Optimal Task Assignment and Collision Avoidance for Mobile Robots

Dimitri Hollosi

Mechanical Engineering Semester Project II

Under the supervision of:
Dr. Tony A. Wood and Prof. Maryam Kamgarpour

July 7, 2022

sycamore lab
SYSTEMS CONTROL AND MULTIAGENT OPTIMIZATION RESEARCH

École Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland

Abstract

This report serves as a general overview of the semester project conducted in the SYCAMORE lab during the Spring 2022 semester. It focuses on multi-agent optimal task assignment methods, which have been implemented on state of the art simulation methods using frameworks such as ROS2 and Gazebo. Different assignment problems, such as the Linear Sum Assignment Problem and the Lexicographic Bottleneck Assignment Problem, are implemented in this simulation platform. This enables a comparative assessment of the respective properties that can be derived from them, such as dynamic consistency and collision-avoidance. Visualisation methods are also generated as outputs to enable user-friendliness for the study of the various assignments.

Contents

Chapter 1

Introduction

1.1 Context and objective

Much interest has gone into the study of multi-robot systems in recent years. Combined with modern optimisation methods, on both a control and planning level, such systems open up a wide array of potential real-life applications. Particularly, the idea of assigning specific tasks (or goals) that agents must travel to, known as the assignment problem, can provide considerable benefits to a number of areas, such as:

- coordinated search and rescue missions,
- large scale agriculture,
- efficient transportation of people and supplies,
- heterogeneous swarms of robots (increasing topic of research, namely for the upcoming lunar missions driven by the private sector).

The task assignment problem, i.e allocating tasks to agents, is the first decision to be made on a planning level for the above mentioned applications. These problems can be written as combinatorial optimisation problems, which aim to minimise a function that depends on the costs related to the agents and tasks' locations. Examples of such cost functions, as well as the different methods and approaches to solving these problems will be outlined in the Literature Review (section 1.2). A particular type of assignment problem, known as the Bottleneck Assignment Problem (BAP) and its subclass, the Lexicographic-BAP, will be introduced and shall serve as the main focus for the remainder of this project report. The latter (Lexico-BAP) presents many advantages, one of them being the inherent collision-avoidance properties that can be derived from it. Other properties, such as dynamic consistency, shall also be covered, and whether the different assignment types meet these properties will be investigated. Despite it being a subject of high relevance, the actual method of solving these assignments problems will be presented but will not be covered extensively; the main objective at hand is to investigate the properties of the solutions of the different assignment problems.

The project also entails the creation of a simulation environment to test and analyse methods to coordinate and control a group of differential-wheeled robots using ROS2.

1.2 Literature review

There is a substantial body of literature investigating the task assignment problem, as it is a widespread topic of interest throughout a number of disciplines, including (navigational) robotics. Different kinds of task assignments can be attributed to the agents depending on what problem is being solved, and what is desired from it. One commonly sought after property is the collision-avoidance constraint, guaranteeing on a global-navigation perspective that agents will not collide with one another. Despite collision avoidance commonly being addressed in the agents' individual local navigation schemes/algorithms, the generation of collision-free assignment trajectories has the added benefit of relaxing some local navigation computation, which is generally solved online (done in real-time).

A brief overview of the different assignment problems considered for this project shall be presented in order to provide the reader with an understanding on what the different considered objective functions are, and what respective properties can be derived from them. Specifically, the property of an assignment remaining the same as the initially computed optimal assignment, while en-route to the task, is known as dynamic consistency and shall be introduced. These problems can generally be solved either by a centralised entity or alternatively in a distributed fashion.

1.2.1 Linear Sum Assignment Problem

The most common assignment problem is known as the Linear Sum Assignment Problem (LSAP) [2], which aims to minimise the sum of individual costs for agent to task pairs. Given a set of agents $\mathcal{A} = \{\alpha_i\}_{i=1}^{n}$ and tasks $\mathcal{T} = \{\tau_j\}_{j=1}^{n}$, the mathematical model of this problem can be represented by the following relation

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\phi_{ij} \tag{1.1}$$

$$\begin{aligned} s.t \quad & \sum_{j=1}^{n} \phi_{ij} = 1 \quad i \in \mathcal{A}, \\ & \sum_{i=1}^{n} \phi_{ij} = 1 \quad j \in \mathcal{T}. \end{aligned} \tag{1.2}$$

where w_{ij} is the cost, or weight, of moving an agent α_i to task τ_j (e.g. the distance between them); $\Phi = (\phi_{ij})$ is a binary matrix such that

$$\phi_{ij} = \begin{cases} 1 & \text{if agent i is assigned to task j} \\ 0 & \text{otherwise.} \end{cases} \tag{1.3}$$

As shown in (1.1) , the objective here it to minimise the *total* cost of each assignment. The constraints shown in (1.8) ensure that each task receives only one agent, and that only one agent is assigned to each task, respectively. The most efficient way of solving this problem in a centralised manner is with the Hungarian Algorithm (Kuhn, 1955; Munkres, 1957), also known as the Kuhn–Munkres algorithm. This being said, solving this algorithm for large systems may become computationally intensive [4].
For illustrative purposes, an example showing the obtained assignment between $m = 5$ agents and $n = 5$ tasks from the LSAP solution is shown in Figure 1.1, consequently highlighting the non-intersecting properties of the assigned agent-task pairs. The green edges in the graph view correspond to the assigned agent-task pairs. The edge weights represent the distance between the robots and the destinations.
Note: the graph view does not currently consider the true position of agents and tasks and is shown simply for completeness. Indeed, they were generated with the Networkx python package [13] and using the "spring-layout", which randomly selects a spread-out view at each compilation. Upcoming package updates may fix this readability inconvenience in the future.

1.2.2 Bottleneck Assignment Problem

Conversely to the LSAP, the Bottleneck Assignment Problem (BAP) is a specific type of the assignment problem where the largest individual agent-to-task cost (distance) is minimised. Maintaining the same constraints as 1.8, and denoting the set of all assignments that satisfy these constraints by $\mathcal{B}_{\mathcal{A},\mathcal{T}}$, the BAP is formulated as follows :

$$\min_{\Phi \in \mathcal{B}_{\mathcal{A},\mathcal{T}}} \quad \max_{(i,j) \in \mathcal{A} \times \mathcal{T}} w_{ij}\phi_{ij}. \tag{1.4}$$

A representation of such an assignment is shown in Figure 1.2, where it can be seen that the so-called *bottleneck* edge, linking Agent a_1 to Task p_3, is the maximum distance that any agents would have to travel. This type of assignment becomes particularly interesting for time sensitive applications; assuming the agents all travel at the same speed, the maximum time that it will take for all agents to reach their goals can therefore be derived from the bottleneck.

