Task Re-Allocation Methodologies for Teams of Autonomous Agents in Dynamic Environments

by

Patricia Kristine Sheridan

A thesis submitted in conformity with the requirements for the Degree of Master of Applied Science
Mechanical and Industrial Engineering
University of Toronto

© Copyright by Patricia Kristine Sheridan 2011
Task Re-Allocation Methodologies for Teams of Autonomous Agents in Dynamic Environments

Patricia Kristine Sheridan

Master of Applied Science

Department of Mechanical and Industrial Engineering
University of Toronto

2011

Abstract

Two on-line task re-allocation methodologies capable of re-allocating agents to tasks on-line for minimum task completion time in dynamic environments are presented herein. The first methodology, the Dynamic Nearest Neighbour (DNN) Policy, is proposed for the operation of a fleet of vehicles in a city-like application of the dial-a-ride problem. The second methodology, the Dynamic Re-Pairing Methodology (DRPM) is proposed for the interception of a group of mobile targets by a dynamic team of robotic pursuers, where the targets are assumed to be highly maneuverable with a priori unknown, but real-time trackable, motion trajectories.

Extensive simulations and experiments have verified the DNN policy to be tangibly superior to the first-come-first-served and nearest neighbour policies in minimizing customer mean system time, and the DRPM to be tangibly efficient in the optimal dynamic re-pairing of multiple mobile pursuers to multiple mobile targets for minimum total interception time.
Acknowledgments

First and foremost, I would like to thank my thesis supervisor Professor B. Benhabib for all his time, patience, advice and encouragement throughout my undergraduate and graduate studies. The guidance, and many comments and suggestions led me to truly understand the topic at hand, and the importance of hard work. Many thanks are also owed to Professor G. Nejat, for her comments and suggestions on autonomous agent systems that assisted in the development of the Dynamic Repairing Methodology, and to Professor B. Balcıoğlu, who aided me in understanding the Operations Research side of task-allocation quickly, and for all his comments and suggestions on how to approach the work on the dial-a-ride problem.

I am very grateful to Faraz Kunwar, with whom I worked on the Advanced Predictive Guidance Law in my undergraduate summer breaks. His previous work in the area of robotic interception and encouragement of my love of robotics research encouraged me to pursue graduate studies. I wish you success in your academic career. I am also very grateful to Pawel Kosicki for his previous contributions to this work on the development of the Optimization Search Engine. Thank you for all your assistance. I would like to thank all my friends and colleagues in the Computer Integrated Manufacturing Laboratory at the University of Toronto for their conversations, friendship, and support during my M.A.Sc. research: Matthew Mackay, Ashish Macwan, David Schacter, Chris Wang, Chen Liu, Hay Azulay, and Gelareh Namdar.

I also had the pleasure to work with several students who worked with me on this thesis. A special thanks is owed to Chen Liu for her time and hard work spent upgrading and calibrating the experimental setup. Thank you for persevering through all the hardware challenges. A special thanks is also owed to Erich Gluck for his work developing the simulation environment for the DNN Policy, his work ensured that different simulation test cases were easy to run. A special thanks is also owed to Qi Guan who worked with me on interpreting simulation results, determining the optimal policy, and looking at policy improvements during his summer work and undergraduate thesis.

I am eternally grateful for the support, prayers, and encouragement of my family and friends, both here and afar, without whom life would be meaningless. A great many thanks are owed to my Mum. Without her tireless devotion to my happiness and education I would not be where I am today. Many thanks are also owed to Vainatey for his encouragement and understanding during many long work hours. Both of your encouragement was always a source of strength.

