various generalisations of Voronoi diagrams, namely: Voronoi diagrams in higher-dimensional spaces [35], Voronoi diagrams in spaces endowed with different distance functions [36], Voronoi diagrams

of various geometric objects (Variable-Radius Voronoi Diagram) [37, 38]. The Voronoi diagram is attractive in three respects: it allows the robot to stay away from the obstacles, there are only $O(n)$ edges in the Voronoi diagram, and its computational time is only $\Omega(n \log n)$, where n is the number of features [39]. The space efficiency of salient point-based roadmap compared with grid-based methods is one of the major reasons why it is preferred over cell decomposition and sampling-based methods because it is one of the project objectives to optimise space efficiency. Moreover, the additional clearance and lower computational requirement motivate us to choose Voronoi diagram over Visibility Graph.

Additionally, one of the benefits of topology representation is that it allows us to consider more than the length of the edges during path selection, for instance, we can also consider the capacity of edges. The concept of path planning inside corridors has also been introduced over the years [40, 41, 42]. R. Geraerts (2010) [8] proposed a data structure for these corridors, referred to as the Explicit Corridor Map. Given a bounded 2D environment with polygonal obstacles, the map was defined as the medial axis of the free space annotated with nearest-obstacle information, and useful in computing smooth and short paths for disk-shaped agents [2]. Our work is similar to theirs in a way that we both used Voronoi diagram as the base skeleton graph and utilised corridor for path planning, however, we have modified the scheme such that it works with the discretised environment instead of the continuous space and we have taken a more conservative approach in determining edges' clearance from the obstacles.

2.3.2 Roadmap Optimisation

For the MAPP problem under pre-defined task location pairs, the most common method of optimising the weight in the roadmap is to use gradient-based approach [9, 10, 23]. For example, C. Henkel and M. Toussaint (2020) [9] considered optimising the position of vertices and the moving cost of each direction on the edges. Our framework takes inspirations from their research, we also added directional preference to edges to optimise the moving cost in each direction. However, our work is different in several ways. Firstly, they used random sampling with Delaunay Triangulation as the roadmap, which resulted in a dense roadmap and showed a lack of consideration of resources utilisation, whereas we used a sparse Voronoi diagram as a skeleton and specified road capacity by adding dimensions to edges. Furthermore, they only considered the single-robot case by assuming the probability of collision of points agents that follow directed paths with constant velocity is zero. Although this assumption allowed the author to alleviate the constraints of the predefined task, the relaxation resulted in a low success rate of solving the MAPP problem. Therefore, in our framework, we have utilised a centralised planner instead to plan for multiple robots simultaneously. Lastly, they used piece-wise gradient in Stochastic Gradient

Descent for the optimisation process, moreover, Stochastic Gradient Descent often requires a high number of sample for convergence, which is less preferable for computationally intensive MAPP solver. Hence, in our proposed schemes, we consider optimisers that have higher sampling efficiency.

As an alternative, one can utilise the joint C-space optimisation approach, which considers the consequences of the joint actions of the agents instead of the impact of the agent separately. An example would be to utilise a simulator with a numerical optimiser. Recently, in the field of parameters search, the Neural Architecture Search communities presented several promising proposals on utilising GP as the simulator of a black-box machine learning model and used BO to find a set of suitable hyperparameters [43, 44]. This technique is well suited for expensive black-box function and known for its sampling efficiency. Our roadmap optimisation problem required constant evaluations on the MAPP solver, which is expensive to enquire. Thus, we are keen on testing this efficient sampling approach on our framework. On top of that, another more conventional simulator would be the neural network. Actually, ANN and GP share some common applications and are often being compared against each other [45]. Therefore, we are also keen on exploring their differences in performance.

Finally, instead of estimations, A. Kleiner et al. (2011) [25] adopted an analytic solution for the roadmap optimisation problem. In their work, they presented a Voronoi-based roadmap and optimised the moving cost of edges in each direction based on capacity and distance. Although we both used the Voronoi-based roadmap, their roadmap was clipped to a grid while ours is not, hence, we have more flexibility in representing the environment. Furthermore, although we both considered capacity in the moving cost and measures roadmap utilisation, our framework also considered space efficiency. In our work, we also estimated the usage of each road and pruned away edges with low estimated usage, which allow us to generate an even smaller and space-efficient roadmap relative to their work. Their optimisation problem was formulated such that it could be solved by Integer Linear Programming, whereas our proposed scheme uses BO. In fact, both of our optimisers suffers from scalability problem, hence, there are rooms for future investigations.

2.4 Research Gap

Upon reviewing various related work, we would like to highlight the research gap again and the improvements brought forward by our proposed schemes.

First of all, there has been active research on high-performance MAPP algorithms [4, 5, 6, 7] with some capable of generating optimum flowtime solutions with guarantees on optimality and completeness. However, a sole focus on timing efficiency may no longer be sufficient. As environmental resources become more scarce alongside the growth of the

city, it is equally important to focus on environmental resources utilisation.

There is a handful of literature relating to utilisation-aware MAPP framework scattered among the field of robotics path planning, communication flow and game design. Some proposed inter-vehicle communication and sequentially planning, moreover, these approaches are deemed as less suitable compared with environment optimisation approaches due to the substantial increase in complexity and the lack of prospects for future framework generalisation.

Then, by environment optimisation, we referred to the optimisation of road style and the roadmap. Moreover, we noticed that these fields were usually investigated separately by researchers in different fields. The game design community has put much more emphasis on environment representation and road style. We have gained a lot of inspiration on how to construct a more efficient representation of space with the Voronoi diagram and Explicit Corridor Map. However, most of the work related to environment representation did not further extend their analysis to roadmap optimisation. And for literature that considered roadmap optimisation, C. Henkel and M. Toussaint (2020) [9] considered optimising for direction but did not attempt to optimised the environment representation or resources utilisation, T. Alpcan and T. Basar (2002) [10] considered utilisation but it focuses on communication networks environment.

Therefore, in this project, we would like to bridge the gap between environment representation and roadmap optimisation and provide a combined analysis that better tackle the environment optimisation problem. We would like to present an environment optimisation framework that promotes resources utilisation. The framework will consist of an environment parameterisation scheme, an optimiser and a MAPP solver and its details will be introduced in later Sections. In fact, A. Kleiner et al. (2011) [25] also have a similar idea, they considered optimising edges' directions to promote utilisation and used a sparse topology roadmap. Moreover, they did not consider one of the most useful side product of adopting sparse roadmap - space efficiency and roadmap pruning, which is also an important aspect of environmental resources utilisation. Furthermore, our work also employed a more flexible base roadmap than one that has been clipped to the grid.

Chapter 3

Background

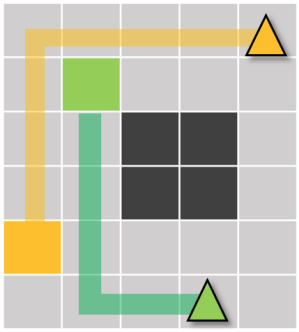
Upon introducing the motivations and related work of this project, we would like to explain some of the high-level concepts and terminologies that are essential in understanding our work. We will first introduce the definition of the MAPP Problem and formalised the notion of utilisation. Then, we will explain some concepts related to environment parameterisation within the field of robotics, including the robot workspace and methodologies for discretising the continuous space. These form the foundations of our framework.

3.1 Multi-Agent Path Planning Problem

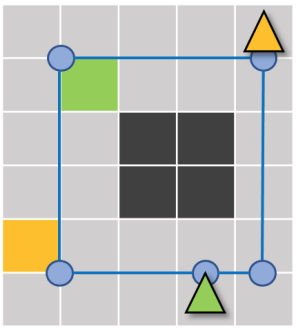
The MAPP problem deals with the question of how a group of agents can travel to their targets without colliding with each other and obstacles. In general, path planning approaches include grid-search based, sampling-based and graph-search based methods. Moreover, due to reasons discussed in Section 2.1, we will focus on discussing the graph-search based methods. Under the graph-search based category, path planning usually comprises of two steps: graph generation and a path planning algorithm.

For the graph generation process, graph G could be generated using skeletonization [46] or cell decomposition [47] techniques and would be discussed in details in Chapter 3.3. The graph, also referred to as roadmap, represents the traversable space in the environment. The graph is usually defined as $G = (V, E)$, where V are vertices that map to coordinates in the continuous environment and E refers to edges that connected the vertices and are within line of sight from one another, an example has been shown in Figure 3.1b, 3.1c. Each edges E may have associated weights w that represents the costs of travel through a certain connection. If the travelling cost depend on the direction, G would be a bi-directional graph similar to one in Figure 3.1d. With differential travel cost for edges in opposite direction, it is possible to incorporate other objectives for the path planning. In our proposed scheme, G is bi-directional.

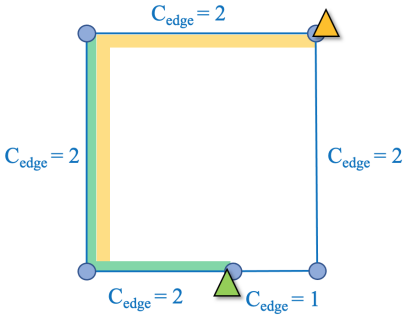
For the algorithm, the input to a k-agent path planning problem would be a tuple



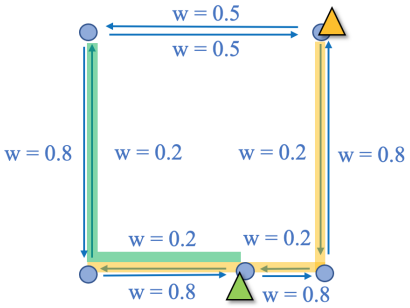
(a) Grid Roadmap



(b) Topology Graph on Grid



(c) Graph-based Roadmap



(d) Bi-Directional-Graph-based Roadmap

Figure 3.1: The figure illustrated the concept of topology-based roadmap. They were based on graphs or bi-directional graphs. The triangles, C_{node} and w refer to the agents, the capacities and weights of edges.

$< G, s, t >$, where $s : [1, ..., k] \rightarrow V$ maps an agent to a source vertex and $t : [1, ..., k] \rightarrow V$ maps an agent to a target vertex [4]. Time is assumed to be discretised. At each time step, agents would be situated on graph vertices. They would then perform actions: wait or traverse to connecting vertices. Hence, a plan π refers to a sequence of actions that lead an agent to its goal, whereas a joint plan (the solution) corresponds to a set of plans. We denoted the location of agent i after x actions in its plan π as $\pi_i[x]$. The goal of the search algorithm is to devised the joint plan given $< G, s, t >$ and would be discussed in details in Chapter 3.4.

In addition, some common design decisions that divided the MAPP problems into different types: the presence of dynamic or static obstacles and agent behaviour at target. In most cases and including this work, we are only concerned with the static environment, hence, agents only have to consider obstructions from the stationary environment and other agents. Dynamic obstacles are usually reactively tackled during execution [48, 49]. Furthermore, upon arriving at the target, the agent could continue to stay or disappear. Although staying would cause potential node conflicts when other agents' attempt to pass through the node, it is a realistic approximation based on real-world scenarios. Most prior work and our work followed this assumption, notwithstanding, there are recent works [50] that considered otherwise.

3.2 Utilisation

We would like to provide clarification on the term *utilisation*, which is the focus of this report. In this piece of work, utilisation refers to making effective use of the resources, where resources include the defined roadmap and the free space. The two metrics that we will use to describe the level of utilisation are road utilisation and space efficiency.

$$Roadmap\ Utilisation = \frac{\text{Area traversed by the agents}}{\text{Total area of the roadmap}} \tag{3.1}$$

$$Space\ Efficiency = \frac{\text{Total area of the roadmap}}{\text{Total area of Free Space}} \tag{3.2}$$

Details of their computational methods will be introduced in Section 4.2.1.

3.3 Environment Parameterisation

3.3.1 Robot Workspace

The continuous environment that the robots operate in is the robot workspace $W = \mathbb{R}^N$. In the classic MAPP problem, W would be further abstracted into the Configuration Space (C-Space), which contains all valid configurations of the robot and space occupied by obstacles. The motivation is to simplify the problem such that the robot could be represented as a point in the C-Space. Finally, we will also introduce the details of the MAPP solver used, as well as the background of various optimisation schemes.

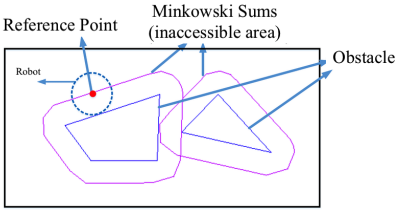


Figure 3.2: The figure shows how the configuration space could be obtained by computing the Minkowski sum. [51]

In particular, the obstacle region in the C-Space C_{obs} could be calculated by the Minkowski sum between the obstacle and the robot. The process is also known as dilution. Let A and B be the position vector of the robot and the obstacle in the Euclidean space:

$$C_{obs} = A \oplus B = \{\mathbf{a} + \mathbf{b} | \mathbf{a} \in A, \mathbf{b} \in B\} \tag{3.3}$$

Figure 3.2 illustrated the formation of C_{obs}. As long as the robot's reference point stays outside dilated area, collisions will not occur.

3.3.2 Cell Decomposition

The continuous C-space could be discretised into the regular and irregular grid through cell decomposition or skeletonization for path planning purposes.

Under the cell decomposition approach, we would subdivide the C-space into smaller regions called cells. There are several cell decomposition techniques, such as Exact Cell Decomposition and Approximate Cell Decomposition, in our work, the regular square grid was formed by Approximate Cell Decomposition. Despite pixel grid, cells could also be triangular, octile, hexagonal, etc. Regular grids could then be formed from tessellations of the aforementioned polygons.

3.3.3 Skeletonisation

With the skeletonisation approach, we aimed to capture the salient topology of the traversable space. The generated irregular grid is usually referred to as a waypoint graph. The nodes in the graph are called waypoints and each of them refers to specific locations in the map. An edge between two waypoints denotes that agents can transverse it without colliding with obstacles. Visibility graph and Voronoi diagram are some of the common approaches for constructing a waypoint graph.

3.3.3.1 Visibility Graph

In computational geometry, a visibility graph contains a set of inter-visible locations. Vertices of the polygonal obstacle are used to form the vertices of the visibility graph. Valid edges are defined as line segments that connect the vertices without intersecting the interior of any obstacle.

3.3.3.2 Voronoi Diagram

Voronoi diagrams of points in the Euclidean plane were first established one and a half centuries ago by Lejeune Dirichlet when investigating quadratic forms [34]. The concept of the Voronoi diagram refers to partitioning a plane into regions close to a given set of objects. In its simplest formation, these objects would be a set of points, also called sites. The boundaries of the regions are defined by perpendicular bisectors of segments joining each pair of sites. Furthermore, any point on the bisector is equidistant from the two sites that it bisects.

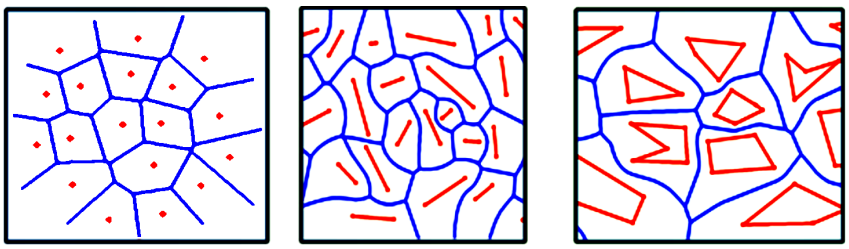


Figure 3.3: The figure shows the Voronoi diagram of points, lines and polygon. Red and blue colours correspond to sites and Voronoi edges respectively. The figure was reproduced from [52].

Definition 3.3.1 (Voronoi Diagram of Points Sites)**.** *Let S be set of sites s and p be points equidistant to s. The Euclidean distance function would be $d(p, s)$. Voronoi diagram of Points Sites is a partition of space into regions $VR(s)$ s.t. for all p in $VR(s)$:*

$$d(p, s) < d(p, t)$$

for $t \in \{S \sim \{s\}\}$.

In our path planning application, a waypoint graph could be constructed with a Voronoi diagram of polygons, which are medial axis in the free space. Its definition is as stated in Theorem 3.3.2.

Definition 3.3.2 (Voronoi Diagram of Polygon Sites)**.** *Let S be set of sites s. Edges of Voronoi diagram of Polygon Sites are the medial axis of the region.*

This medial axis based graph forms the base roadmap of our proposed scheme.