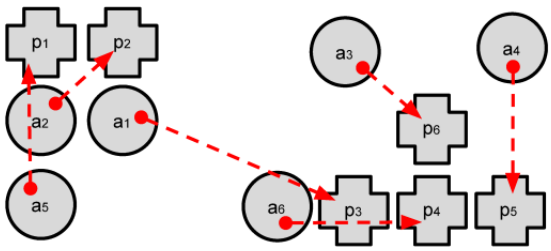
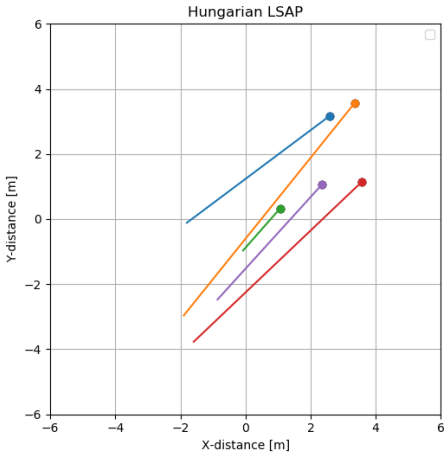
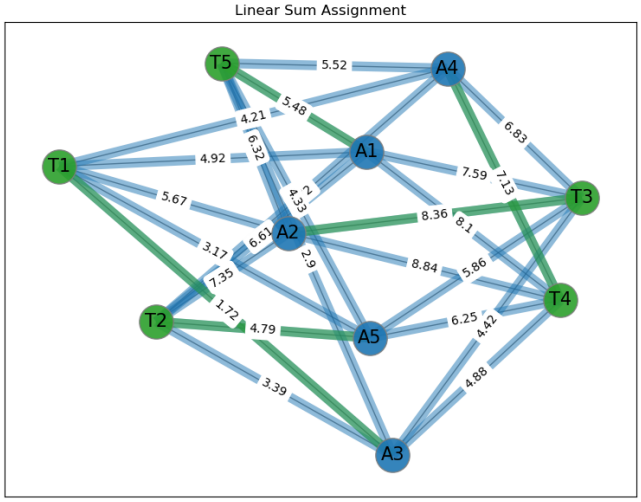


Figure 1.2: Role assignment problem with minimised largest distance (A1 - P3) (from [5]



(a) LSAP agent to task allocation



(b) Graph view with displayed costs

Figure 1.1: Example of LSAP agent-to-task pairing (5 agents and tasks)

1.2.3 Lexicographic Bottleneck Assignment Problem

The Lexicographic Bottleneck Assignment Problem, or Lexico-BAP, is a subclass of the BAP, where the largest assignments costs are sequentially minimised in decreasing hierarchical order. For simplicity, only the main results shall be shown in the aim of providing an intuitive understanding behind the sequential counterpart of the BAP, and how it can provide insights on the sensitivity of the BAP (taken from [6]). Consider a complete bipartite assignment graph $\mathcal{G} := (\mathcal{A}, \mathcal{T}, \mathcal{E})$, with the edge set $\mathcal{E} := \mathcal{A} \times \mathcal{T}$; the *assignments weights* $\mathcal{W} := \{w_{i,j}0 | (i,j) \in \mathcal{E}\}$.

Definition 3 (Robustness margin) [6]: Given the assignment weights $\mathcal{W}$, consider the complete bipartite graph formed by a subset of agents, $\bar{\mathcal{A}}$, and a subset of tasks, $\bar{\mathcal{T}}$, i.e sub-graph $\bar{\mathcal{G}} := (\bar{\mathcal{A}}, \bar{\mathcal{T}}, \bar{\mathcal{E}})$, with edges set $\bar{\mathcal{E}} = \bar{\mathcal{A}} \times \bar{\mathcal{T}}$. For $|\bar{\mathcal{E}}| > 1$, the set of the so-called *maximum-margin bottleneck edges* is defined as

$$e_{\bar{\mathcal{A}},\bar{\mathcal{T}}}(\mathcal{W}) := \arg \max_{(i,j) \in E_{\mathcal{A},\mathcal{T}}(\bar{\mathcal{E}},\mathcal{W})} \mathcal{B}_{\bar{\mathcal{A}},\bar{\mathcal{T}}}(\bar{\mathcal{E}} \backslash \{(i,j)\}, \mathcal{W}) \tag{1.5}$$

and the corresponding *robustness margin*

$$r_{\bar{\mathcal{A}},\bar{\mathcal{T}}}(\mathcal{W}) := \max_{(i,j) \in E_{\mathcal{A},\mathcal{T}}(\bar{\mathcal{E}},\mathcal{W})} \mathcal{B}_{\bar{\mathcal{A}},\bar{\mathcal{T}}}(\bar{\mathcal{E}} \backslash \{(i,j)\}, \mathcal{W}) - w_{i,j} \tag{1.6}$$

For $|\bar{\mathcal{A}}| = |\bar{\mathcal{T}}| = |\bar{\mathcal{E}}| = 1$, the maximum order bottleneck edge is set to be the singleton edge $e_{\bar{\mathcal{A}},\bar{\mathcal{T}}}(\mathcal{W}) = \bar{\mathcal{E}}$ and the robustness margin is assumed to be infinity, $r_{\bar{\mathcal{A}},\bar{\mathcal{T}}}(\mathcal{W}) = \infty$.

Definition 4 (Sequential bottleneck assignment) [6]: Consider the agents, $\mathcal{A}$, the tasks, $\mathcal{T}$, and the assignment weights, $\mathcal{W}$. An assignment, Π^* , is sequential bottleneck optimising if it is bottleneck minimising for the assignment graph, $\mathcal{G}$ = ($\mathcal{A}$, $\mathcal{T}$, $\mathcal{E}$), and the sequence of subgraphs $\bar{\mathcal{G}}_2, \bar{\mathcal{G}}_3, ..., \bar{\mathcal{G}}_n$, where $\bar{\mathcal{G}}_k = (\bar{\mathcal{A}}_k, \bar{\mathcal{T}}_k, \bar{\mathcal{E}}_k)$ is the complete bipartite graph of the subset of agents, $\bar{\mathcal{A}}_k \subset \mathcal{A}$, and the subset of tasks, $\bar{\mathcal{T}}_k \subset \mathcal{T}$, obtained by removing a maximum-margin bottleneck agent and task from $\bar{\mathcal{G}}_{k-1}$, i.e $\Pi^* \in \mathcal{S}_{\mathcal{A},\mathcal{T}}(\mathcal{W})$

$$\mathcal{S}_{\mathcal{A},\mathcal{T}}(\mathcal{W}) := \{\Pi \in \mathcal{B}_{\mathcal{A},\mathcal{T}} | \forall k \in \{1, ..., n\} \Pi \in \mathcal{B}_{\bar{\mathcal{A}}_k,\bar{\mathcal{T}}_k}(\bar{\mathcal{E}}_k, \mathcal{W}), \tag{1.7}$$

where $\bar{\mathcal{E}}_k = \bar{\mathcal{A}}_k \times \bar{\mathcal{T}}_k, \bar{\mathcal{A}}_1 = \mathcal{A}, \bar{\mathcal{T}}_1 = \mathcal{T}$,

$$\begin{aligned} \bar{\mathcal{A}}_k &= \bar{\mathcal{A}} \backslash \{i_1^*, ..., i_{k-1}^*\}, \\ \bar{\mathcal{T}}_k &= \bar{\mathcal{T}} \backslash \{i_1^*, ..., i_{k-1}^*\}. \end{aligned} \tag{1.8}$$

with the so-called *k-th order bottleneck edge*,

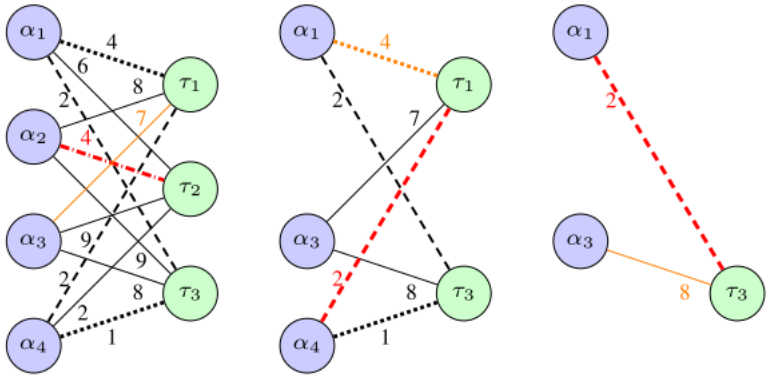
$$(i_k^*, j_k^*) \in e_{\bar{\mathcal{A}},\bar{\mathcal{T}}}(\mathcal{W}) \tag{1.9}$$

and *k-th order robustness margin*,

$$\mu_k = r_{\bar{\mathcal{A}},\bar{\mathcal{T}}}(\mathcal{W}) \tag{1.10}$$

As per definition 4, an assignment is a sequential bottleneck optimising assignment if it is a bottleneck minimising for the assignment graph