Finally, I gratefully acknowledge the financial support of the Natural Science and Engineering Research Council, and the University of Toronto Graduate Fellowship.
# Table of Contents

Abstract ................................................................................................................................. ii
Acknowledgments ................................................................................................................ iii
Table of Contents ................................................................................................................ iv
List of Tables ......................................................................................................................... viii
List of Figures ....................................................................................................................... x
Nomenclature and Acronyms ................................................................................................ xii

1. Introduction ..................................................................................................................... 1
   1.1. Motivation .................................................................................................................... 1
   1.2. Literature Review ......................................................................................................... 4
       1.2.1. Task-Allocation for the Dial–a–Ride Problems ..................................................... 4
       1.2.2. Task-Allocation for Multi-Target Interception Problems .................................... 7
       1.2.3. Pursuer Motion-Planning ....................................................................................... 9
   1.3. Research Objectives ..................................................................................................... 10
   1.4. Thesis Organization ..................................................................................................... 11

2. Dynamic Nearest Neighbour Policy ................................................................................. 13
   2.1. Problem Definition ....................................................................................................... 14
   2.2. Dynamic Nearest Neighbour Policy (DNN) ................................................................. 16
       2.2.1. DNN Algorithm ..................................................................................................... 18
       2.2.2. Anticipatory Behaviour ......................................................................................... 20
           2.2.2.1. Anticipatory DNN Policy ................................................................................ 21
       2.2.3. Customer Wait-Time Limits .................................................................................. 23
   2.3. Simulations .................................................................................................................. 25
       2.3.1. Methodology .......................................................................................................... 25
           Simulation Cases .......................................................................................................... 26
       2.3.2. Simulation Environment ......................................................................................... 27
       2.3.3. Case 1: Analysis of DNN, NN, and FCFS in Base-case scenario .......................... 27
           2.3.3.1. Base Case Analysis of the FCFS and NN Policies ....................................... 28
           2.3.3.2. Base Case Analysis of the NN and DNN Policies ....................................... 31
       2.3.4. Case 2: Analysis of DNN and NN in City-like Environments ............................... 33
           2.3.4.1. City Scenario ................................................................................................. 33
2.3.4.2. Performance of DNN and NN in the City Scenario ......................... 34
2.3.5. Case 3: Analysis of DNN and NN in City-like Environments with
Anticipatory Behaviour ............................................................................. 36
  2.3.5.1. Placement of Vehicle Stations for Anticipatory Behaviour .............. 37
  2.3.5.2. Analysis of the NN and DNN policies under the City Scenario with
Anticipatory Behaviour ............................................................................. 40
2.3.6. Case 4: Analysis of DNN and NN in the City Scenario with Anticipatory
Behaviour and Capping of Customer Wait Times .................................. 43
2.4. Summary ............................................................................................ 46
3. Dynamic Re-Pairing Methodology ........................................................... 47
  3.1. Problem Definition ............................................................................ 48
  3.2. Dynamic Re-pairing Methodology ...................................................... 51
    3.2.1. Data-Management Module (Blackboard) .................................. 53
    3.2.2. Information-Acquisition Module ............................................... 55
    3.2.3. Task-Allocation Module .......................................................... 55
    3.2.4. Navigation Module .................................................................... 56
  3.3. Dynamic Re-Pairing Methodology Implementation ............................. 57
    3.3.1. Pursuer/Target Pairing (Task-Allocation Module) ......................... 57
      3.3.1.1. Interception Time Metric .................................................... 57
      3.3.1.2. Optimization Search Engine ............................................. 59
      3.3.1.3. Coping with Inequalities in the Numbers of Pursuers and Targets .... 60
    3.3.2. Target Tracking (Information-Acquisition Module) ....................... 61
    3.3.3. Pursuer Motion Trajectory-Planning (Navigation Module) ............. 61
      3.3.3.1. Guidance Method .................................................................... 61
        3.3.3.1.1. Proportional Navigation Guidance Law ......................... 62
        3.3.3.1.2. Advanced Predictive Guidance Law .............................. 62
      3.3.3.2. Obstacle Avoidance .......................................................... 63
      3.3.3.3. Motion Controller .............................................................. 63
  3.4. Simulations ....................................................................................... 64
    3.4.1. Methodology ............................................................................... 64
    3.4.2. Effectiveness of the On-line Re-pairing Methodology .................... 64
    3.4.3. Benefits of Using a Rolling Horizon ........................................... 65
3.4.4. Re-pairing Examples of Proposed Methodology ........................................ 66
  3.4.4.1. Case 1: Fixed Numbers of Pursuers and Targets throughout Pursuit .... 67
  3.4.4.2. Case 2: Dynamic Numbers of Targets with Fixed Number of Pursuers .................................................................................................................. 68
  3.4.4.3. Case 3: Dynamic Numbers of Targets with Pursuers Consumed on
             Interception .......................................................................................... 70
  3.4.4.4. Case 4: Dynamic Numbers of Pursuers and Targets in a
             Disengagement Pursuit......................................................................... 71