3.3.4 Random Sampling Techniques

Alternatively, a sampling-based approach could also be used to form an irregular grid. Vertices are first found by randomly sampling points in C_{free} (free space in C-space). Edges are then constructed using Delaunay Triangulation to maximise the angular separation between edges, where Delaunay Triangulation is a technique for maximising the minimum angle of all the angles of the triangles in the triangulation.

3.4 Path Planning Algorithms

In this project, we have utilised the CBS algorithm for path planning.

3.4.1 $\mathcal{A}*$ Search Algorithm

$\mathcal{A}*$ is a graph search algorithm commonly used for determining the least-cost path for a agent to traverse from a starting node to a goal node. The planner maintains a tree of paths originating at the start node and extending these paths one edge at a time until the end node is found. At each iteration, $\mathcal{A}*$ decides which paths to explore based on an aggregated cost $f(n)$. Let n be the next node on the path,

$$f(n) = g(n) + h(n) \tag{3.4}$$

where $g(n)$ is the cost of the path from the start node to n and $h(n)$ is the estimated cost from n to the goal. $h(n)$ could be any heuristic function, as long as it generates an underestimate of the actual cost to goal, $\mathcal{A}*$ is guaranteed to return a least-cost path. Therefore, Euclidean distance has often been used as the heuristic function.

3.4.2 Conflict-based Search

CBS [7] is a optimal multi-agent pathfinding algorithm for minimising Flowtime (total travelled distance) or Makespan (time taken for all the robots to reach their destinations). It is a two-level algorithm and resolves conflicts based on a Constraint Tree (CT). First, the algorithm would solve for each agent individually. Then, at a high level, for each conflict encountered, it will split the conflict into separate constraints and create CT nodes for each of the constraint. Then, at the low level, for each node, fast single-agent searches are performed to find a solution that satisfied the constraint. In addition, $\mathcal{A}*$ search algorithm is often utilised for single-agent searches. Finally, a minimum cost solution could be found by searching the CT. In this project, we have made some modifications such that CBS will work with edges with capacity.

3.5 Optimiser

The optimisation problem will be discussed in full length in Chapter 4.2. The purpose of the optimiser is to find the direction and usage of edges that gives the minimum global cost (the aggregated cost of flowtime, utilisation and penalty). In general, it consists of 2 parts, one to approximate the distribution of the function, one for finding the minimum cost sample based on the approximate distribution. In this work, two optimisers were investigated: GP with BO and ANN with Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS).

3.5.1 Gaussian Process

GP provide a framework for us to model distribution of functions f. Let $\mathcal{X}$ be the input domain, a GP can be denoted by a mean function $\mu : \mathcal{X} \to \mathbb{R}$ and a covariance kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. For a N-samples dataset $D = \{\mathbf{x}_i, y_i\}_{i=0}^{N}$, the posterior of a input point $\mathbf{x}_*$ is:

$$
\begin{aligned}
p(f_*|\mathbf{x}_*, \{\mathbf{x}_i, y_i\}_{i=0}^{N}) = \mathcal{N}(\mu_{\mathbf{x}_*|\mathbf{x}}, k_{\mathbf{x}_*|\mathbf{x}}), \text{ where,} \\
\mu_{\mathbf{x}_*|\mathbf{x}} = k(\mathbf{x}_*, \mathbf{x})\Big(k(\mathbf{x}, \mathbf{x}) + \beta^{-1}\mathbf{I}\Big)^{-1}\mathbf{y} \\
k_{\mathbf{x}_*|\mathbf{x}} = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{x})\Big(k(\mathbf{x}, \mathbf{x}) + \beta^{-1}\mathbf{I}\Big)^{-1}k(\mathbf{x}, \mathbf{x}_*) \ ,
\end{aligned}
$$

where $\mathbf{I}$ and $\mathbf{y}$ are the identity matrix and output values in the dataset respectively. The covariance kernel k acts as a handle for us to input our domain knowledge about f into the process. Common kernel functions includes Radial Basis Function (RBF) kernel or Matern kernel [53].

3.5.2 Bayesian Optimisation

BO is a global optimisation technique for functions that are expensive to evaluate and one of the main advantages of BO is it does not assume any functional forms.

Upon initialising the GP with several initial random samples obtained from the MAPP simulator, the uncertainty estimates of the GP over the input domain for f were used to query the simulator at specific inputs according to the experimental objective. Our experiment objective would then affect our choice of Acquisition function, which is the function used to decide which points to sample. In the case of minimising uncertainty over the function f, Integrated Variance Reduction (IVR) could be used. Moreover, our objective is to determine input variables that minimise the global cost. Towards this end, the Expected Improvement (EI) acquisition function would be more suitable.

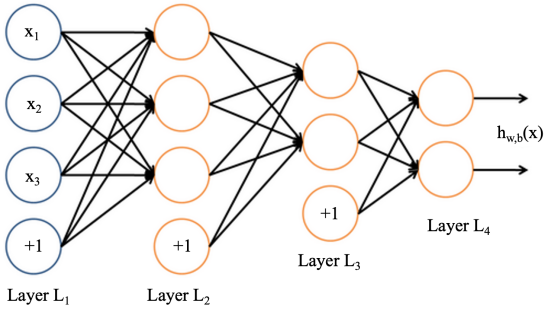
$$
a_{EI}(\mathbf{x}) = \int_{-\infty}^{f(\mathbf{x}^*)} (f(\mathbf{x}^*) - f(\mathbf{x}))\mathcal{N}(f|\mu, k)\mathrm{d}f \ , \tag{3.5}
$$

where $f(\mathbf{x}^*)$ is our current best estimate. The acquisition function was defined as the weighted sum of the difference between the current best estimate of the function and all possible function outputs given input $\mathbf{x}$ weighted by our current belief over f. Therefore, at every iteration, f was evaluated at input $\mathbf{x}$ which maximises EI, then, refit the GP to the new sampled point. The process repeats until the number of iterations equal to the number of sample specified.

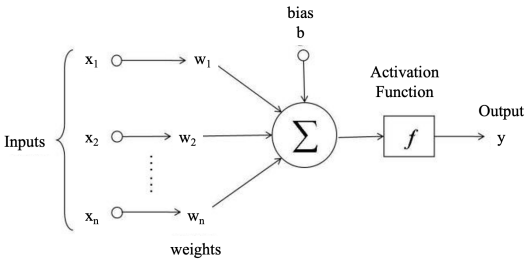
BO is often used in hyper-parameters search of Machine Learning model. However,

despite being a powerful technique in searching for the global optimum of expensive black-box functions, it has been recommended that the number of parameters to be optimised should not exceed 20[54] for meaningful exploration. Therefore, we have emphasised the importance of reducing the search space in Section 4.2 and have developed some mitigation measures.

3.5.3 Artificial Neural Network



(a) Representation of a neural network. [55]



(b) Operations within a neuron. [55]

Artificial Neural Network (ANN) is an information-processing paradigm inspired by the way biological nervous systems works. An ANN consists of nodes (neurons) and edges (synapses). They are organised in a layer fashion as shown in Figure 3.4a. Each synapse can transmit a signal to other neurons. An artificial neuron would then combined signals from all the connected neurons multiplied by their respective weight. The sum of these signals alongside a bias term would then be passed to a non-linear function. The function output would then be transmitted to other neurons. Figure 3.4b gave an illustration of this process. Given a dataset, the set of weights and biases in the network would be adjusted to minimising the ANN output with the desired output.

Once again, ANN could be used to model the distribution of a black-box function f with samples obtained from the MAPP simulator. We wll also compare the performance of this optimisation approach relative to BO in this project.

3.5.4 Limited-memory Broyden–Fletcher–Goldfarb–Shanno Algorithm

Numerical optimiser usually starts with an initial guess for x when minimising an objective function $f(x)$. A sequence of improving approximate of x would then be generated until it has fulfilled a termination criteria. In each iteration, the algorithm determines a direction of travel p_k such that the next sample point x_{k+1} equals to:

$$x_{k+1} = x_k + a_k p_k \tag{3.6}$$

where a_k is the step length. The value a_k and p_k are computed differently under difference optimisation schemes. Under the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method, a_k is chosen dynamically based on the Wolfe Condition [56]. The direction of travel p_k is computed based on an approximation of the inverse of the Hessian matrix. Hessian is a square matrix of second-order partial derivatives of the objective function, which described the local curvatures. Also, the approximation of the inverse of the Hessian matrix is determined based on the position difference $(x_{k+1} - x_k)$ and the gradient difference $(\Delta f_{k+1} - \Delta f_k)$ in the iteration. L-BFGS modified BFGS such that the size of Hessian approximations could be stored in a $m \times n$ matrix instead of $n \times n$, where $m << n$ and n is the size of x.

Chapter 4

Methodology

With understandings of the background knowledge and the research gap, in this chapter, we would like to introduce our proposed methods. First of all, the overview of the environment optimisation framework will be presented. Following that, the formal definitions of our roadmap optimisation problem will be introduced alongside the performance metrics used in this project. Finally, we will also give details related to the implementation of the environment parameterisation schemes, the optimisers and the path planner, which are the pillars of our framework.

4.1 Framework Overview

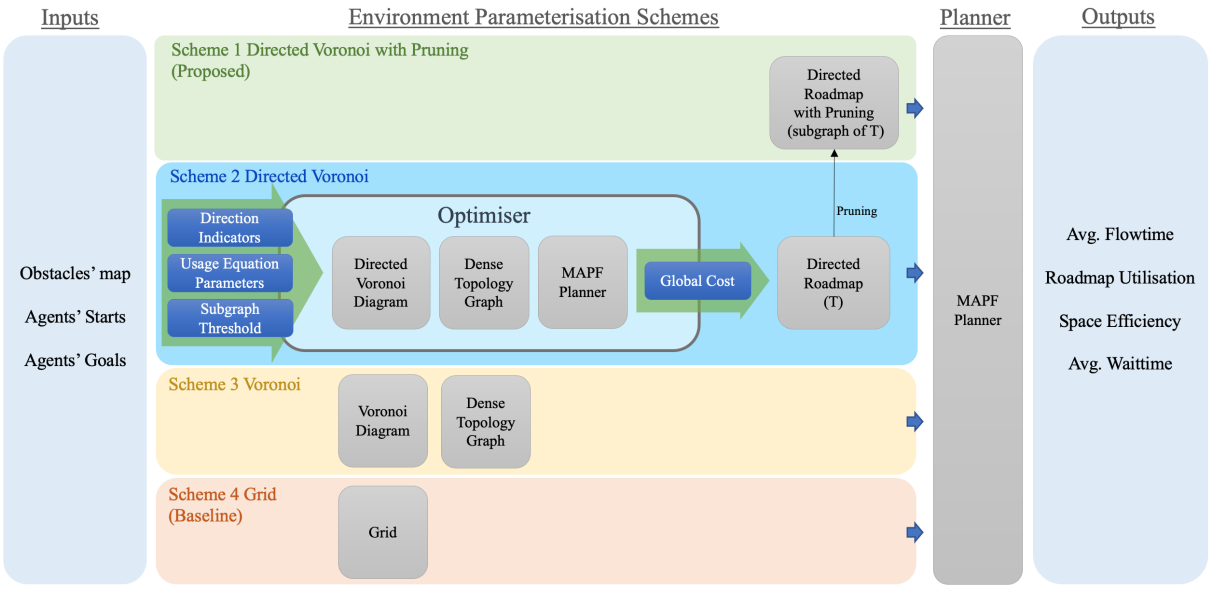


Figure 4.1: The figure shows the overview of the framework.

Referring to the primary hypothesis of the project, we would attempt to answer the question of whether topology-based environment parameterisation schemes will help

improve environmental resources utilisation in multi-agent systems. Our proposed environment optimisation framework will consist of an environment parameterisation schemes, an optimiser and a MAPP solver as shown in Figure 4.1.

In terms of environment parameterisation, upon thorough investigation of alternatives to the grid parameterisation scheme in Chapter 3, we would like to focus on investigating the impact of adopting the Voronoi diagram as the Roadmap. Additionally, as the third and fourth research objectives, we would also like to evaluate the impact of adding direction and usage preferences in the roadmap. To produce a more sparse roadmap, on top of using the Voronoi diagram, we proposed to prune the edges based on the estimated usage preference of the edges. Overall, we would like to compare the performance of the following parameterisation schemes with respect to that of the Grid:

- Voronoi Roadmap
- Directed Voronoi Roadmap
- Directed Voronoi Roadmap with Pruning

After that, the process of determining the direction, usage and the threshold of pruning the edges (subgraph threshold) could be written as an optimisation problem. Its formal mathematical definition will be introduced in Section 4.2.

4.2 Problem Formulation

The roadmap optimisation problem can be defined as follows:

Let $R = r_1, ..., r_n$ be a set of robots acting on a topology graph

$$G = (V, E) \tag{4.1}$$

where V are vertices that map to coordinates in the continuous environment and r_i has start and goal locations $v_{start_i}, v_{goal_i} \in V$. $E(\alpha, \beta)$ are bi-directional edges in G that takes a general form of:

$$e_{src \to tar} = (v_{src}, v_{tar}, c_{src,tar}, d_{e_{src \to tar}}, u_{src,tar}) \tag{4.2}$$

Hence, between any 2 connecting vertices v_a and v_b there exist 2 edges:

$$e_{a \to b} = (v_a, v_b, c_{a,b}, d_{e_{a \to b}}, u_{a,b})$$

$$e_{b \to a} = (v_b, v_a, c_{a,b}, d_{e_{b \to a}}, u_{a,b})$$

where c indicates the capacity of the connection between the two node, d, u correspond to the direction and usage indicators of the edges. Also, opposite edges share identical usage indicators but the direction indicators are of inverse relationship $d_{e_{b\to a}} = (1 - d_{e_{a\to b}})$.

Furthermore, in order to introduce additional usage attribute to road without enlarging the search space, we looked into characteristic of edges that would potentially correlate with road activities. City sustainable design often considered walkability planning [57] and road's connectivity to services and facilities [58]. On top of that, research [11, 12, 13] also shows that street connectivity are often associated with likelihood of physical activity. Hence, we hypothesis that usage indicators and the connectivity of the edges are correlated. We defined connectivity of edges $\gamma_{src,tar}$ as the total number of nodes connected to either ends of the edges. Since road usage would not be increased indefinitely for capacity-limited road, we approximated usage and connectivity with a logarithmic relationship. Moreover, since usage indicator should lie between 0 and 1, a Sigmoid function was used and the α and β refer to the parameters in the function. Therefore, for the edges that exists between vertices v_a and v_b, their connectivity $\gamma_{a,b}$ and usage indicator $u_{a,b}$ are defined as :

$$\gamma_{a,b} = |\{e \in E_G | \textit{if } v_{src} \textit{ or } v_{tar} \textit{ equals to } v_a \textit{ or } v_b \}| \tag{4.3}$$

$$u_{a,b} = (1 + \alpha \times e^{-\gamma_{a,b}+\beta})^{-1} \tag{4.4}$$

Then, we introduced a subgraph threshold δ, edges with usage indicators below a certain threshold would be pruned to improve roadmap utilisation and space efficiency. To assist our analysis, we also introduced a term called Connectivity Threshold δ_c, which is defined as the interception of δ and the Sigmoid Function. It shows the minimum number of connections required to retain the edge.

$$\delta_c = \beta - \ln(\frac{1}{\delta\alpha} - \frac{1}{\alpha}) \tag{4.5}$$

Finally, let G' be the subgraph of G, which is also the final version of the optimised roadmap.

$$E_{G'} = \{e_{src\to tar} \in E_G | u_{src,tar} > \delta\} \tag{4.6}$$

4.2.1 Performance Metrics

The experiments aim to evaluate whether the roadmaps generated from the schemes would be beneficial to utilisation and congestion while maintaining a reasonable completion time. Hence, we adopt 5 different metrics for measuring the schemes' performances and some of them will be part of the optimisation equation.