$\mathcal{G} := (\mathcal{A}, \mathcal{T}, \mathcal{E})$ and the sequence of subgraphs $\bar{\mathcal{G}}_k = (\bar{\mathcal{A}}_k, \bar{\mathcal{T}}_k, \bar{\mathcal{E}}_k)$, which is the complete bipartite graph of the subset of agents and tasks, $\bar{\mathcal{A}}_k \subset \mathcal{A}$ and $\bar{\mathcal{T}}_k \subset \mathcal{T}$ respectively.



(a) Order $k = 1$ (b) Order $k = 2$ (c) Order $k = 3$

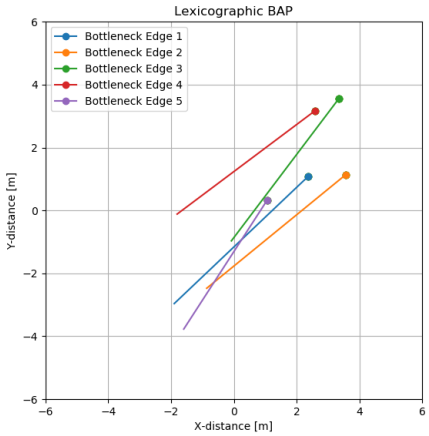
Figure 1.3: Sequential bottleneck assignments of agent to tasks (example, from [6]

As an illustrative example, consider the sequential bottleneck assignment shown in Figure 1.3. The k-th order botteleneck edge is represented in red, and the critical edge in orange. It follows that, for k=1, the bottleneck edge , the first order bottleneck is $\mathcal{B}_{\mathcal{A},\mathcal{T}}(\mathcal{E}\backslash\{(i,j)\}, \mathcal{W})$. From the above definitions, the k-th order robustness margin, $\mu_k = r_{\bar{\mathcal{A}},\bar{\mathcal{T}}}(\mathcal{W})$, thus the difference between the bottleneck and the critical edge. In this example, the first order bottleneck margin is therefore $\mu_1 = 7 - 4 = 3$, the second is $\mu_2 = 2$ and $\mu_3 = 6$. This can be considered as a quantification metric measuring how sensitive a particular assignment order is; the bottleneck edge weight can be increased by "up to" the bottleneck margin before the assignment may change. This assignment may be solved in a centralised fashion, as well as distributed [7].
The concept of robustness margins relate well to that of dynamic consistency [5], which is the property that an assignment remains the same as the initial optimal assignment throughout the travelling of agents to their respective destinations. Despite this being a sought after property, it does raise the question as to how *sensitive* this condition is to being verified, i.e how robust it is to potential trajectory deviations.

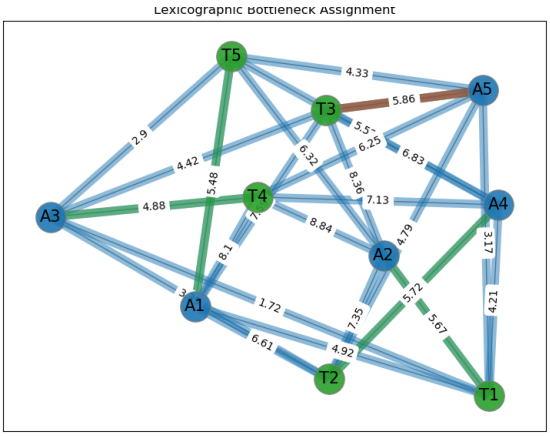
In order to obtain a comparative visualisation of the difference between the LSAP and Lexico-BAP assignments, Figure 1.4 is shown here in order to highlight the differences between its LSAP counterpart shown in Figure 1.1. The corresponding graph view highlights the first order bottleneck edge, i.e the minimal maximum distance that an agent would have to travel.

1.2.4 Collision Avoidance

It is interesting to note that collision-free trajectories can be derived from the types of assignments presented above, albeit with varying assumptions. For



(a) LexBAP agent to task allocation



(b) Graph view with displayed costs

Figure 1.4: Example of LexBAP agent-to-task pairing (5 agents and tasks)

instance, the authors in [1] have shown that straight trajectories generated by the LSAP will not intersect. Should one consider an agent as a point-mass, this does indeed imply that no collisions will occur in such an assignment. More generally however, the collision avoidance constraint can be modelled mathematically for robots occupying a finite volume in space. Indeed, it can be shown as a minimum requirement that Agents $i, i' \in \mathcal{A}$ do not collide at time t if

$$||p_i(t) - p_{i'}(t) > 2R|| \tag{1.11}$$

where R is the radius of the space that the agent's body occupies. However, the non-convex constraint that this generates is challenging to satisfy, and therefore motivates to obtain other conditions which could be implemented with more ease. To this end, it was shown in [4] that agents following a trajectory generated by the LSAP solution with squared distances as initial costs will not collide as long as the *initial* inter-agent spacing is greater than $2\sqrt{2}R$ (i.e at time $t = 0$).
It must be noted, however, that the BAP does not guarantee that the generated trajectories will not intersect, unlike the LSAP solution. Despite this, the authors in [5] propose alternative approaches to satisfying the collision avoidance condition, and stipulate that no collisions will occur in a Lexico-BAP assignment if all agents travel at constant speed (acting as point masses). The authors in [6] build on this assumption and derive local safe sets for which, should all agents remain within their safe sets, it is guaranteed to yield collision avoidance. The dynamic consistency is a conjecture that has yet to be proven and is to be empirically investigated in this project (see Section 3).

1.3 Simulation methods

1.3.1 State of the art

As a first step to implementing these optimisation methods on real robotic hardware, it is furthermore desired to obtain a deployable and modular simulation framework to assess the performance of multi-agent task optimisation with the assignment properties described above. As will be detailed throughout the report, this framework will make use of state of the art robotic development tools such as ROS2 and Gazebo. This is motivated by being consistent with recent technological advancements, and hence by using methods that are becoming increasingly more common for robotic performance assessment. The main idea behind the ROS2 middleware is that it allows for communication between different robotic entities (joint states, sensors, ...) by means of messages, topics, services and actions, which namely enables the interaction of different programs and packages through a standardised process.

1.3.2 ChoiRbot

In the early stages of this project, it was discovered that an open source ROS2 modular toolbox for cooperative robotics named ChoiRbot [9] had been developed for similar topics of interest. This toolbox can be easily cloned from the github repository [10] and readily used with some pre-defined examples.

Features such as Model Predictive Control and Formational Control are implemented, as well as a dynamic distributed task assignment module. As such, it was deemed worthwhile to use this framework as an initial architecture for customised assignment problems. This framework is built on 3 dependant layers (see Figure 1.5): Team Guidance, Planning and Robot Actuation.

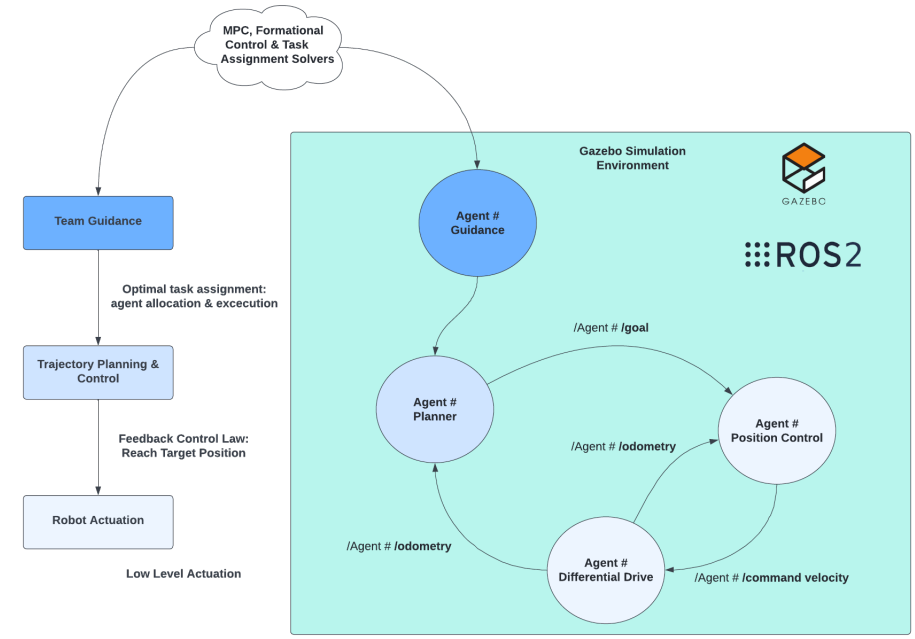


Figure 1.5: ChoiRbot[9] - 3 layer architecture

The main advantage in re-using this existing framework namely stemmed from the flexibility in design on a global navigation/planning level. Indeed, the low level robot control and actuation is already implemented using the common turtlebot3 two-wheeled differential drive robot; this saved considerable time as the research interest in the case of this project lied in assigning robots to specific locations, and therefore did not require developing an actuation law for the robots. Note that the current implementation of the controller assumes that the robots all travel at constant velocity.

1.3.3 Desired outcome

It is desired to obtain a platform where the user can, as an input, generate a specific assignment problem, and obtain not only an agent-to-task mapping but also a rendition in real time of the physical implications for agents undergoing such an assignment. From there, the user would be free to extract the properties of relevance for the topic of interest (such as collision avoidance investigation). A representation of the high level overview of the developed software can be seen in Figure 1.6. As an initial implementation, main focus will be brought in *analysing* existing assignment problem types. That is, it

shall serve solely as a centralised entity extracting properties of interest (such as dynamic consistency evaluation, collision avoidance, robustness margins, ...) as agents travel to their initially assigned distance, and shall not interfere by effectively re-assigning agents to new goals while the simulation is underway.

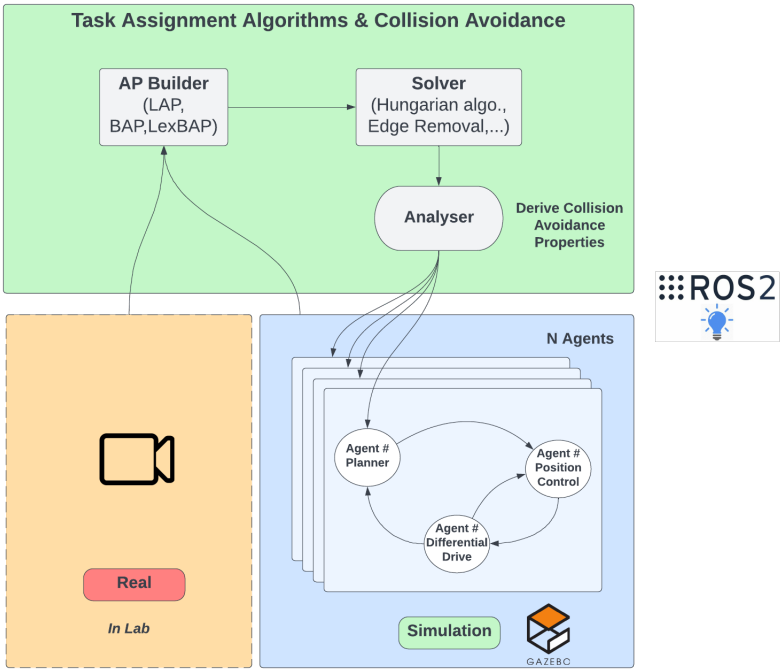


Figure 1.6: High level overview of developed software

Chapter 2

Methods

2.1 Empirical Sensitivity Analysis

In light of the material presented in the previous section, it is of interest to deploy the assignments problems into a robotic simulation environment. This will aid in not only determining the validity of the assumptions, but also enable closer monitoring of their derived properties, and how these evolve in a given situation.
As an initial case study, it was deemed worthwhile to investigate how the robustness margins evolve at each timestep from a pre-defined assignment. From the mathematical relations presented in Section 1.2, initial considerations show that they would either remain constant or increase, as long as the assignments are dynamically consistent. Furthermore, closer investigation of the safe sets and their evolution over time may help provide additional insights on the dynamic consistency of a generated assignment.

To this end, an "Empirical Sensitivity Analysis" is conducted, using the developed simulation platform to analyse the performance of specific assignments. The flexibility of the platform is exploited to simplify and decouple the analysis into two components. The first shall act as an assignment problem generation module, where agents and task positions are generated from a pre-defined set of inputs (see section 2.2). If so desired, the user can then deploy this assignment into the simulation environment, where a set of agents shall be spawned and travel to their allocated destinations.

2.2 Software Engineering Philosophy

2.2.1 ROS2 & Gazebo implementation

Figure 2.1 provides an overview on the different packages and modules used for the simulation platform. Novel contributions and/or modifications of existing modules are represented in green, while the ones in blue and purple are based on the existing modules from the main ChoiRbot package. Essentially, the package developed in this project is nested within the ChoiRbot framework, and makes use of the custom developed topics, messages, actions and services within it.