For timing performance: We would like to measure the distance cost of the solution. To this end, we utilise the average flowtime measurements, which refers to the sum of the distance cost of all the edges in the solution divided by the total number of agents. Let $E = e_1, e_2...e_m$ be the set of edges in the MAPP solution P(G) that involves n number of agents. Also, $e_i = (v_{a_i}, v_{b_i}, c_i, d_i, u_i)$.

$$\text{Average Flowtime } F(G) = \frac{1}{n}\sum_{i=1}^{m}\|v_{a_i} - v_{b_i}\| \tag{4.7}$$

For utilisation performance: We would like to measure the size of the roadmap used relative to the total area of the roadmap, as well as the size of the roadmap with respect to the free space in the map.

Area of Roadmap Used $A_{used}(G)$ is roughly equal to sum of the product of the capacity and distance of edges in P(G). Moreover, to avoid double-counting roads that were being used more than once, Algorithm 1 was used to compute it. Firstly, we define a table that matches vertices pairs with the maximum number of transverse at a single time step (Table_A in Algorithm 1). By the maximum number of transverse at a single time step, we mean to create a list that stored the number of agents that travelled between the respective vertices at different time steps, and the maximum value in the list is the target value. Finally, $A_{used}(G)$ is the sum of the entries in the table.

Algorithm 1: Compute Area of Roadmap Used

```
Table_A = {};
for all timestep t do
    Table_B = {};
    for all agent r do
        (v1, v2) = vertices that agent r passed through at time t;
        Table_B [(v1, v2)] += 1 ;
    end
    Table_A = max ( Table_A, Table_B );
end
A_used(G) = Sum of entries in Table_A
```

$$\text{Total Area of Roadmap } A_{total}(G) = \sum_{e} L(e)C(e) \tag{4.8}$$

where $L(e)$ is the length of edge

$C(e)$ is the capacity of edge

$$\text{Total Area of Freespace } A_{free} = \textit{Map - Obstacles} \tag{4.9}$$

Hence, roadmap utilisation and space efficiency are defined as:

$$\text{Roadmap Utilisation } U(G) = \frac{A_{used}(G)}{A_{total}(G)} \tag{4.10}$$

$$\text{Space Efficiency } S(G) = (\frac{A_{total}(G)}{A_{free}})^{-1} \tag{4.11}$$

For congestion performance: The level of congestion could be reflected by time spent on waiting for available paths. In particular, we look into the Average Waittime $W(G)$. Average Waittime equals to the sum of all the timestep agents spent waiting divided by the total number of timestep taken, and further normalised by the total number of agents. An agent is "waiting" when it stays stationary but it is not at the goal position.

Algorithm 2: Compute Average Waittime

```
n = Number of agents ;
for all agent r do
    for all timestep t do
        cur_step = vertices that agent r passed through at time t;
        if cur_step = last_step then
            wait += 1 ;
        end
        else
            walk += 1 ;
        end
        last_step = cur_step
    end
end
```

$W(G) = \frac{1}{n}\frac{wait}{wait+walk}$

4.2.2 Problem Statement

Finally, the optimisation problem statement is as follows:

Find the set of direction indicators $d \in D$, the parameters (α, β) in Equation 4.4 and subgraph threshold δ, such that it minimise average waittime W(G) and flowtime F(G) while maximising roadmap utilisation U(G) and space efficiency S(G).

We could achieve it by customising the cost functions, the local cost function governs the weights of the edges, whereas the global cost function alerts the optimiser decision.

Local Cost Function: The path planner makes path selections based on the cost of the edges C_{edge}, hence, the cost should reflect the distance and capacity characteristic

of the edge. The cost function will also determine the focus of the system, whether it is optimised for flowtime, optimised for utilisation or both. Experiments in the next chapter will further discuss the impact of these design decisions.

$$\text{Both: } C_{edge}(e_{a\to b}) = \frac{\|v_a - v_b\|}{(u_{a,b})(d_{a\to b})(c_{a,b})} \tag{4.12}$$

$$\text{Utilisation Focused: } C_{edge}(e_{a\to b}) = \left[\frac{1}{(u_{a,b})(d_{a\to b})(c_{a,b}) + \varepsilon}\right] + \varepsilon\|v_a - v_b\| \tag{4.13}$$

$$\text{Flowtime Focused: } C_{edge}(e_{a\to b}) = \|v_a - v_b\| + \varepsilon\left[\frac{1}{(u_{a,b})(d_{a\to b})(c_{a,b}) + \varepsilon}\right] \tag{4.14}$$

where ε is a small number equals to 1e-3

Global Evaluation Function: The global evaluation function returns a numerical value to the solver as feedback of the overall system performance given a roadmap G. The solver will aim to test a set of direction indicators, α, β, subgraph threshold that returns the lowest cost solution.

In addition, there is also a need to reflect negative samples to the optimiser, hence a penalty term has been introduced. When the CBS failed to converge within 1e3 iterations, a penalty term would be used to penalise it. It is based on how far away the agent is from their target location at the end of the iterations.

Let v_{end_i} and v_{goal_i} be the ending locations and goal locations of the i^{th} agents and n be the total number of agent,

$$Penalty = \sum_{r=0}^{n} \|v_{end_r} - v_{goal_r}\| \tag{4.15}$$

The global cost is then defined as:

$$C_{global}(G) = \frac{F(G)(W(G)+1)}{U(G)+\varepsilon} + \rho(Penalty + 1) \tag{4.16}$$

where ε and ρ are constants with magnitude of 1e-3 and 1e3 respectively.

The C_{global} is composed of average flowtime, roadmap utilisation and the penalty terms, which reflects the expectations defined in the problem statement. Space efficiency has not been incorporated into the equation as its performance often mirrors that of roadmap utilisation U(G). Also, we observed that the first part of the C_{global} is often within the order of magnitude of 2 and that resorting to an incomplete solution is undesirable. Hence, we have set an arbitrary high weight ρ, which is an order of magnitude from other costs to discourage this behaviour.

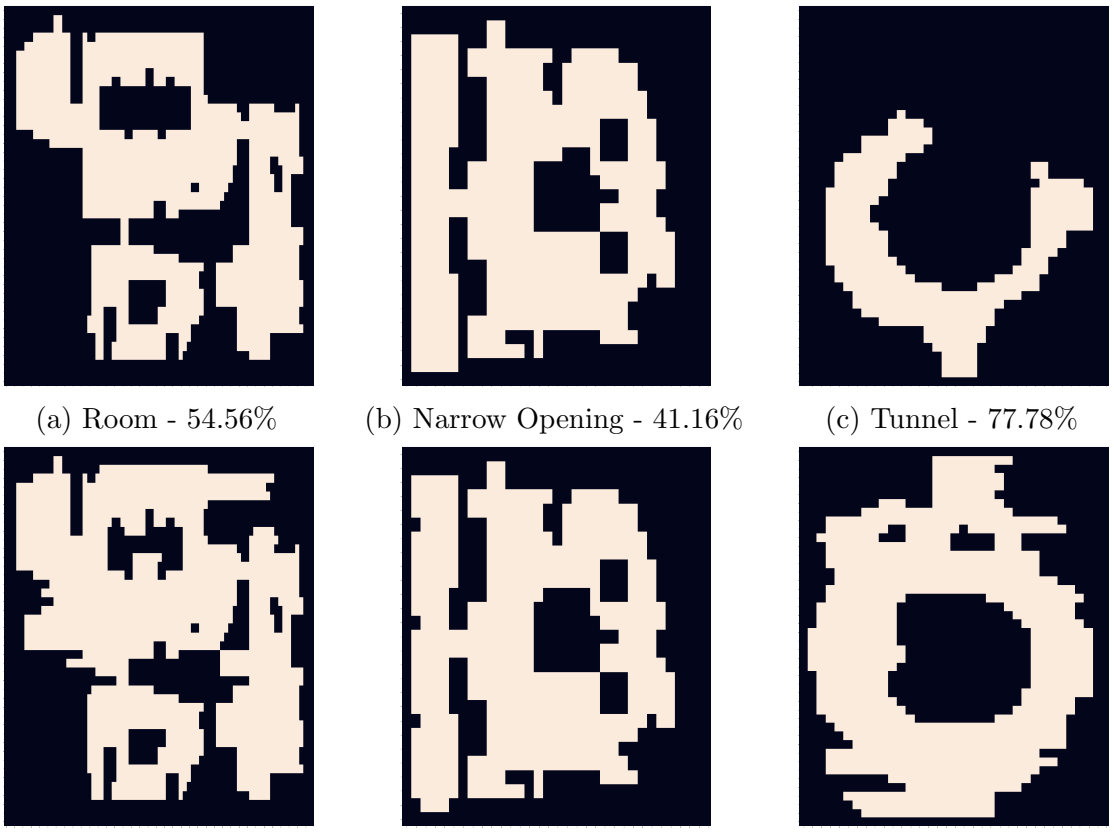
4.3 Experiment Setup

4.3.1 Dataset

In this pieces of work, we have utilised 3 benchmarks datasets from the Moving AI Lab [59] as the basis. They are part of the "Dragon Age: Origins" commercial game benchmarks. They feature a room-like environment (den101d), a narrow opening (lak105d) and a circular tunnel (lak109d).

As part of the investigation, we would like to evaluate the impact of the overall shape of the environment and the obstacle density on the performance of various environment parameterisation schemes. Therefore, we have derived two different sets of data based on the three aforementioned datasets: Data Series A) Varying Shape with Similar Obstacle Density; Data Series B) Varying Obstacle Density with Similar Shape.

4.3.1.1 Varying Shape with Similar Obstacle Density



(a) Room - 54.56% (b) Narrow Opening - 41.16% (c) Tunnel - 77.78%

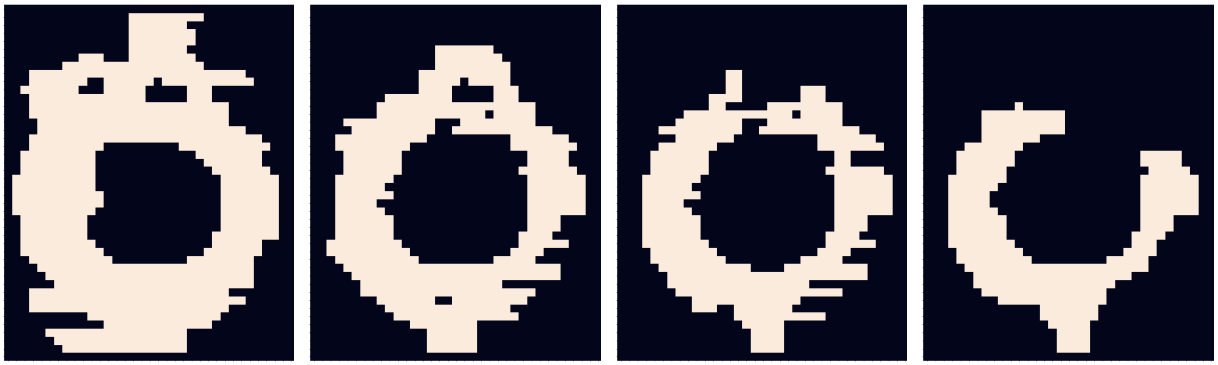
(d) Room - 45.44% (e) Narrow Opening - 43.87% (f) Tunnel - 45.81%

Figure 4.2: The dataset that contains environments with varying shapes. It also shows the changes in obstacles density for the 3 environments relative to the original benchmark.

The comparison of the changes in obstacle density is as shown in Figure 4.2. 44-46 % was chosen because it is the minimum obstacle density among the maps, which other

datasets with higher original obstacle density only need to remove obstacles to reach similar density, which means the prefined set of tasks in the benchmark will remain solvable after the modifications. These sets of maps would be used to evaluate the performance of the parameterisation and roadmap generation schemes under different environment. Although the obstacle density for these set of maps was fixed, the next set of data will help fill the void.

4.3.1.2 Varying Obstacle Density with Similar Shape



(a) 45 % (b) 60 % (c) 70 % (d) 75 %

Figure 4.3: The dataset that contains environments with varying obstacle density.

The second set of maps were designed for evaluating the impact of obstacle density. The benchmark with the highest initial obstacle density - Tunnel was utilised. This ensures that most of the scenarios set out in the benchmark would remain solvable because a minimal amount of obstacles were added to the map. The number of obstacles within the map was then adjusted accordingly to 45%, 60%, 70% and 75%.

4.3.1.3 Limitations of Dataset

There are a few limitations of the methodologies used in data sourcing and could be further improved in future work.

Firstly, although we aim to adhere to the commonly available benchmarks as set out by N. Sturtevant (2012) [59], such that our results would be reproducible and comparable with other research, they have been slightly adjusted to ensure fairness in the experiments. Furthermore, due to the nature of the congestion problem, a specific environment is often required to highlight characteristics of various utilisation-aware schemes. Most literature that considered the congestion problem have used a custom environment without any correlations to open-sources benchmarks. Therefore, future work should consider setting up a benchmark for the MAPP congestion problem, such that results in the relevant field could be unified.

Another concern would be the lack of considerations of other environmental factors, such as the local agent density and local agent/obstacle density. These may induce bias when comparing the performance of the schemes in environments of varying shapes (Data Series A). Therefore, future work should also consider setting level playing fields in terms of these metrics.

4.4 Implementation of Environment Parameterisation Schemes

In the most basic path planning formulation, occupancy information could be stored in a Manhattan distance 8-connected grid. Afterwards, as mentioned in Section 3.3.1, we dilated the obstacles, such that as long as the control point of the robot is outside of the boundary, it will not collide with the obstacle. This forms our 2D occupancy grid. It also acted as our grid-based baseline (Scheme 4).

A Voronoi diagram was then computed using Qhull Library [60] through an API from SciPy (*scipy.spatial.Voronoi* [61]). Moreover, the library came with the limitation of only considering point site Voronoi diagram. Therefore, with the Voronoi vertices obtained, we further removed invalid edges and vertices that lie within the non-free space area based on the occupancy information from the occupancy grid. The start and goal locations were also added to the graph as new nodes and edges to ensure the path planning is possible as illustrated in Figure 3.1b. This forms the skeleton graph of the environment.

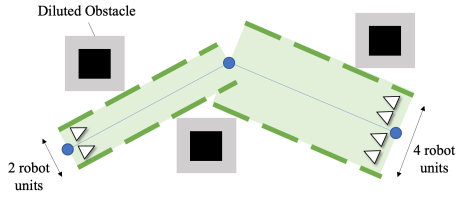


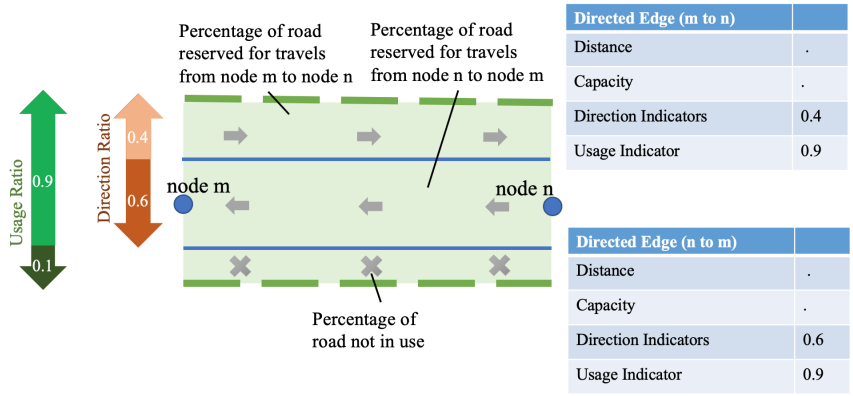
Figure 4.4: The edges in the roadmap would be expanded until they meet any obstacle. The capacity of the edges are in robot units as illustrated by the triangular agents in the figure.

In addition, the capacity of the edges in the skeleton graph was also calculated, which would transform these edges into corridors. This process has been demonstrated in Figure 4.4. Similar to daily traffic, we considered the capacity of the edges as how many "lanes" there are along the two target vertices. Hence, we expanded the edges along with the orthogonal directions and stopped them once they encountered any obstacles. To reduce overlapping of corridors, we adopted two measures:

- Form a temporary occupancy grid and add the road with capacity already being calculated as occupied.
- Add a tolerance to determining invalid edges, such that the edge will not be classified as invalid at the first sight of overlap. But it will be classified as overlapped if the overlapping is serious.

The skeleton graph combined with the capacity information formed our Scheme 3 - the Voronoi Roadmap.

Furthermore, we added the additional direction as additional attributes to the edges of the Voronoi Roadmap. The direction indicators will then optimised by our optimiser. This becomes our Scheme 2 - the Directed Voronoi Roadmap.