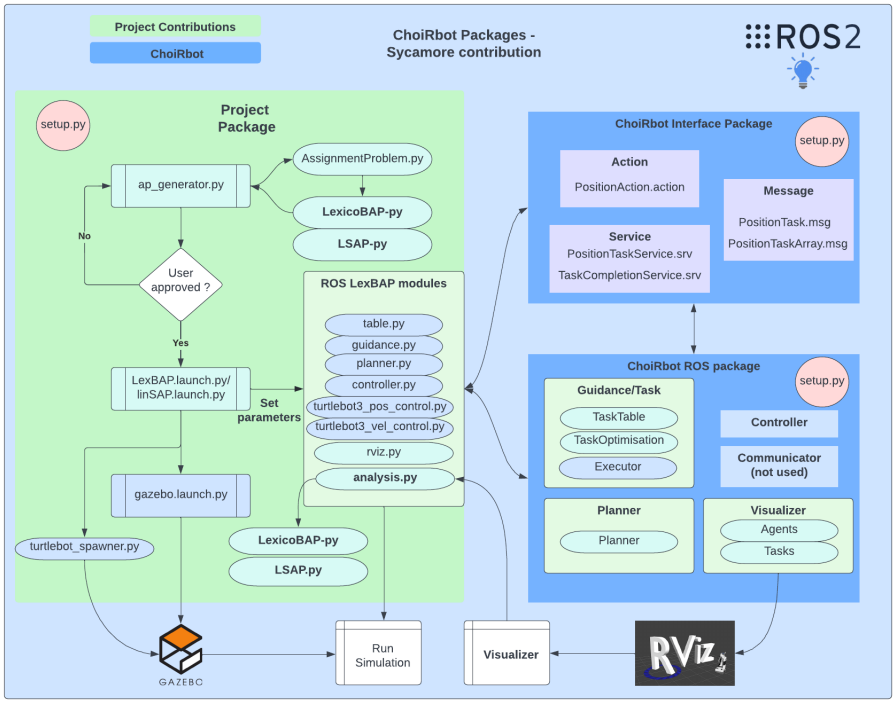


Figure 2.1: Detailed overview of python packages and modules used

From the custom developed *ap_generator.py* module, the user can generate randomised agent and task positions and assign them with either the Lexico-BAP or LSAP methods. The modules *AssignmentProblem.py*, *LexicoBAP.py* and *LSAP.py* have been implemented as Python classes which also include the different solvers (Edge-removal algorithm for Lexico-BAP, and Hungarian method for LSAP). Should the user like to pursue a simulation of a given assignment, the corresponding ROS2 launch file can be exectued, which will trigger the generation of the necessary nodes and topics to proceed. Simultaneously, the turtlebot3 models are spawned in the Gazebo simulation environment, and await the action commands to proceed from the guidance, planning and actuation layers.
As an additional layer of monitoring, the visual debugger RVIZ has also been implemented in a customised fashion: the agents and tasks are represented as markers that are subscribers to the corresponding topics. Whenever a new message is published through each agent's odometry and goal topics, the message is received and displayed to provide the user with a status of the overall simulation. Figure 2.4 represents the core of the developed platform from a user perspective, and aims to provide a detailed understanding on the underlying architecture.

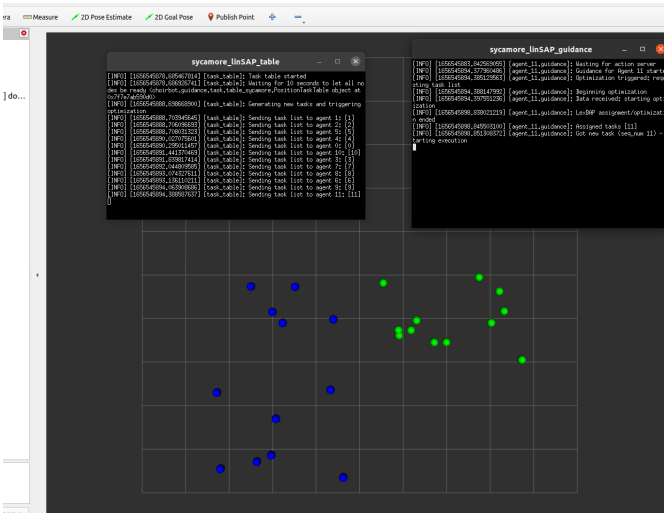


Figure 2.2: Example of RVIZ visualiser

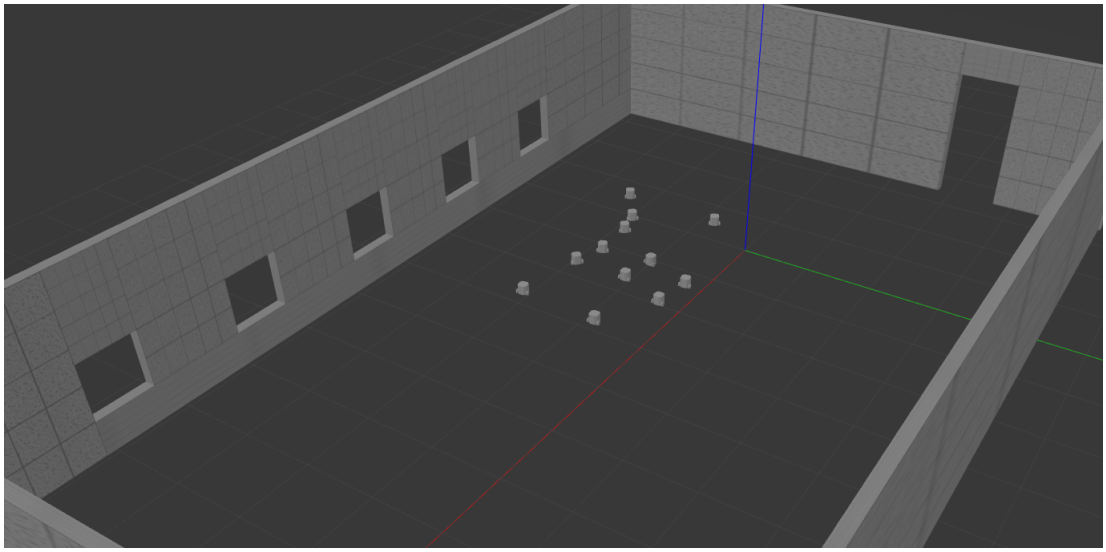


Figure 2.3: Example of obtained simulation in Gazebo

2.2.2 Simulation Setup

To proceed with an experiment, the user is required to input a minimum of two parameters, namely the number of agents and tasks (agent and task dimensions, respectively). This calls the *AssignmentProblem* to generate a problem, and the agent-to-task pairing is obtained both with the *LexicoBAP* and *LSAP*. From there, both assignments can be stored in the current working directory as well as in the shared ROS2 package directory, should the assignment be satisfactory in a user-sense. This is in direct conformity with the

current ROS2 functionalities and package directories, and therefore allows to run the launch files without having to manually move the files.
In order to proceed with the simulation, it is required to provide an ID for each agent and task pairing - this is done by ordering the assignments in a sequential bottleneck manner (for Lexico-BAP) or randomly for the LSAP. This assignment is directly fed into the Task Table node, which then sets the guidance trigger for each agent to proceed with the allocated task assignment.