Directed Edge (m to n)	
Distance	.
Capacity	.
Direction Indicators	0.4
Usage Indicator	0.9

Directed Edge (n to m)	
Distance	.
Capacity	.
Direction Indicators	0.6
Usage Indicator	0.9

Figure 4.5: Each edge e consists of 4 attributes as shown in the table. The direction ratio 0.4 : 0.6 will be the the direction indicators of $e_{m\to n}$ and $e_{n\to m}$, whereas the usage indicator takes the 0.9 in the usage ratio.

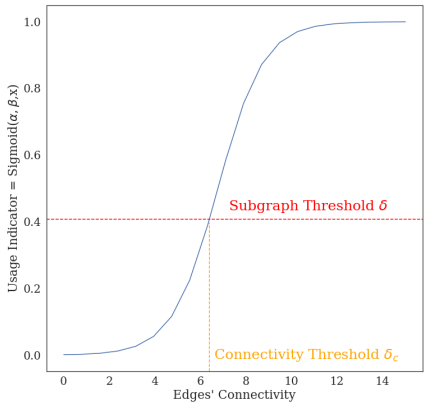


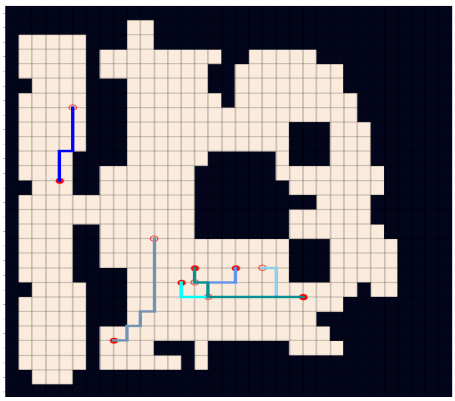
Figure 4.6: The figure shows the hypothetical relationship between connectivity and existence of edges. α, β corresponds to the parameters in the Sigmoid Function introduced in Equation 4.4

Next, we also introduced usage indicators to the edges. The list of attributes of edges in the final scheme could be viewed in 4.5. Between node m and node n, the "lane" would be partition into three sessions: the percentage that is reserved for traversing from m to n, the percentage that is reserved for traversing from n to m and the remaining area is not used. These characteristics could be summarised with 2 ratios: usage ratio and direction ratio as shown in Figure 4.5, where the direction ratio would become the direction indicators of respective edges, and in the used verse unused ratio, the usage of the connection will be documented.

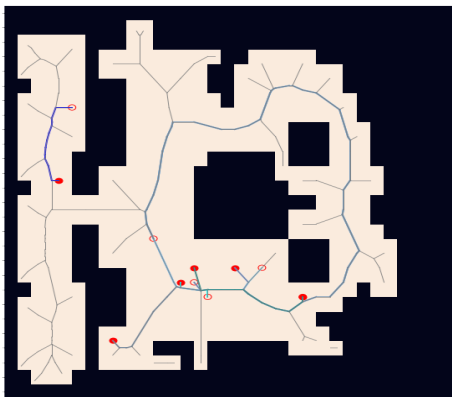
On top of that, to reduce the search space, we hypothesised that usage indicators and the connectivity of the edges are correlated, and represented their relationship with a Sigmoid function, the coefficient in the function α and β as shown in Equation 4.4 would then be optimised alongside the subgraph threshold δ. As shown in Figure 4.6, edges with usages indicator below the subgraph threshold δ, i.e. edges with connections less than the connectivity threshold δ_c, would be pruned away. A subgraph of the Directed Voronoi Roadmap would then be formed, which helps improve roadmap utilisation and space efficiency. This is similar to the transformation from Figure 4.7c to 4.7d. The possibility of the edge pruning process adversely affecting the global connectivity of the roadmap is low because the system will be heavily penalised by the penalty cost and the optimiser will help us avoid these high-cost samples. To further safeguard the global connectivity, we also restraint the system from pruning edges that are connected to any starting or goal locations. Also, we predict that as the environment becomes more challenging, less edge could be pruned away without affecting the global connectivity of the roadmap, hence, the requirements for retaining the edges would loosen as the environment become denser, or when there are more agents.

Furthermore, as discussed in Section 3.5.2, BO is only suited for limited amount of parameters. Moreover, the impact of edges on the overall system performance is proportional to the area that it occupied, hence, the longer the edge, the more impactful it will be to optimise its direction and usage. Therefore, in this work, we only considered the optimisation of direction and usage for connections with a Euclidean distance of 3 or above. The threshold was chosen based on empirical experiments and the size of maps. These leave us with around 41 to 60 parameters for optimisation, which still exceeded the recommendation 20, but is already a large improvement from the original amount of 700 to 800 parameters if no steps were taken to reduce the search space. In Chapter 5, there will be a thorough investigation of the performance of the schemes, which shows that despite exceeding the recommended parameter search space, BO still shows adequate performance.

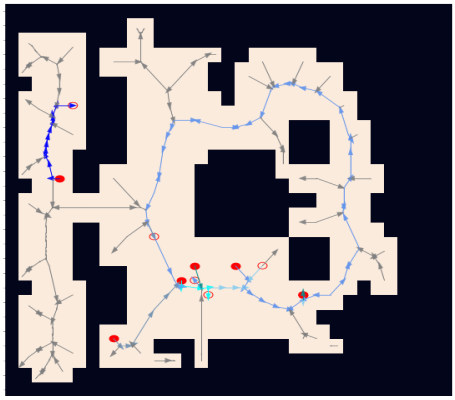
In conclusion, during each iteration in the optimisation loop, a pair of (α, β, δ) alongside the direction indicators were used to set up a roadmap, then a global cost would be obtained



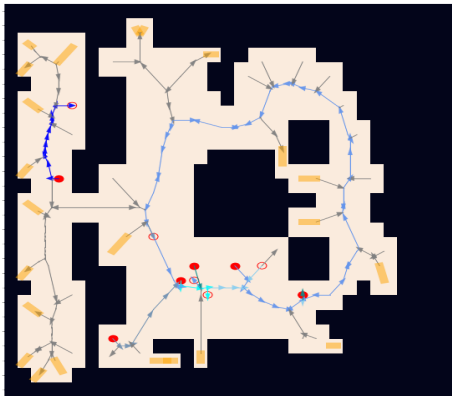
(a) Grid



(b) Voronoi Roadmap



(c) Directed Voronoi Roadmap



(d) Directed Voronoi Roadmap with Pruning

Figure 4.7: The figure shows the roadmaps from different Environment parameterisation schemes. Grey lines and the colour lines represents the roadmap and the chosen paths respective. The empty and filled circles represents start and goal locations. Orange patches highlighted the pruned edges.

from the MAPP solver based on the average flowtime, roadmap utilisation and the penalty cost. This would be the final proposed scheme - Directed Voronoi Roadmap with Pruning (Scheme 4). Overall, Figure 4.7 gives a summary of the schemes used in this work.

4.5 Implementation of Optimisers

Optimisers were used to determine the set of (α, β, δ) and direction indicators that generate the lowest global cost. We have investigated the performance 3 different optimisation schemes:

- Naive Method (baseline)
- GP with BO

- ANN with L-BFGS

The naive way of determining the values of the parameters is through random assignments. For more robust experiments, we tested the system performance with 3 randomly chosen random seeds and recorded the one that achieved the best performance for each setting. This acts as our baseline for verifying the performance of other optimisation schemes. The computational time for this method is:

$$t_{Naive} = n \times t_{samples} \tag{4.17}$$

where $t_{samples}$ is the time taken to generate 1 sample and n equals 3.

The GP with BO method was implemented with the open-source GPy library from the Sheffield machine learning group [62]. An RBF kernel was used for the GP and the Expected Improvement acquisition function was used for the Bayesian experimental design. To verify the sample efficiency of different optimisers, we will also test the method under a different number of total samples. Towards this end, 60% would be the random samples for fitting the GP and the remaining 40% would be the samples obtained from the Expected Improvement acquisition function. The ratio was chosen based on our observations on the performance of the optimiser. In addition, the computation time of the method t_{BO} equals to:

$$t_{BO} = t_{Random-sample} + t_{GP} + t_{EI-samples} \tag{4.18}$$

where $t_{Random-sample}, t_{GP}, t_{EI-samples}$ refers to the time spent on obtaining the random samples, fitting them to the GP and acquiring Expected Improvements samples respectively.

Layer (type)	Output Shape	Param #
dense_19 (Dense)	(None, 1, 128)	5376
dropout_14 (Dropout)	(None, 1, 128)	0
dense_20 (Dense)	(None, 1, 32)	4128
dropout_15 (Dropout)	(None, 1, 32)	0
dense_21 (Dense)	(None, 1, 1)	33

Total params: 9,537
Trainable params: 9,537
Non-trainable params: 0

Table 4.1: Table shows the ANN architecture.

The ANN with L-BFGS method was implemented with *Keras* model [63] and *scipy.optimize* library [64]. A simple ANN architecture shown in Figure 4.1 was used. Mean squared error was used as the cost function. Based on the observations on the model's training and validation loss and to prevent under or overfitting, we have used 16 as batch size, 1×10^{-5} as the learning rate and 100 epochs with early stopping for the model training. For the L-BFGS algorithm, the variation presented by C. Zhu et al. (1997) [65] and S.

Nash (1984) [66] was used for the bound-constrained minimisation. The hyperparameters of the algorithms were chosen based on empirical experiments, which aims to balanced the performance and computation time. Details of them can be found in the Appendix A. Lastly, the computation time of the method t_{ANN} equals to:

$$t_{ANN} = t_{Random-sample} + t_{ANN} + t_{L-BFGS} \tag{4.19}$$

where $t_{Random-sample}, t_{ANN}, t_{L-BFGS}$ refers to the time taken to generate the required amount of random samples, time used for training the ANN and time required for the L-BFGS algorithms to converge.

4.6 Implementation of modified Conflict-based Search

Based on the analysis in Section 2.1, we will utilise CBS as the path planner. However, the conventional CBS usually consider edges with single robot capacity. Therefore, in our work, we have modified the algorithm in terms of collision conflicts, swap conflicts and cost function such that it can work with the Directed Voronoi Roadmap.

In CBS, as long as another agent is traversing at the same connection at the same time, conflict would be flagged, moreover, in our framework, we consider conflicts only when there are the violation of capacity. There is an edge conflicts at e at time t for some agents r, if and only if,

$$\sum_{r} f(e,t) > c_{edge}(e) \tag{4.20}$$

where $f(e,t) =$ Number of agents that traversed through edge e at time t

$c_{edge}(e) =$ Capacity of Edge e

Similarly, node conflicts is defined as:

$$\sum_{r} f(n,t) > c_{node}(n) \tag{4.21}$$

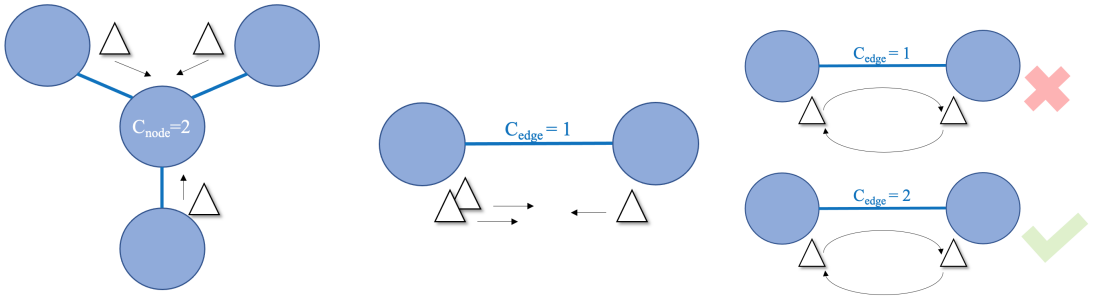
where $f(n,t) =$ Number of agents at node n at time t

$c_{node}(n) =$ Capacity of Node n

It is also worth noting that although there exists a preferred direction and usage for each edge implied by the indicators, they are soft constraint only. These parameters would have an impact on the moving cost of edges and hence, potentially encourage or discourage the selection of some paths, but they are not a hard limit. Hence, they have not been included in the above equations.

Also, in the classic MAPP problem, nodes and edges capacities are usually of 1 robot

unit, where a unit refers to the dimension of a single agent. Moreover, in our problem formulation, the capacity term was utilised to help form a more sparse roadmap. It is also worth noting that in conventional MAPP problems, agents should not be allowed to swap locations in a single time step, moreover, under our framework, such transitions are allowed as long as the involved edges have capacities of more than 1 robot unit.



(a) Node (b) Edge (c) Swap

Figure 4.8: There are 3 possible collision scenarios in roadmaps with capacity. The triangles, C_{node}, C_{edge} refer to the agents and the capacity of nodes and edges.

Lastly, the cost function used in path search is also slightly different from the conventional one. In the $\mathcal{A}*$ search algorithm, the $g(n)$ in Equation 3.4 changed from distances cost of edges to C_{edges} listed in Equation 4.12, 4.13, 4.14.

Chapter 5

Evaluation

In this chapter, we would like to justify the choice of the optimiser. After that, the impact of varying the local cost function will be evaluated. Then, the performance of the environment parameterisation schemes will also be analysed under different maps and obstacle densities. Finally, we will also discuss the trend of the subgraph threshold.

5.1 Experiments Setup

The performance metrics have been defined in Section 4.2.1: **Average Flowtime, Roadmap Utilisation, Space Efficiency, Average Waittime**. Flowtime and waittime are negatively correlated with the system performance, whereas roadmap utilisation and space efficiency are positively correlated to it. To better visualise the performance of the proposed scheme, the following analysis is documented with the "Percentage Gain" of different schemes relative to the corresponding experiments' baseline. A point to note is that "Loss" instead of percentage has been used to visualise changes in the average waittime. This is because waittime in the baseline has always been observed as 0, which makes it impossible to compute the percentage change. Overall, **more positive "Percentage Gain w.r.t. baseline", indicates higher performance, whereas lower "Loss w.r.t. baseline" indicates better performance.**

Additionally, in Figure 5.1, we have used a modified logarithm scale, also known as the symmetric logarithm scale, as the y-axis instead of linear scale due to the range of the data. The modified scale was used because it is capable of handles negative values while maintaining continuity across zero. It was introduced by J. B. W. Webber (2012) [67] and has been utilised by popular mathematical simulator [68]. The scale is defined as:

$$y = sign(x) * \frac{log_{10}(1 + |x|)}{10^C} \tag{5.1}$$

where the scaling constant C determines the resolution of the data around zero and in our

work, it has been defaulted as 0.

Finally, unless otherwise specified, all the schemes were evaluated on 3 datasets per settings. The standard deviation of the results will be listed in tables or as error bars in bar charts.

5.2 Optimiser Design

Before diving into the roadmap optimisation experiments, we would like to find the right tool for the intended optimisation problem, which has also been the second research objective of the project. Therefore, a set of experiments have been formulated to compare the performance of BO and ANN optimiser relative to a Naive Methods baseline. BO optimiser is the abstract of the GP with BO optimisation schemes, while ANN optimiser refers to the ANN with L-BFGS optimisation schemes.

These optimisation schemes would try to find a minimum cost solution to the MAPP problem. Table 5.1 shows a summary of the General Cost of the solutions attained under different optimisation approaches. By general cost, we referred to the global cost (in Equation 4.16) excluding the penalty term, i.e.

$$C_{general}(G) = \frac{F(G)(W(G)+1)}{U(G)} \tag{5.2}$$

$$\tag{5.3}$$

where $F(G)$, $W(G)$, $U(G)$ corresponds to the average flowtime, average waittime and roadmap utilisation achieved when using the roadmap G. Also, ε is a small number and equals $1e-3$. General cost instead of the global one because solutions in this set of experiments are complete and there is no associated penalty cost, hence the weight constant ρ could be removed. It gives us a more balanced view of the performance of the optimiser. In addition, Figure 5.1 also gives an overview of the computational complexity of various schemes.

		10 Samples	**50 Samples**	**100 Samples**	**500 Samples**	**1000 Samples**	**2000 Samples**
4 ag.	**ANN**	-1.11 % ± 4.16	-1.90 % ± 3.30	-38.67 % ± 30.19	-1.90 % ± 3.30	-1.11 % ± 4.16	0.01 % ± 0.80
	BO	2.88 % ± 2.72	5.09 % ± 3.78	3.44 % ± 2.68	8.70 % ± 11.58	nan % ± nan	nan % ± nan
8 ag.	**ANN**	5.41 % ± 10.12	-8.97 % ± 37.18	-0.29 % ± 0.49	-0.10 % ± 0.88	0.22 % ± 0.49	10.11 % ± 12.69
	BO	10.20 % ± 9.88	21.07 % ± 3.23	23.41 % ± 6.06	26.44 % ± 4.99	nan % ± nan	nan % ± nan

Table 5.1: Percentage gain in general cost relative to the naive optimiser.

Overall, it has been observed from Table 5.1 that BO brings consistent improvements to the global cost achieved, whereas, in contrast, ANN performed worst than the baseline in some cases. Furthermore, under the same number of samples, BO always manage to

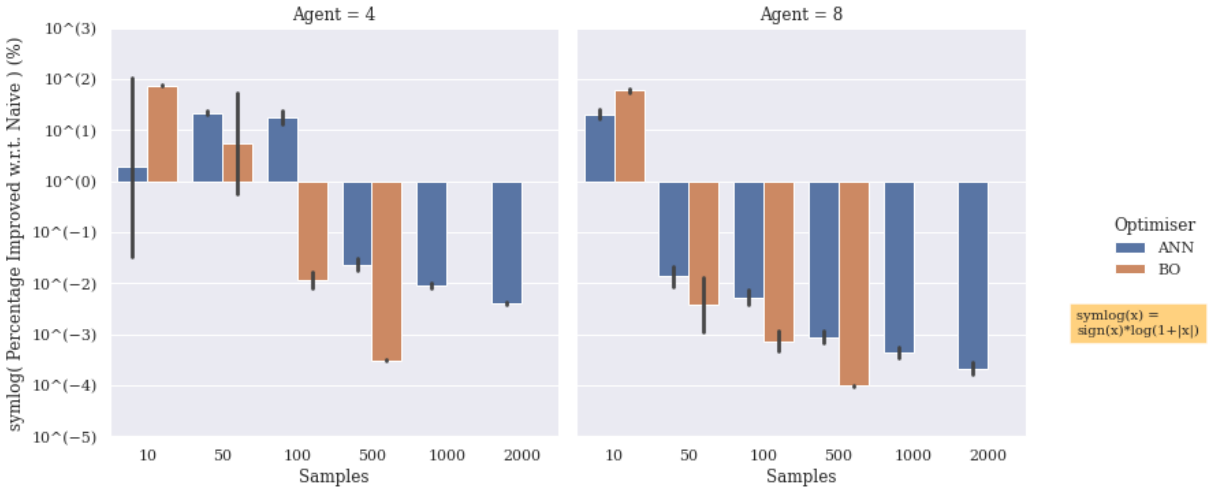


Figure 5.1: Percentage gain in computational time relative to naive optimiser.

find a lower-cost solution compared to ANN, as well as the naive methods. This can be understood as BO attempts to find a lower-cost solution based on probability, whereas ANN optimiser is a gradient-based method, which has a higher chance of convergence at local minima.

In terms of the computational time shown in Figure 5.1, both ANN and BO are usually more computationally expensive than the naive methods. This has been expected because they utilised more samples to fit their probabilistic models, whereas the baseline method does not require any fitting or training of models. It can also be seen that ANN requires less computational time than BO in most cases. Referring to Equation 3.5, the computation time for Expected Improvement samples increases substantially at high dimensions, which in turn increase the $t_{EI-samples}$ term in Equation 4.18. The high computational cost of BO can be explained as the complexity of the scheme is around $O(N^3)$, where N is the number of data points. This is also the reason for displaying the data on a symmetric logarithm scale. Note that beyond 500 samples, BO becomes intractable to compute.

Although BO optimiser does not scale well the number of samples, its sample efficiency appears to be better than that of ANN. Taking the case with 8 agents as an example, the BO optimiser can generate results 2.1 times better than that of ANN using only $\frac{1}{40}$ of samples. This can be understood as BO utilises the Expected Improvement acquisition function to balance exploration and exploitation whereas the L-BFGS algorithms does not. An appropriate balance of exploration and exploitation will allow us to search the parameter space much more efficiently.

There is also another point to note regarding the variance in their computational time. The performance of the optimisers was more consistent when the MAPP problem involves more agents. We observed instability especially when the number of agents and the number of samples used are both at the lower end of the spectrum. This might be

because the optimisers were yet to converge and that a lower number of agents implies that there is higher flexibility in selecting paths, hence the high variations.

In conclusion, BO optimiser manages to double the performance of ANN with only one-forty of the samples, which also implies using 94% less computational time. Besides, BO also generate lower-cost solutions relative to the baseline at all time. This, however, has been achieved at a cost of computational time. Upon balancing the trade-off between performance and computational resources, it has been determined that BO with a relatively small samples size, such as 50, would be the most appropriate optimiser among the three for the following experiments.

5.3 Performance of the Environment Parameterisation Schemes

To provide a comprehensive review of the proposed scheme, we will first analyse the impact of the local cost function and verify its performance. This will address the second research objective. Next, the performances of the environment parameterisation schemes would be evaluated under different maps and obstacle density. This addressed the first, third and fourth research objectives. Finally, we will also investigate the trend of the subgraph threshold, as part of the fourth research objective. The baseline of experiments in Section 5.3 would be the performance achieved by using a Grid-based Roadmap.

5.3.1 Local Cost Functions with Different Objectives

The local cost function that governs the path selection process in the MAPP solver can shifted its parameters in the equation to achieve various objectives, namely, Flowtime-Focused (Equation 4.14), Utilisation-Focused (Equation 4.13) and Both-Focused (Equation 4.12). We summarised the impact of changing these cost functions under 4 and 6 agents cases in Figure 5.2.

In general, the observations match the expectations. It is expected that the highlighted column in Figure 5.2 to have a generally better performance than their counter-part columns in their respective plots, as the Flowtime-Focused Cost Function should enable us to achieve better flowtime performance than that of Utilisation-Focused and vice versa for Utilisation-Focused.

An interesting point is that the Both-Focused Cost Function allows us to achieve the best roadmap utilisation performance and even surpassed that of Utilisation-Focused under both 4 and 6 agents case. This shows that optimising for both objectives together may allow us to acquire additional gain.

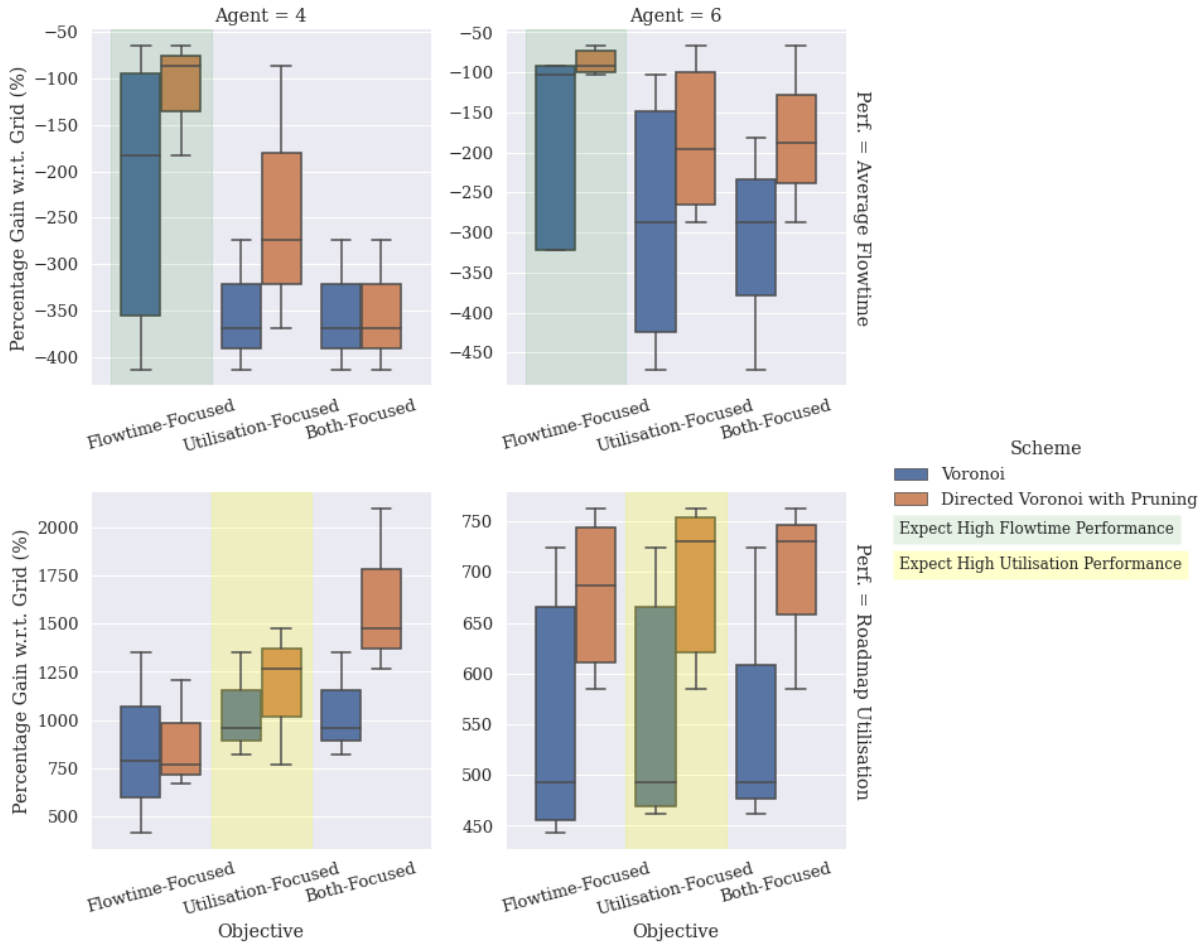
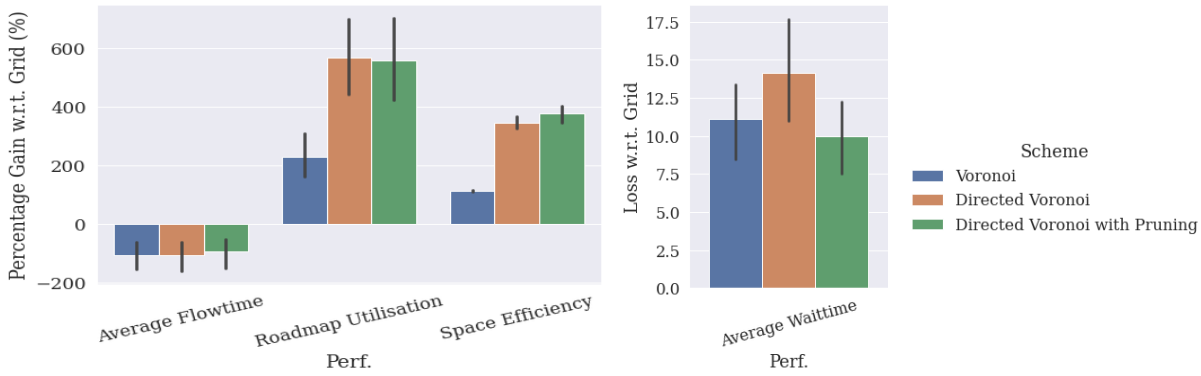


Figure 5.2: The impact of using local cost function with different objectives.

5.3.2 General Performance



(a) Avg. Flowtime, Roadmap Util., Space Effi. (b) Avg. Waittime

Figure 5.3: A summary of the performance of different environment parameterisation scheme. The proposed scheme has achieved the best performance compared to the Voronoi and Directed Voronoi, it attained the highest percentage gain in average flowtime, roadmap utilisation and space efficiency, as well as the lowest loss in average waittime.

Overall, the proposed schemes - Directed Voronoi with Pruning has achieved much better environmental resources utilisation relative to the baseline grid method at the cost of some flowtime performance. On average, our proposed scheme achieved a 574.78%, 386.85% improvements in roadmap utilisation and space efficiency relative to the grid baseline, at a cost of 88.09% decrease in flowtime performance. In comparison, the Voronoi scheme was only able to improve environmental resources utilisation performance by 226.11%, 113.63%, while trading off more flowtime performance of 99.72%. It only brings 0.39 (roadmap utilisation) and 0.29 (space efficiency) of the performance gain attained by the proposed scheme but cost 1.13 times of flowtime performance. For the scheme without roadmap pruning (Scheme 2 - Directed Voronoi), we achieved a similar performance gain as that of the proposed methods, namely, the performance gain of 567.34% and 349.08%, but also sacrificed more flowtime performance relative to scheme 1, i.e., flowtime performance declined by 100.83%. These findings confirmed our hypothesis on topology-based environment parameterisation schemes can help improve environmental resources utilisation, and that including directions and usages specifications have a positive impact on maintaining flowtime performance.

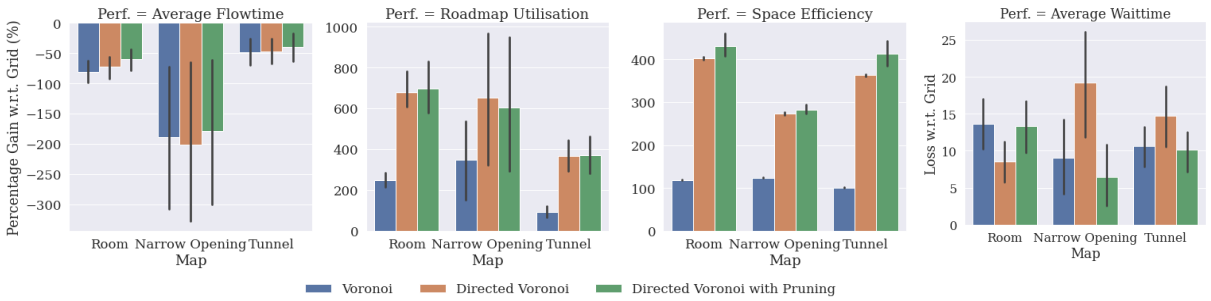
In most cases, the flowtime achieved by the schemes is worst than the baseline. This can be explained as the Voronoi-based roadmaps only use roughly 25.85% of the free space to construct the graphs, whereas the grid method uses 100% of the free space, as reflected by the high space efficiency in Figure 5.3. flowtime performance declined as expected due to less traversable space.

In terms of the waittime performance shown in Figure 5.3, due to the difference in the size of the roadmap, waiting rarely occurs in the dense Grid-based roadmap, which is not the case in a sparse Voronoi-based roadmap. Hence, average waittime is often negative relative to that of the Grid, and is represented as loss in Figure 5.3. However, the proposed scheme has still achieved the lowest loss in average waittime, and is 37.54% and 10.30% less than that of the Directed Voronoi and the Voronoi scheme.

5.3.3 Vary Environment Shape

In this set of experiments, the schemes were tested on the 3 different maps (Data Series A). Figure 5.4 shows a summary of how the schemes perform under environment of different shape, and a detailed breakdown of how they scale with the number of agents have been provided in Figure 5.5, 5.6). Also, the numerical details can be found in Appendix B.

Overall, Figure 5.4 shows that the proposed scheme can achieve most performance gain w.r.t. Grid in terms of roadmap utilisation of 697.75% in the Room environment, and the best space efficiency and flowtime performance at the Tunnel environment of 431.74% and -41.56%. Moreover, the phenomenon could be explained because the Room environment is less restrictive than that of Tunnel, hence easier to achieve high roadmap utilisation. For



(a) Avg. Flowtime, Roadmap Util., Space Effi. (b) Avg. Waittime

Figure 5.4: A summary of percentage gain and loss of different performance metric w.r.t. grid under different environment of different shape. The best results in roadmap utilisation, space efficiency and average waittime was achieved under Room, Tunnel and Narrow Opening environment respective.

the high space efficiency, it is because the Tunnel environment has less discrete obstacles than that of the Room, meaning that agents could concentrate on travelling at the main road, hence, less free space was being used to construct the roadmap. While for the congestion performance, the proposed scheme achieved the least loss in waittime w.r.t to the Grid of 6.96 at the Narrow Opening environment. This suggested that the proposed scheme is useful in resolving congestion in Narrow Opening environments by spreading out the agents and encouraging roadmap utilisation. Similar improvements have not been observed in the Tunnel environment, which is also likely to have congestion due to the concentration of agents at the main road, is because encouraging roadmap utilisation is not useful in handling congestion at tunnels when there is only a single road to traverse from one end to the other. These sets of experiments help us identify the strength and weakness of our proposed scheme and demonstrated that different types of environments may benefit from the proposed scheme in different ways.