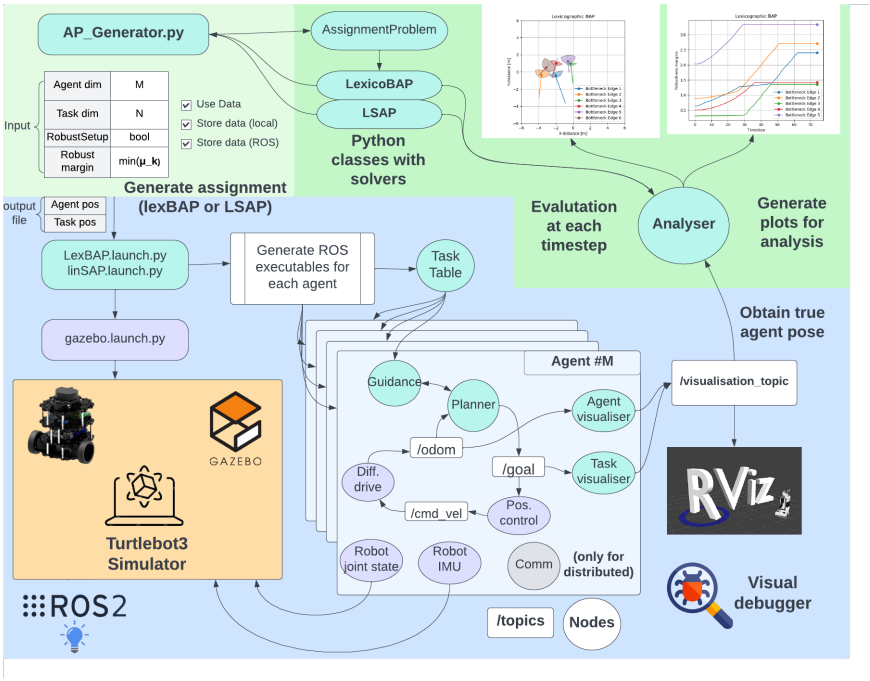


Figure 2.4: Python architecture and input-output flowchart using ROS2 and Gazebo

The *Analyser* node receives updated agent positions from the topic *visualisation_topic*, which already processeses the odometry data of all agents. The node waits until all agent poses are received, and then proceeds to recompute an assignment. It consequently enables the investigation of whether the assignment properties are preserved by outputting various plots that are dependent on the problem at hand. In the case of the LSAP, agent positions can be shown at all elasped timesteps; the user is provided with stored trajectory data at each timestep of all agents positions as well their respective safety sets and robustness margins. Observations of these plots as well as the simulation within Gazebo provides insights to investigate whether collisions occur, and whether an alternative re-assignment has been computed at a particular timestep. Recall that, as previously mentioned, this remains solely a centralised analysis of what occurs during the en-route trajectory of an agent towards its assigned task - the current implementation does not allow for the agents to actually be re-assigned towards their goals.

Chapter 3

Results

3.1 Preliminaries

As a direct derivation from the afore-mentioned theoretical properties, the following conjectures are stated to form the basis for the remainder of the study. It mainly states as a sufficient condition that dynamic consistency is preserved as long as the agents remain within their safe sets. This is presumed to be the case for all cases where the initial heading is aligned to that of its destination, adding some relaxation regarding the control strategy used for the robotic actuation, as it is assumed that it will travel in a straight line (little to no deviation from intended path).
Conjecture 1: If all assigned agents $i_k* \in \mathcal{A}$ stay within their safe sets $\mathcal{L}_i(t)$, then the assignment is dynamically consistent for all $t \in [0, T]$. That is, the optimal assignment remains the same as the initially computed one $\forall t \in [0, T]$.
Conjecture 2: Given an initial heading aligned with their respective task, all assigned agents $i_k* \in \mathcal{A}$ are within their safe sets $\mathcal{L}_i(t)$ for all $\forall t \in [0, T]$, and thus dynamic consistency is preserved.

These conjectures are stated as closer inspection of the control strategy used for the Turtlebot3 Burger robot showed that, despite it being a two-wheel differential drive robot, the current implementation does not enable on-position rotation. As mentioned in Section 1.3.2, the robots travel at constant velocity, and actuation is made in a "twist and move" fashion, which implies that a certain deviation in trajectory can arise should the initial heading not be aligned with the agent's assigned goal. The quantification of this curvature is known as the instantaneous Center of Curvature - ICC, and is represented in Figure 3.1 below. Additionally, considering the turtlebot width dimensions are approximately 180 mm , this led to setting the minimum safety distance of $s = 0.1m$ (i.e slightly higher than the radius of the robot).

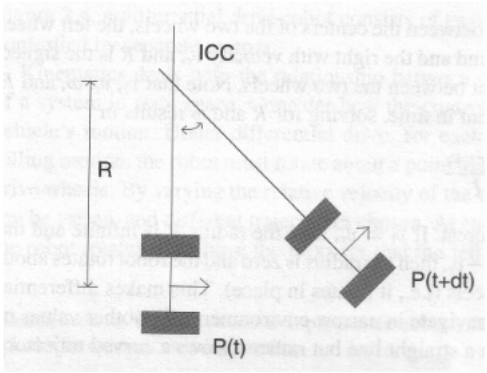


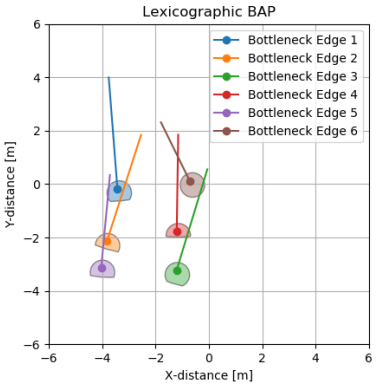
Figure 3.1: Forward kinematics for differential robot [11]

3.2 Analysis: case-study

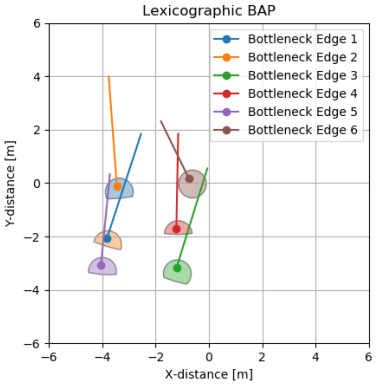
3.2.1 6 agents

Lexico-BAP with $\min \mu = 0.4$

In this experiment, an assignment with a minimum robustness margin of $\mu = 0.4$ is generated. The assignment preserves dynamic consistency as the agents-to-task matching remains the same throughout the simulation. Interestingly, it can be seen that the initial first and second order bottleneck edges switch at the third timestep. Closer inspection of the costs of each assignment sheds some light into this - it can be seen that the second order bottleneck is extremely close to the first order one. Indeed, despite an initial heading of the agents aligned with the goals, the slightest deviation of the agents from the intended path consequently changes the order.

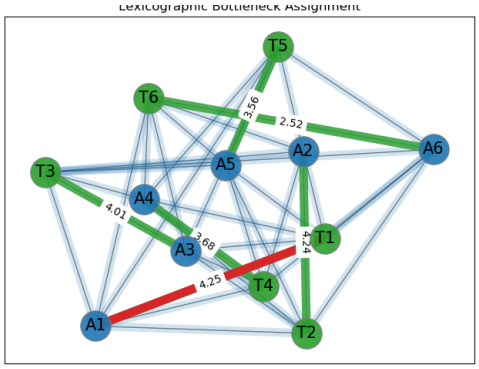


(a) Initial assignment

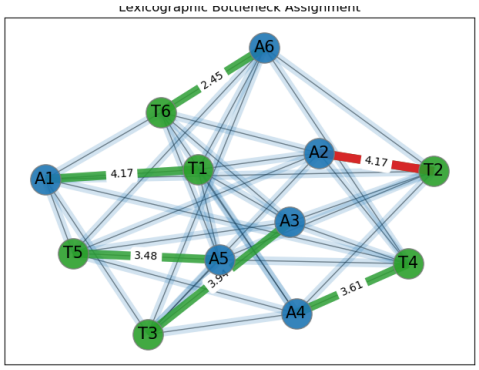


(b) Conserved assignment with varying order (at following timestep)