In terms of the scalability of the framework, flowtime performance improved while that of roadmap utilisation and space efficiency declined as the number of agents increases for the Room and Narrow Opening environment. This can be observed in the Room environment case in Figure 5.5, as all the schemes establish downward trends for roadmap utilisation and space efficiency. This is because as the environment becomes more crowded, there is less flexibility in the path selections for agents. Moreover, some abnormalities are expected at the highlighted red column in Figure 5.5 when the MAPP solver failed to converge because it means that all the agents stayed at their starting location. This explains the slightly positive flowtime performance in the figure.

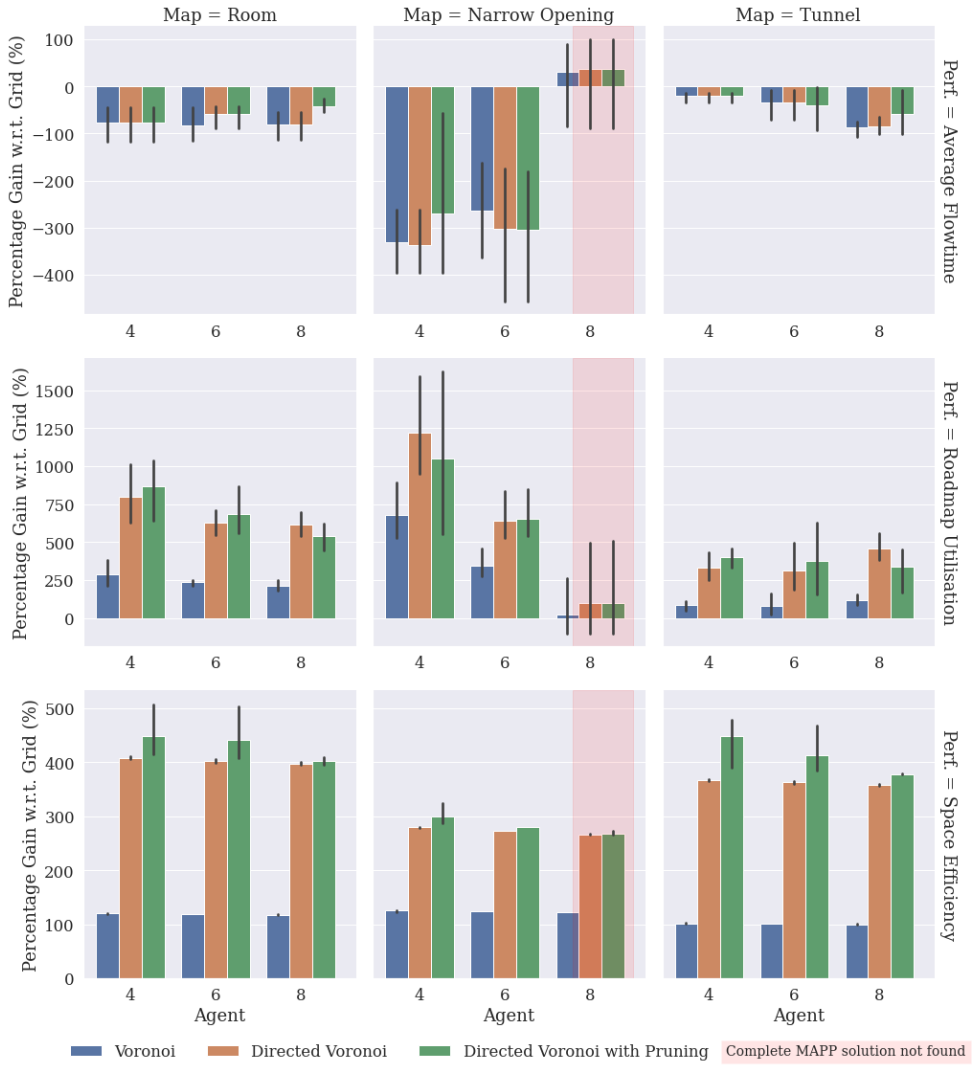


Figure 5.5: Performance plot of the parameterisation schemes on flowtime, roadmap utilisation and space efficiency under environment of different shape. Note that higher percentage gain means better performance.

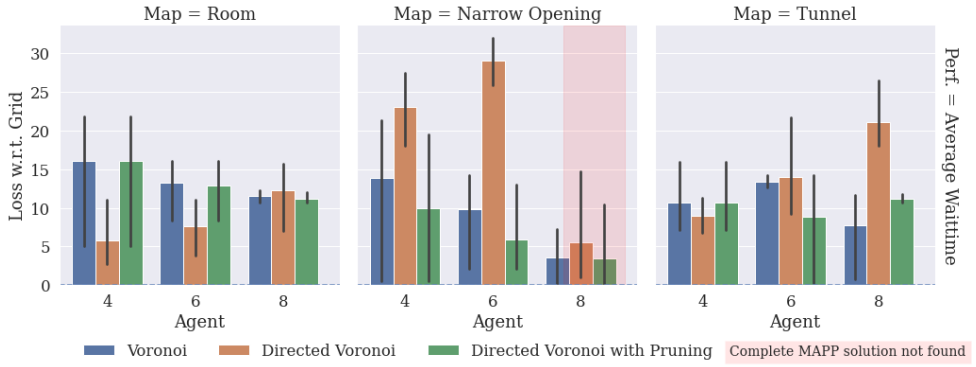
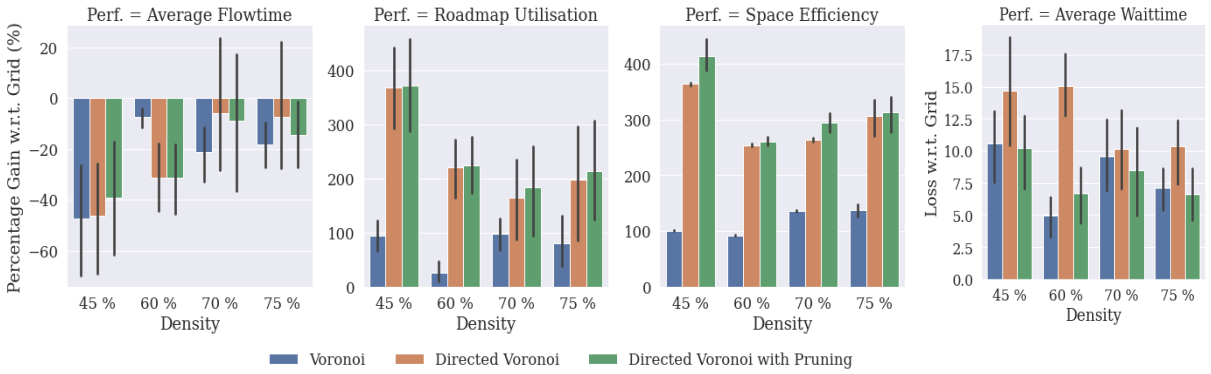


Figure 5.6: Performance plot of the parameterisation schemes on average waittime under environment of different shape. Note that lower loss means better performance.

5.3.4 Vary Obstacle Density



(a) Avg. Flowtime, Roadmap Util., Space Effi. (b) Waittime

Figure 5.7: A summary of percentage gain and loss of different performance metric w.r.t. grid under different obstacle density . Under high obstacle density (75%), the proposed scheme (green) still achieves better performance in roadmap utilisation, space efficiency and average waittime that Directed Voronoi (orange) and Voronoi (blue).

On top of the impact from the variations in environment shapes, obstacle density also plays a singular role. In this set of experiments, the schemes were tested on the Tunnels Map with varying obstacle density (Data Series B). Figure 5.7 shows a summary of how the schemes perform under different obstacle density, and a detailed breakdown of how they scale with the number of agents have been provided in Figure 5.8, 5.9. Also, the numerical details can be found in Appendix B.

Overall, Figure 5.7 shows that at a higher obstacle density environment, the proposed scheme brings 356.53% and 326.04% improvements in roadmap utilisation and space efficiency relative to the grid-based method, while suffering from a 35.00% decrease in flowtime performance. Although the performance gain is only 1.09 and 1.01 times that of the Directed Voronoi scheme, it shows a much better performance than the Voronoi schemes. The improvements achieved by the proposed scheme is 2.29, 2.31 times that of the Voronoi roadmap and sacrificed 2.58% less the flowtime performance. For the average waittime, at the environment with 75% obstacle density, the proposed scheme also shows the lowest loss among the 3 schemes. It attained a 6.96 loss, which is 30.41% and 4.33% less than that of the Directed Voronoi and the Voronoi. This shows that our proposed scheme performance still surpass other schemes despite in a high-density environment, giving evidence that edges with direction and usage specifications are capable of promoting environmental resources utilisation while maintaining a reasonable flowtime. Moreover, its significant improvements in roadmap optimisation and space efficiency of the schemes relative to the grid baseline also highlight the benefit of adopting a topology-based roadmap. These observations support our primary hypothesis.

Moreover, we observed that as obstacle density increase, the gain in roadmap utilisation

and space efficiency relative to the baseline slightly declined as shown by the downwards sloping trend of the green bar in Figure 5.7a. Under a lower obstacle density environment, the proposed scheme brings 461.35% and 431.75% improvements in roadmap utilisation and space efficiency relative to the grid-based method, while suffering from a 41.56% decrease in flowtime performance. This is a larger performance gained compared with the achievements in the high-density environment. This is because, in a more challenging environment, it is more difficult to improve utilisation while balancing flowtime.

Last but not least, as the number of agents increase, waittime performance declines under the environment with the same density for the Directed Voronoi scheme. This can be observed from the upward trend of the orange bars in Figure 5.9, the statistic within the red patches did not follow the trend due to incomplete solutions. This is as expected because congestion is more likely to occur when there are more agents in the environment. Moreover, our proposed scheme may be more useful in resolving congestion compared to the Directed Voronoi scheme. We gathered the following evidences. Firstly, although there is no obvious trend of declining waittime performance under the Tunnel environment in Figure 5.9 for the proposed scheme, in Figure 5.6, its waittime performance improved as the agents' number increased under the Room and Narrow Opening environment, as indicated by the downward trends of the green bars. Secondly, when the scheme transform from the low-density environment to the high-density environment, the proposed scheme also improved the average waittime performance from having a loss of 9.62 to 6.96, i.e. a 27.65% improvement, as shown in the green bar in Figure 5.7b. This hinted that the proposed scheme is also useful in resolving congestion, such that despite the increase in obstacle density or the number of agents, the average waittime decrease.

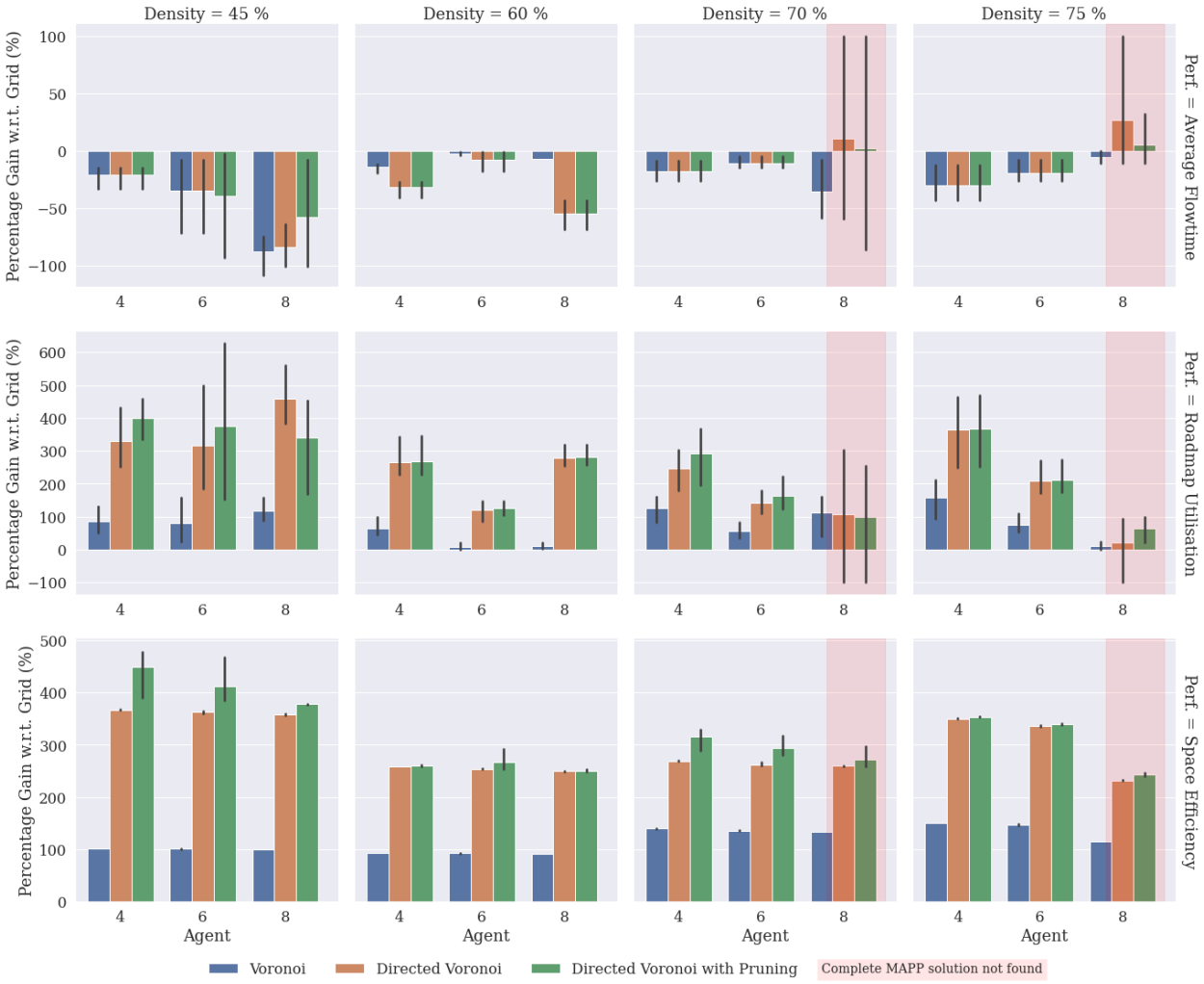


Figure 5.8: Performance plot of the parameterisation schemes on flowtime, roadmap utilisation and space efficiency under environment of different obstacle density. Note that higher percentage gain means better performance. As the number of agents increase, there is a general trend of declining performance gain in space efficiency and roadmap utilisation.

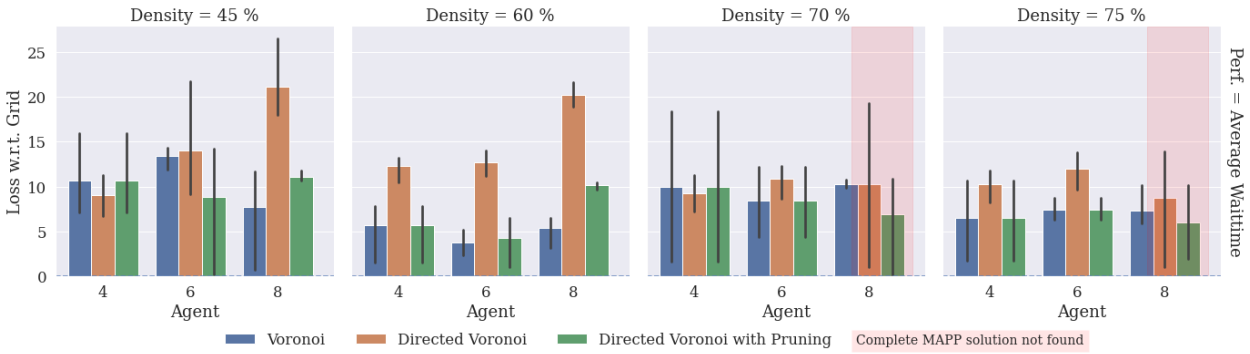


Figure 5.9: Performance plot of the parameterisation schemes on average waittime under environment of different obstacle density. Note that lower loss means better performance.

5.3.5 Relationship between Connectivity and Existence of Edges

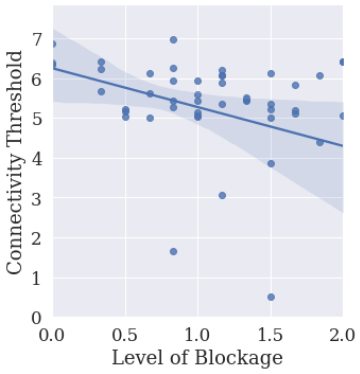


Figure 5.10: The connectivity threshold is negatively correlated with the level of blockage in the environment. This implies that as the environment become denser, the requirement for retaining an edge relaxed.

Based on the research of walkability planning [57, 58, 11, 12, 13] discussed in Section 4.2, we hypothesis that the total number of nodes connected to either end of the edges are correlated with the popularity and hence usage of the roads. We further defined the term Connectivity Threshold in Equation 4.5 to show the minimum number of connections required to retain the edges. Nevertheless, the global connectivity will be protected due to some of the mitigation measures: the penalty cost will ensure the optimiser to avoid incomplete solution and only eligible edges will be considered for pruning, where eligibility refers to edges with length ≤ 3 and that they are not connected to any starting or ending locations.

In this experiment, we would like to show the relationship between the Connectivity Threshold and the level of blockage $l_{blockage}$ in the environment. $l_{blockage}$ refers to the impact brought by increasing number of agents and the obstacle density:

$$l_{blockage} = Normalised(\text{Agents}) + Normalised(\text{Obstacle Density}) \tag{5.4}$$

Note that the terms has been normalised to generate a non-bias plot. The original range of the number of agents and the obstacle density was $2 - 8$ and $45\% - 75\%$ respectively. The data were collected based on the Tunnels Map (Data Series B).

Figure 5.10 illustrated that with increasing blockages in the environment, the minimum number of connections required to retain the edges reduced. This matches with our predictions on the trend of the threshold. When there are more agents or the level of obstacle density is high, more area of the roadmap was utilised for the agents to reach their goals, hence, fewer edges could be pruned without adversely impacting the completeness of the MAPP solution. This causes the relaxation in the requirements for retaining an edge, hence, the downward trend observed in the figure.

The observations coincide with our predictions reflects that our assumptions on the relationship between the connectivity and the existence of edges hold some truth. Moreover, more investigations would be required to validate it, which will be left as future work.

Chapter 6

Conclusion

With the rapid development of cities and the increasing popularity of deploying robot teams in these environments, there are growing calls for designers to consider environmental resources utilisation when developing multi-agent systems. Furthermore, although environment representation and roadmap optimisation are closely related and fall under the same umbrella of environment optimisation, most literature [28, 29, 9, 40, 41, 42, 8, 2] did not considered them together.

Therefore, in this work, we present a novel utilisation-aware MAPP environment optimisation framework that unites the two fields. It includes using a directed Voronoi diagram as the base roadmap and utilises GP with BO to optimise the direction and usage preferences of edges. It also features an original edge pruning scheme inspired by walkability planning research [11, 12, 13], which removes edges based on connectivity. Moreover, the primary hypothesis of the project is that environmental resources utilisation in multi-agent systems can be improved by using topology-based environment parameterisation schemes that include directions and usages specifications. The experiments findings support the claim. Our framework successfully brings 574.78% and 386.85% improvements in roadmap utilisation and space efficiency compared to the grid method, at a cost of 88.09% decrease in flowtime performance. This matches our hypothesis that topology-based roadmaps are beneficial to environmental resources utilisation. On top of that, its results on roadmap utilisation and space efficiency is 2.42 and 3.27 times better than that of using a basic Voronoi roadmap, while sacrificing 12.90% less flowtime performance. This shows that adding direction specifications brings positive benefits. In addition, comparing the proposed scheme with a similar scheme that exclude usage specifications, they achieved similar performance gain in environmental resources utilisation but our scheme traded off 14.46% less flowtime performance. These confirm the second part of the primary hypothesis that there is potential gain in adopting direction and usage-guided roads.

Furthermore, the proposed scheme was tested under environments of different shape and different obstacle density. We identified that the proposed scheme offers the highest

performance gain in roadmap utilisation in Room, best space efficiency in Tunnel and good at resolving congestion in Narrow Openings. This highlighted the strength and weakness of the scheme, for instance, encouraging agents to spread out is more useful in resolving congestion at narrow opening but not at tunnels. Moreover, despite its limitations, the proposed scheme still achieved the best overall performance relative to other schemes (Directed Voronoi, Voronoi) under the 75% obstacle density environment. Relative to the grid baseline, it still brings 357% and 326% improvements in roadmap utilisation and space efficiency, at a cost of 34% decrease in flowtime performance.

In addition, we followed the second research objective and considered various tools for the optimisation problem. GP with BO with low sample number was chosen as the optimisation strategy as it manages to double the performance of ANN with only one-forty of the samples, which equivalent to saving 94% of the computational time. We also show that the strategy of varying the local cost function to achieve various global objectives works as we have anticipated. Finally, it has also been observed that the subgraph threshold declines as expected when the level of blockage in the environment increase. This shows that the hypothetical relationship between the connectivity and the existence of edges may hold some truth, but further investigations are required to verify it.

Last but not least, our research also filled the gap in the literature. Compared with R. Geraerts (2010) [8] work on representing environment as explicit corridor maps, we considered optimising the travelling cost of edges in the map as well. Compared with C. Henkel and M. Toussaint (2020) [9] work on directed roadmap, our project relaxed the environment representation from dense grid structure to topology graph and considered resources utilisation. Compared with A. Kleiner et al. (2011) [25] work on Voronoi-based roadmap, we enhanced the framework by incorporating an edge pruning scheme and further considered the utilisation of free space.

In conclusion, we showcase the potentials of topology-based roadmaps and the benefits of optimising the directions and usage of roads. We hope that our work could shine some lights on the development of utilisation-aware MAPP frameworks, which has shown to be in demand in the modern era of automation.

6.1 Limitations and Future Work

We acknowledge that there are several limitations regarding the framework and would like to present directions for future work as well.

First of all, the scalability of the current implementation lies on the lower end of the spectrum. The framework is only able to produce a partial solution for some of the cases that involve 8 agents. This may be due to the tight constraints on the traversable space. Therefore, more investigation should be conducted on improving and accelerating the

computation of "corridors".

Secondly, the framework has been constrained by predefined tasks, which means each optimisation loop only improved the roadmap for a predefined set of start and goal locations. Therefore, future work could also look into learning-based methods to generalise the framework. Using our framework to generate expert data, we can develop a probabilistic model to produce the optimum weights for the roadmap given any start and target locations.

Although limitations exist, it is hoped that this piece of work can promote different aspects of environment optimisation and gather attention on the significance of utilisation-aware solution.

References

[1] Hoshang Kolivand and Mohd Shahrizal Sunar. A survey on shadow volume in computer graphics. *IETE Technical Review*, 4, 01 2013. Cited on pages 7 and 1.

[2] W. Toll, Atlas F. Cook, M. V. Kreveld, and R. Geraerts. The explicit corridor map: Using the medial axis for real-time path planning and crowd simulation. In *Symposium on Computational Geometry*, 2016. Cited on pages 7, 1, 9, and 53.

[3] Minghe Hu. Alibaba launches logistics robot for last-mile deliveries to lower costs and as pandemic pushes automation. *South China Morning Post*. Cited on pages 7 and 1.

[4] Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Eli Boyarski, and Roman Bartak. Multi-agent pathfinding: Definitions, variants, and benchmarks, 2019. Cited on pages 7, 2, 6, 10, and 14.

[5] Daniel D. Corkill. Hierarchical planning in a distributed environment. San Francisco, CA, USA, 1979. Morgan Kaufmann Publishers Inc. Cited on pages 7, 2, 5, and 10.

[6] Prajakta Desai, Seng W. Loke, Aniruddha Desai, and Jugdutt Singh. Caravan: Congestion avoidance and route allocation using virtual agent negotiation. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1197–1207, 2013. Cited on pages 7, 2, 5, 6, and 10.

[7] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015. Cited on pages 7, 2, 6, 10, and 18.

[8] Roland Geraerts. Planning short paths with clearance using explicit corridors. In *2010 IEEE International Conference on Robotics and Automation*, pages 1997–2004, 2010. Cited on pages 7, 9, 53, and 54.

[9] Christian Henkel and Marc Toussaint. Optimized directed roadmap graph for multi-agent path finding using stochastic gradient descent, 2020. Cited on pages 7, 2, 6, 8, 9, 11, 53, and 54.

[10] T. Alpcan and T. Basar. A game-theoretic framework for congestion control in general topology networks. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 2, pages 1218–1224 vol.2, 2002. Cited on pages 7, 2, 6, 9, and 11.

[11] Sarah Elshahat, Michael O'Rorke, and Deepti Adlakha. Built environment correlates of physical activity in low- and middle-income countries: A systematic review. *PLOS ONE*, 15(3):1–19, 03 2020. Cited on pages 7, 25, 50, and 53.

[12] Adriano A. F. Hino, Rodrigo S. Reis, Olga L. Sarmiento, Diana C. Parra, and Ross C. Brownson. Built environment and physical activity for transportation in adults from curitiba, brazil. *Journal of Urban Health*, 91(3):446–462, Jun 2014. Cited on pages 7, 25, 50, and 53.

[13] Luis F. Gómez, Diana C. Parra, David Buchner, Ross C. Brownson, Olga L. Sarmiento, José D. Pinzón, Mauricio Ardila, José Moreno, Mauricio Serrato, and Felipe Lobelo. Built environment attributes and walking patterns among the elderly population in bogotá. *American Journal of Preventive Medicine*, 38(6):592–599, 2010. Cited on pages 7, 25, 50, and 53.

[14] International Federation of Robotics Press Conference 24th September 2020 Frankfurt. `https://ifr.org/downloads/press2018/Presentation_WR_2020.pdf`. Accessed: 2021-05-25. Cited on page 1.

[15] Joel Rodrigues, Federico Bergenti, and Agostino Poggi. *Developing Smart Emergency Applications with Multi-Agent Systems*, pages 31–44. 01 2012. Cited on page 1.

[16] Nathan Sturtevant and Robert Geisberger. A comparison of high-level approaches for speeding up pathfinding. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 6(1), Oct. 2010. Cited on page 1.

[17] Jur van den Berg, Rajat Shah, Arthur Huang, and Kenneth Goldberg. Ana*: Anytime nonparametric a. *Proceedings of the National Conference on Artificial Intelligence*, 1, 05 2012. Cited on page 1.

[18] N.A. Lang, J.M. Moonen, F. Srour, and Rob Zuidwijk. Multi agent systems in logistics: A literature and state-of-the-art review. *Erasmus Research Institute of Management (ERIM), ERIM is the joint research institute of the Rotterdam School of Management, Erasmus University and the Erasmus School of Economics (ESE) at Erasmus Uni, Research Paper*, 01 2008. Cited on page 1.

[19] Amir Mahmudzadeh, Mahmoud Ghorbani, and Ali Hakimelahi. Providing an emergency evacuation model for the stadium. *Transportation Research Procedia*, 48:620–631,

2020. Recent Advances and Emerging Issues in Transport Research – An Editorial Note for the Selected Proceedings of WCTR 2019 Mumbai. Cited on page 1.

[20] Olaf Jahn, Rolf Möhring, Andreas Schulz, and Nicolas Stier-Moses. System-optimal routing of traffic flows with user constraints in networks with congestion. *Operations Research*, 53, 02 2004. Cited on page 1.

[21] Wanli Chang, Samarjit Chakraborty, and Anuradha Annaswamy. Utilization-aware adaptive back-pressure traffic signal control. 10 2015. Cited on page 2.

[22] Manuel Mühlig Charlie Street, Bruno Lacerda and Nick Hawes. Multi-robot planning under uncertainty with congestion-aware models. 2020. Cited on pages 2 and 7.

[23] Sejoon Lim and Daniela Rus. Congestion-aware multi-agent path planning: Distributed algorithm and applications. *The Computer Journal*, 57(6):825–839, 2014. Cited on pages 2, 6, and 9.

[24] Gilbert Laporte and Silvano Martello. The selective travelling salesman problem. *Discrete Applied Mathematics*, 26(2):193–207, 1990. Cited on page 2.

[25] Alexander Kleiner, Dali Sun, and Daniel Meyer-Delius. Armo: Adaptive road map optimization for large robot teams. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3276–3282, 2011. Cited on pages 2, 7, 10, 11, and 54.

[26] Christos Voudouris and Edward Tsang. Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research*, 113(2):469–499, 1999. Cited on page 2.

[27] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning, 2011. Cited on pages 5 and 8.

[28] Hanan Samet and Robert E Webber. Storing a collection of polygons using quadtrees. *ACM Transactions on Graphics July, 1985. vol. 4: pp. 182-222 : some ill. includes bibliography.*, 4, 07 1985. Cited on pages 8 and 53.

[29] C. Faloutsos, H.V. Jagadish, and Yannis Manolopoulos. Analysis of the n-dimensional quadtree decomposition for arbitrary hyperrectangles. *Knowledge and Data Engineering, IEEE Transactions on*, 9:373 – 383, 06 1997. Cited on pages 8 and 53.

[30] J.P. van den Berg and M.H. Overmars. Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 1, pages 453–460 Vol.1, 2004. Cited on page 8.

[31] Tomás Lozano-Pérez and Michael A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22(10):560–570, October 1979. Cited on page 8.

[32] Yong K. Hwang and Narendra Ahuja. Gross motion planning—a survey. *ACM Comput. Surv.*, 24(3):219–291, September 1992. Cited on page 8.

[33] James A. Storer and John H. Reif. Shortest paths in the plane with polygonal obstacles. *J. ACM*, 41(5):982–1012, September 1994. Cited on page 8.

[34] P. M. Gruber. *Convex and Discrete Geometry*. Springer-Verlag New York, 2007. Cited on pages 8 and 16.

[35] Hugo Ledoux. Computing the 3d voronoi diagram robustly: An easy explanation. In *4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2007)*, pages 117–129, 2007. Cited on page 8.

[36] L. Paul Chew and Robert L. (Scot) Dyrsdale. Voronoi diagrams based on convex distance functions. In *Proceedings of the First Annual Symposium on Computational Geometry*, SCG '85, page 235–244, New York, NY, USA, 1985. Association for Computing Machinery. Cited on page 8.

[37] S. Huber and Peter Palfrader. Generalized offsetting using a variable-radius voronoi diagram. 2015. Cited on page 9.

[38] Martin Held, Stefan Huber, and Peter Palfrader. Generalized offsetting of planar structures using skeletons. *Computer-Aided Design and Applications*, 13(5):712–721, 2016. Cited on page 9.

[39] Franco P. Preparata and Michael Shamos. *Computational Geometry An Introduction*. Monographs in Computer Science. Springer-Verlag New York, 1985. Cited on page 9.

[40] A. Kamphuis and M. Overmars. Finding paths for coherent groups using clearance. pages 19–28, 08 2004. Cited on pages 9 and 53.

[41] Julien Pettre, Pablo de Heras Ciechomski, Jonathan Maïm, Barbara Yersin, Jean-Paul Laumond, and Daniel Thalmann. Real-time navigating crowds: Scalable simulation and rendering. *Computer Animation and Virtual Worlds*, 17:445–455, 07 2006. Cited on pages 9 and 53.

[42] Roland Geraerts and Mark H. Overmars. The corridor map method: A general framework for real-time high-quality path planning: Research articles. *Comput. Animat. Virtual Worlds*, 18(2):107–119, May 2007. Cited on pages 9 and 53.

[43] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric Xing. Neural architecture search with bayesian optimisation and optimal transport, 2019. Cited on page 10.

[44] Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. Neural architecture search using bayesian optimisation with weisfeiler-lehman kernel. 06 2020. Cited on page 10.

[45] J. Kocijan, B. Banko, B. Likar, A. Girard, R. Murray-Smith, and C.E. R. Rasmussen. A case based comparison of identification with neural networks and gaussian process models. In M.G. Ruano, P.J. Fleming, and A.E. Ruano, editors, *Intelligent Control Systems and Signal Processing 2003: (ICONS 2003): A Proceedings Volume from the IFAC International Conference, Faro, Algarve, Portugal, 8-11 April 2003*. International Federation of Automatic Control, Oxford, UK, 2003. Cited on page 10.