Figure 3.2: Same assignment with varying order in safe sets



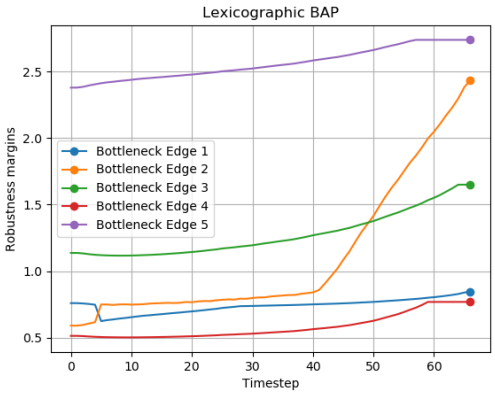
(a) Initial assignment



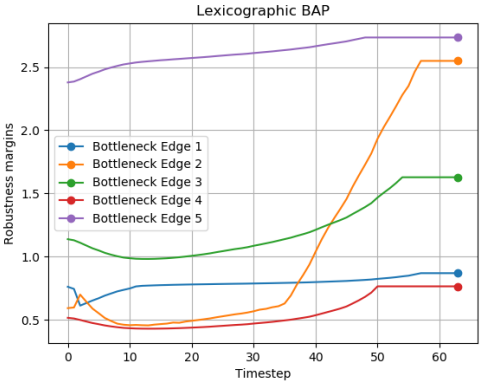
(b) Conserved assignment with varying order

Figure 3.3: Same assignment with varying order in safe sets (graph view)

Additionally, investigation of the robustness margin plots in function of the timestep highlights this switch at the very early phase of the assignment. The same conclusion can be drawn when inspecting the plot with an initial heading opposed to that of the task. Furthermore, as intuition would have it, the robustness margins actually decrease until the robot is once again aligned with the final task (see fig 3.4 (b)).



(a) Initial heading aligned with task

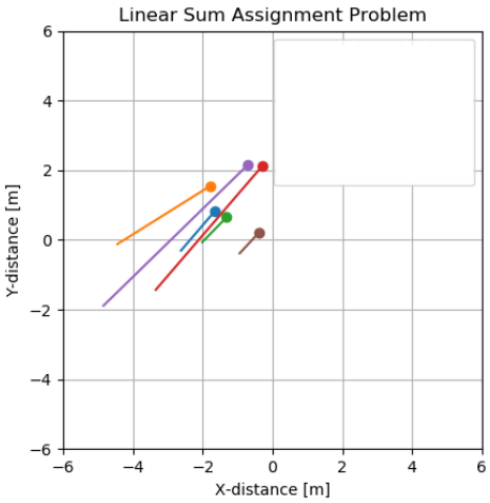


(b) Initial heading opposite to task

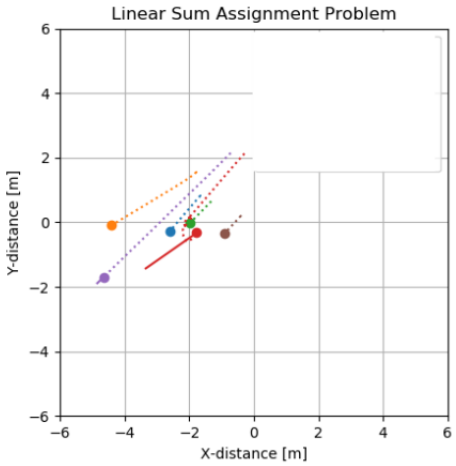
Figure 3.4: Robustness margins comparison with initial heading variation : aligned (top) and random (bottom)

Performance comparison Lexico-BAP & LSAP

After generating an assignment with a $\mu = 0.15 \simeq s$, the performance of the Lexico-BAP was compared to that of the LSAP's. The latter's optimal assignment led to a collision occurring for the red agent, and consequently never reaching its end goal, despite having an initial heading aligned. This is represented in figure 3.5 - as can be seen, the turtlebot3 dimensions proved to be too large in order for the red agent to pass through. (Note that the bottleneck edges as a legend is a typo and was unable to be removed...)



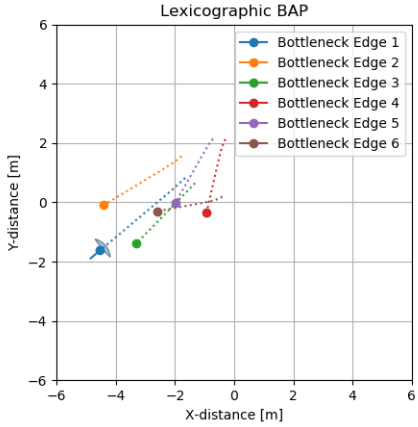
(a) Initial LSAP solution



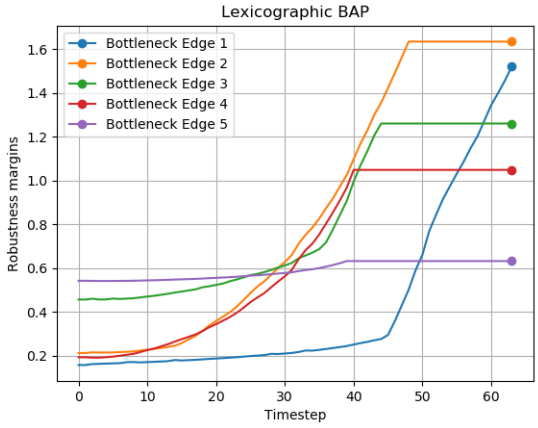
(b) Observed collision of red agent

Figure 3.5: LSAP view of before and after collision

In the case of the Lexico-BAP, however, the agents remained within their safe sets throughout the robots' navigation. As previously mentioned, it is presumed that this provides dynamic consistency and ensures that the agents do not collide with one another. This can be confirmed by visual inspection through the gazebo software.



(a) Lexico-BAP final timestep with trajectory data

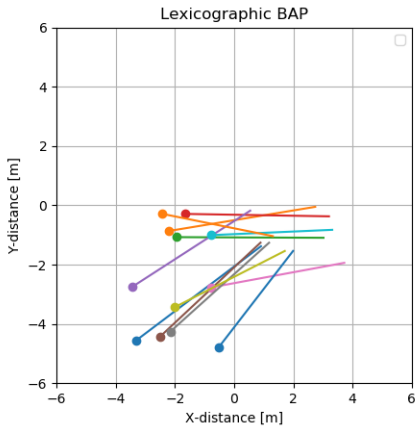


(b) Associated robustness margin

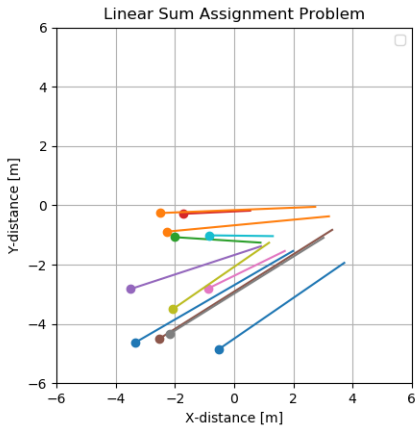
Figure 3.6: Robustness margins comparison with initial heading

3.2.2 12 Agents

As an additional experiment, 12 agents were generated in the same confined space as the previous examples. Furthermore, no minimal robustness margins were set : computation of the assignment showed that $\mu = 0.07$, which is smaller than the safety distance. The initial assignment for both assignment problems is shown in figure 3.7. This being said, no collisions occurred for the Lexico-BAP assignment. However, the LSAP solution once again proved to be lacking the same properties as the Lexico-BAP regarding collision avoidance; in fact the robots collided multiple times while en-route. An example of a collision is shown in 3.8.