[46] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, May 2005. Cited on page 13.

[47] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009. Cited on page 13.

[48] D. Ferguson, M. Darms, C. Urmson, and S. Kolski. Detection, prediction, and avoidance of dynamic obstacles in urban environments. *2008 IEEE Intelligent Vehicles Symposium*, pages 1149–1154, 2008. Cited on page 14.

[49] C. Fulgenzi, A. Spalanzani, and C. Laugier. Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1610–1616, 2007. Cited on page 14.

[50] Hang Ma, Daniel Harabor, Peter J. Stuckey, Jiaoyang Li, and Sven Koenig. Searching with consistent prioritization for multi-agent path finding, 2018. Cited on page 14.

[51] Liu Liu and Sisi Zlatanova. An approach for indoor path computation among obstacles that considers user dimension. *ISPRS International Journal of Geo-Information*, 4:2821–2841, 12 2015. Cited on page 15.

[52] Mir Abolfazl Mostafavi, Leila Hashemi Beni, and Karine Hins-Mallet. Geosimulation of geographic dynamics based on voronoi diagram. *Transactions on Computational Science*, 9:183–201, 01 2010. Cited on page 17.

[53] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006. Cited on page 19.

[54] Riccardo Moriconi, Marc Peter Deisenroth, and K. S. Sesh Kumar. High-dimensional bayesian optimization using low-dimensional feature spaces. *Machine Learning*, 109(9):1925–1943, Sep 2020. Cited on page 20.

[55] Arthur Arnx. Understand and create a Perceptron. `https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf`. Accessed: 2021-06-10. Cited on page 20.

[56] Philip Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235, 1969. Cited on page 21.

[57] Ramit Debnath, Gianna Monteiro Farias Simoes, Ronita Bardhan, Solange Maria Leder, Roberto Lamberts, and Minna Sunikka-Blank. Energy justice in slum rehabilitation housing: An empirical exploration of built environment effects on socio-cultural energy demand. *Sustainability*, 12(7), 2020. Cited on pages 25 and 50.

[58] Arnab Jana, Ronita Bardhan, Sayantani Sarkar, and Vaibhav Kumar. Framework to assess and locate affordable and accessible housing for developing nations: Empirical evidences from mumbai. *Habitat International*, 57:88–99, 2016. Cited on pages 25 and 50.

[59] N. Sturtevant. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games*, 4(2):144 – 148, 2012. Cited on pages 29 and 30.

[60] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE*, 22(4):469–483, 1996. Cited on page 31.

[61] The SciPy community. Scipy Spatial Voronoi. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.Voronoi.html`. Accessed: 2021-06-10. Cited on page 31.

[62] Sheffield Machine Learning Group. GPy: a Gaussian processes framework in python. `https://sheffieldml.github.io/GPy/`. Accessed: 2021-06-10. Cited on page 35.

[63] Francois Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015. Cited on page 35.

[64] The SciPy community. Scipy Optimize Minimize. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html`. Accessed: 2021-06-10. Cited on pages 35 and 65.

[65] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, December 1997. Cited on page 35.

[66] S. Nash. Newton-type minimization via the lanczos method. *SIAM Journal on Numerical Analysis*, 21:770–788, 1984. Cited on page 36.

[67] J Beau W Webber. A bi-symmetric log transformation for wide-range data. *Measurement Science and Technology*, 24(2):027001, dec 2012. Cited on page 39.

[68] Robert. Matlab symlog. `https://www.mathworks.com/matlabcentral/fileexchange/57902-symlog`. Cited on page 39.

Appendix A

Hyperparmeters of L-BFGS algorithms

The hyperparmeters used in L-BFGS algorithms are as follows. We only highlight the one that are different from the default implementations [64] and the reason for changing them is to accelerate the algorithm convergence.

Parameters	**Value**	**Remarks**
ftol	1.220446049250313	The iteration stops when $(f^k - f^{k+1})/max\|f^k\|, \|f^{k+1}\|, 1 <= ftol$
maxfun	500	Maximum number of function evaluations.
maxiter	500	Maximum number of iterations.

Table A.1: Hyperparmeters of L-BFGS algorithms

Appendix B

Results Tables

In the following tables, V, DV and DVP represents the Voronoi, Directed Voronoi and Directed Voronoi with Pruning schemes. "FT", "Util.", "Space Effi." and "W" are abbreviations for average flowtime, roadmap utilisation, space efficiency and Average Waittime. Note that for the Average Waittime, contract to the Figures, "Gain w.r.t Grid" instead of "Loss" was listed. This is to synchronised the theme of the table such that more positive means higher performance for all the values.

		10 Samples	**50 Samples**	**100 Samples**	**500 Samples**	**1000 Samples**	**2000 Samples**
4 ag.	**ANN**	60.39 % ± 31.89	75.60 % ± 0.74	74.47 % ± 2.85	55.55 % ± 6.16	35.26 % ± 7.63	-7.68 % ± 13.60
	BO	92.10 % ± 0.69	74.00 % ± 6.68	39.52 % ± 17.61	-925.58 % ± 121.73	nan % ± nan	nan % ± nan
8 ag.	**ANN**	75.68 % ± 1.34	45.53 % ± 9.24	9.22 % ± 18.69	-282.55 % ± 94.48	-644.60 % ± 191.41	-1374.39 % ± 376.32
	BO	87.38 % ± 3.12	-70.69 % ± 114.73	-371.01 % ± 198.34	-3054.13 % ± 362.58	nan % ± nan	nan % ± nan

Table B.1: Percentage Gain in Computational Time relative to Naive Optimiser

		Room		
		V	**DV**	**DVP**
4 agents	**FT (%)**	-77.09 % ± 37.94	-77.09 % ± 37.94	-77.09 % ± 37.94
	Util. (%)	289.71 % ± 84.44	800.91 % ± 198.08	869.70 % ± 206.66
	Space Effi. (%)	119.99 % ± 0.53	408.33 % ± 2.84	448.39 % ± 51.81
	W (unit)	-16.06 ± 9.58	-5.83 ± 4.50	-16.06 ± 9.58
6 agents	**FT (%)**	-83.66 % ± 36.18	-59.13 % ± 26.61	-59.13 % ± 26.61
	Util. (%)	238.83 % ± 22.79	625.66 % ± 80.86	686.23 % ± 159.59
	Space Effi. (%)	118.94 % ± 0.77	402.76 % ± 4.05	441.56 % ± 54.23
	W (unit)	-13.28 ± 4.29	-7.61 ± 3.60	-12.86 ± 4.03
8 agents	**FT (%)**	-81.23 % ± 30.23	-81.23 % ± 30.23	-41.75 % ± 14.09
	Util. (%)	213.05 % ± 34.54	614.80 % ± 78.76	537.34 % ± 89.16
	Space Effi. (%)	117.98 % ± 0.63	397.73 % ± 3.30	402.77 % ± 6.84
	W (unit)	-11.50 ± 0.82	-12.25 ± 4.63	-11.11 ± 0.80

Table B.2: Percentage Gain and Gain attained by various Environment Parameterisation Schemes w.r.t. Grid under Room Environment

Narrow Opening

		V	DV	DVP
4 agents	**FT (%)**	-330.08 % ± 67.09	-337.80 % ± 68.75	-269.55 % ± 185.00
	Util. (%)	682.14 % ± 190.52	1219.45 % ± 332.92	1050.42 % ± 538.73
	Space Effi. (%)	125.28 % ± 1.77	279.53 % ± 0.25	299.71 % ± 22.04
	W (unit)	-13.81 ± 11.55	-23.08 ± 6.95	-9.89 ± 9.52
6 agents	**FT (%)**	-264.63 % ± 166.50	-302.22 % ± 142.94	-304.12 % ± 140.41
	Util. (%)	345.90 % ± 100.27	642.54 % ± 167.32	656.61 % ± 170.50
	Space Effi. (%)	124.26 % ± 0.20	273.42 % ± 0.55	280.50 % ± 0.57
	W (unit)	-9.80 ± 6.68	-29.06 ± 3.09	-5.86 ± 6.21
8 agents	**FT (%)**	30.94 % ± 100.78	36.95 % ± 109.21	36.95 % ± 109.21
	Util. (%)	20.47 % ± 208.66	98.43 % ± 343.70	102.13 % ± 350.10
	Space Effi. (%)	122.94 % ± 0.00	266.30 % ± 0.80	268.58 % ± 4.74
	W (unit)	-3.64 ± 6.30	-5.58 ± 7.94	-3.48 ± 6.04

Table B.3: Percentage Gain and Gain attained by various Environment Parameterisation Schemes w.r.t. Grid under Narrow Opening Environment

Tunnel

		V	DV	DVP
4 agents	**FT (%)**	-20.55 % ± 10.75	-20.55 % ± 10.75	-20.55 % ± 10.75
	Util. (%)	86.22 % ± 40.24	330.51 % ± 92.32	399.61 % ± 63.21
	Space Effi. (%)	102.03 % ± 0.25	367.18 % ± 1.16	448.94 % ± 50.62
	W (unit)	-10.66 ± 4.63	-9.00 ± 2.25	-10.66 ± 4.63
6 agents	**FT (%)**	-34.44 % ± 33.17	-34.44 % ± 33.17	-39.56 % ± 47.28
	Util. (%)	80.44 % ± 70.30	315.87 % ± 162.76	376.41 % ± 237.49
	Space Effi. (%)	101.30 % ± 0.57	363.81 % ± 2.72	413.10 % ± 47.02
	W (unit)	-13.39 ± 1.30	-14.00 ± 6.71	-8.81 ± 7.49
8 agents	**FT (%)**	-87.87 % ± 17.89	-84.15 % ± 18.48	-57.48 % ± 46.57
	Util. (%)	117.54 % ± 35.34	457.34 % ± 91.71	341.53 % ± 152.30
	Space Effi. (%)	100.33 % ± 0.22	358.48 % ± 1.15	378.45 % ± 1.25
	W (unit)	-7.75 ± 6.08	-21.12 ± 4.68	-11.12 ± 0.55

Table B.4: Percentage Gain and Gain attained by various Environment Parameterisation Schemes w.r.t. Grid under Tunnel Environment

45 % Obstacle Density

		V	DV	DVP
4 agents	**FT (%)**	-20.55 % ± 10.75	-20.55 % ± 10.75	-20.55 % ± 10.75
	Util. (%)	86.22 % ± 40.24	330.51 % ± 92.32	399.61 % ± 63.21
	Space Effi. (%)	102.03 % ± 0.25	367.18 % ± 1.16	448.94 % ± 50.62
	W (unit)	-10.66 ± 4.63	-9.00 ± 2.25	-10.66 ± 4.63
6 agents	**FT (%)**	-34.44 % ± 33.17	-34.44 % ± 33.17	-39.56 % ± 47.28
	Util. (%)	80.44 % ± 70.30	315.87 % ± 162.76	376.41 % ± 237.49
	Space Effi. (%)	101.30 % ± 0.57	363.81 % ± 2.72	413.10 % ± 47.02
	W (unit)	-13.39 ± 1.30	-14.00 ± 6.71	-8.81 ± 7.49
8 agents	**FT (%)**	-87.87 % ± 17.89	-84.15 % ± 18.48	-57.48 % ± 46.57
	Util. (%)	117.54 % ± 35.34	457.34 % ± 91.71	341.53 % ± 152.30
	Space Effi. (%)	100.33 % ± 0.22	358.48 % ± 1.15	378.45 % ± 1.25
	W (unit)	-7.75 ± 6.08	-21.12 ± 4.68	-11.12 ± 0.55

Table B.5: Percentage Gain and Gain attained by various Environment Parameterisation Schemes w.r.t. Grid under Environment with 45% Obstacle Density

60 % Obstacle Density

		V	DV	DVP
4 agents	**FT (%)**	-13.94 % ± 4.66	-31.43 % ± 7.93	-31.43 % ± 7.93
	Util. (%)	63.21 % ± 29.78	265.38 % ± 67.11	267.65 % ± 68.56
	Space Effi. (%)	93.66 % ± 0.07	259.02 % ± 0.22	261.13 % ± 1.95
	W (unit)	-5.73 ± 3.61	-12.33 ± 1.59	-5.73 ± 3.61
6 agents	**FT (%)**	-1.90 % ± 1.92	-7.54 % ± 8.96	-7.54 % ± 8.96
	Util. (%)	7.69 % ± 12.59	119.10 % ± 31.67	126.49 % ± 21.15
	Space Effi. (%)	92.25 % ± 0.23	253.96 % ± 0.37	267.88 % ± 21.33
	W (unit)	-3.82 ± 2.41	-12.72 ± 1.44	-4.24 ± 2.85
8 agents	**FT (%)**	-6.55 % ± 0.03	-54.95 % ± 12.65	-54.95 % ± 12.65
	Util. (%)	9.23 % ± 10.72	279.28 % ± 34.16	280.30 % ± 33.08
	Space Effi. (%)	90.96 % ± 0.15	249.63 % ± 0.88	250.64 % ± 2.31
	W (unit)	-5.39 ± 1.96	-20.17 ± 1.38	-10.14 ± 0.46

Table B.6: Percentage Gain and Gain attained by various Environment Parameterisation Schemes w.r.t. Grid under Environment with 60% Obstacle Density

70 % Obstacle Density

		V	DV	DVP
4 agents	**FT (%)**	-17.77 % ± 9.05	-17.77 % ± 9.05	-17.77 % ± 9.05
	Util. (%)	124.56 % ± 39.70	245.79 % ± 61.51	291.42 % ± 87.25
	Space Effi. (%)	139.80 % ± 0.53	269.21 % ± 1.25	315.52 % ± 23.24
	W (unit)	-10.00 ± 8.33	-9.25 ± 2.00	-10.00 ± 8.33
6 agents	**FT (%)**	-10.60 % ± 5.56	-10.60 % ± 5.56	-10.60 % ± 5.56
	Util. (%)	57.20 % ± 24.32	141.31 % ± 35.38	164.00 % ± 52.93
	Space Effi. (%)	136.03 % ± 0.75	262.64 % ± 3.33	294.75 % ± 19.85
	W (unit)	-8.46 ± 3.93	-10.89 ± 2.10	-8.46 ± 3.93
8 agents	**FT (%)**	-35.54 % ± 25.56	10.94 % ± 81.34	1.96 % ± 93.57
	Util. (%)	111.01 % ± 62.87	106.19 % ± 201.44	97.75 % ± 180.85
	Space Effi. (%)	133.32 % ± 0.37	259.54 % ± 0.72	271.99 % ± 21.55
	W (unit)	-10.25 ± 0.50	-10.25 ± 9.13	-6.92 ± 6.02

Table B.7: Percentage Gain and Gain attained by various Environment Parameterisation Schemes w.r.t. Grid under Environment with 70% Obstacle Density

75 % Obstacle Density

		V	DV	DVP
4 agents	**FT (%)**	-30.27 % ± 16.01	-30.27 % ± 16.01	-30.27 % ± 16.01
	Util. (%)	158.15 % ± 60.07	364.59 % ± 108.61	368.36 % ± 109.50
	Space Effi. (%)	150.21 % ± 0.38	350.22 % ± 1.22	353.86 % ± 1.24
	W (unit)	-6.55 ± 4.49	-10.25 ± 1.80	-6.55 ± 4.49
6 agents	**FT (%)**	-18.81 % ± 10.10	-18.81 % ± 10.10	-18.81 % ± 10.10
	Util. (%)	75.22 % ± 29.79	209.79 % ± 53.25	212.22 % ± 53.67
	Space Effi. (%)	146.86 % ± 1.84	336.36 % ± 1.74	339.79 % ± 1.76
	W (unit)	-7.41 ± 1.21	-12.00 ± 2.13	-7.41 ± 1.21
8 agents	**FT (%)**	-5.31 % ± 5.57	27.27 % ± 63.01	5.04 % ± 23.99
	Util. (%)	11.21 % ± 12.90	19.82 % ± 104.59	63.24 % ± 38.77
	Space Effi. (%)	114.65 % ± 0.40	232.20 % ± 0.96	243.13 % ± 3.90
	W (unit)	-7.34 ± 2.41	-8.75 ± 6.83	-6.00 ± 4.10

Table B.8: Percentage Gain and Gain attained by various Environment Parameterisation Schemes w.r.t. Grid under Environment with 75% Obstacle Density