(a) Initial assignment with Lexico-BAP



(b) Initial assignment with LSAP

Figure 3.7: Lexico-BAP and LSAP comparison with 12 agents (example)

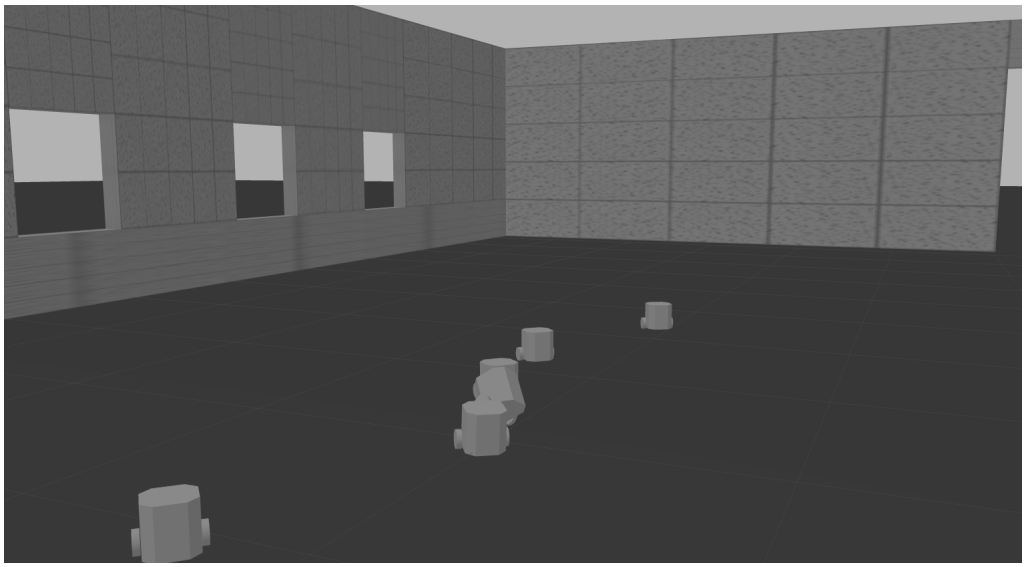


Figure 3.8: Rendering of observed collision in Gazebo

3.3 Interpretation and discussion

As was briefly shown in the above simulations, the generation methodology is relatively straightforward and can provide many insights on the performance of one task allocation methods with regards to another.
Interestingly, after many simulations (not all shown here) it is observed that none of the Lexico-BAP solutions presented collisions. The same cannot be said for the LSAP case as shown above. The Lexico-BAP, however, did not always present dynamic consistency; it could however always be ascertained that the agents were not remaining within their safe sets.
Additionally, the inspection of the robustness margins never decreasing given a truly aligned initial heading help confirm the intuitions and conjectures presented earlier. As was shown, the assignments do remain rather sensitive to physical properties of the robot, such as the dimensions and control. The latter can for instance influence the turning curvature, which in turn can deviate the agents from its intended goal. This has the consequence of reducing the robustness margins until a straight path is obtained again. Depending on the minimum margin, this can however imply the lack of dynamic consistency for a Lexico-BAP assignment.

Chapter 4

Conclusions

Section 1 introduced the supporting theoretical material to conduct the case study presented throughout the report. Different types of assignments, namely the Linear Sum Assignment Problem (LSAP), the Bottleneck Assignment Problem (BAP) and its sequential subclass, the Lexicographic BAP (Lexico-BAP) were objectively compared in terms of what properties can be derived from them. The most noteworthy properties rely on collision avoidance as well as the dynamic consistency of an assignment, i.e whether an assignment remains the optimal assignment during the time that the agents travel to their allocated destinations.

Section 2 brought further insights into the strategies applied to obtain a comparative assessment of these assignment problems. This was done by building upon an existing toolbox, named ChoiRbot, using ROS2 and Gazebo to simulate various robotic control and navigation methods. Furthermore, assumptions were introduced in order to make a qualification study for an empirical sensitivity analysis.

The latter served as the basis for the obtained results presented in section 3. As was shown, a modular and deployable task-assignment simulation platform enabled the assessment and analysis of randomly generated agent and task locations, and whether phenomena such as collision avoidance or dynamic consistency was given.

4.1 Next Steps

For future, it would be of interest to include actual jetbot models as opposed to the currently used Turtlebot3 Burger robots.

In order to make the platform more coherent with the current architecture, it would be of relevance to further extend it and add actual agent-relocalisation should a more optimal assignment be computed while on route. On a similar note, it would be of interest to investigate the case where the minimal robustness margin is not set, and in the case that a margin is smaller than the safety distance (i.e $\mu < s$), keep these agents static and move the other ones. Furthermore, it is worth noting that only two assignment problems were effectively integrated into the simulation platform; additional task-allocation

methods would undoubtedly make for more relevant comparisons in the future.

As an additional remark, it would be wise to optimise the coding of the python modules. Indeed, much of the code used for the Lexico-BAP solver was adapted from a previously established Matlab version, and as such could perhaps be improved to better comply with the python IDE.

4.2 Python code

All of the python code can be found in the corresponding github repository [12] following from this project. A detailed README is present within the *sycamore* package.

Bibliography

[1] I. Shames, B. Fidan, and B. D. Anderson, "Close target reconnaissance with guaranteed collision avoidance", in *Int. J. Robust Nonlinear Control.*, vol. 21, no. 16, pp. 1823–1840, 2011.

[2] Linear Sum Assignment Problem, `http://www.assignmentproblems.com/doc/LSAPIntroduction.pdf` (accessed: 27.06.2022)

[3] Hungarian algorithm, python `https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.optimize.linear_sum_ass` 27.06.2022)

[4] M. Turpin, N. Michael, and V. Kumar, "Capt: Concurrent assignment and planning of trajectories for multiple robots", in *Int. J. Robot. Res.*, vol. 33, no. 1, pp. 98–112, 2014.

[5] P. MacAlpine, E. Price, and P. Stone, "Scram: Scalable collision-avoiding role assignment with minimal-makespan for formational positioning", in *Proc. AAAI Conf. Artif. Intell.*, 2015, pp. 2096–2102

[6] Tony A. Wood, Mitchell Khoo, Elad Michael, Chris Manzie, and Iman Shames "Collision Avoidance Based on Robust Lexicographic Task Assignment", in *IEEE Robotics and automation letters,*, VoL. 5, No. 4, OCTOBER 2020

[7] Mitchell Khoo, Tony A. Wood, Chris Manzie, Iman Shames. "Distributed Algorithm for Solving the Bottleneck Assignment Problem", in *IEEE 58th Conference on Decision and Control (CDC),*, Palais des Congrès et des Expositions Nice Acropolis, 2019

[8] Mathias Bürger, Giuseppe Notarstefano, Francesco Bullo, Frank Allgöwer "A distributed simplex algorithm for degenerate linear programs and multi-agent assignments", in *Automatica*, p.2298–2304 (2012)

[9] Testa, Andrea and Camisa, Andrea and Notarstefano, Giuseppe "ChoiRbot: A ROS 2 toolbox for cooperative robotics", in *IEEE Robotics and Automation Letters*, VoL. 6, No. 2, p.2714-2720, 2021

[10] OPT4SMART/ChoiRbot Github repository `https://github.com/OPT4SMART/ChoiRbot`

[11] Differential Drive Robots Kinematics
https://www.cs.columbia.edu/ allen/F17/NOTES/icckinematics.pdf
(accessed: 27.06.2022)

[12] Semester project code github repository
https://github.com/dhollosi/Sycamore$_{L}exBAP$(*accessed* : 05.07.2022)

[13] Networkx Python package
https://networkx.org/ (accessed: 05.07.2022)