3.5. Summary .................................................................................................. 76

4. Dynamic Re-Pairing Methodology – Experiments ........................................ 77
  4.1. Experimental Set-up ............................................................................... 77
    4.1.1. Vision System .................................................................................. 78
      4.1.1.1. Hardware .................................................................................. 78
      4.1.1.2. Software .................................................................................. 79
    4.1.2. Pursuer Robots ............................................................................... 82
    4.1.3. Communication System .................................................................. 83

4.2. Results and Discussion ........................................................................... 84
  4.2.1. Experiment 1: Constant Numbers of Pursuers and Targets ............... 85
  4.2.2. Experiment 2: Dynamic Numbers of Targets ................................... 87
  4.2.3. Example 3: Dynamic Numbers of Pursuers ................................... 89

4.3. Summary .................................................................................................. 91

5. Conclusions and Recommendations ............................................................. 92
  5.1. Summary and Contributions .................................................................. 93
    5.1.1. Dynamic Nearest Neighbour Policy .............................................. 93
    5.1.2. Dynamic Re-Pairing Methodology ................................................ 93

5.2. Recommendations for Future Work ......................................................... 94
  5.2.1. Dynamic Nearest Neighbour Policy ................................................ 94
    5.2.1.1. Realistic City-Traffic Conditions ............................................ 94
    5.2.1.2. Dynamically Varying Numbers of Vehicles ............................. 94
  5.2.2. Dynamic Re-Pairing Methodology ................................................... 95
    5.2.2.1. Interception and Target Types Addressed .................................. 95
    5.2.2.2. Dynamic and Realistic Environment ....................................... 95
5.2.2.3. Hardware and Software Issues .......................................................... 95
5.2.2.4. Implementation .............................................................................. 96

5.3. Final Concluding Statement .................................................................. 96

References ...................................................................................................... 98

Appendix A: Complete DNN Policy Algorithm ......................................... 103
Appendix B: Blackboard Example ................................................................. 105
Appendix C: Kinematic Model of the Mobile Robot .................................. 107
List of Tables

Table 2.1. Simulation Cases Studied .................................................................................................................. 26

Table 2.2. Percent Improvement of the NN policy over the FCFS policy ............................................................ 29

Table 2.3. Percent Improvement of the DNN policy over the NN policy ............................................................ 31

Table 2.4. Hourly Changes in Customer Arrival Rates and Demand Distributions in the City Scenario .......................................................... 34

Table 2.5. Mean System Time increase in City scenario as compared to the Base-case scenario. .......................... 35

Table 2.6. Percent Improvement of the DNN policy over the NN policy in the City Scenario ........................... 36

Table 2.7. Vehicle Station Distribution ........................................................................................................... 37

Table 2.8. Mean System Time decrease in the City Scenario with Anticipatory Behaviour .................................. 41

Table 2.9. Percent Improvement of DNN over NN in the City Scenario with Anticipatory Behaviour .................. 42

Table 2.10. Average Longest Wait Times with respect to the Mean Wait Time with DNN in Case 3 ............................................................................. 43

Table 2.11. Decrease in the Longest Wait Times with Capping as compared to no Capping in Case 3 .............................................................................. 44

Table 2.12. Greatest Increases in Mean System Time as a result of using Wait-Time Limits .............................. 45

Table 2.13. Percent Improvement of DNN over NN in the City Scenario with Anticipatory Behaviour and Capping ................................................................................. 46

Table 3.1. Data Exchange Rights on the Data-Management Module (Blackboard) ............................................ 53

Table 3.2. Pursuer State Data Structure [P] ......................................................................................................... 53

Table 3.3. Target State Data Structure [T] ......................................................................................................... 54

Table 3.4. Pairings Matrix Data Structure [X] .................................................................................................... 54

Table 3.5. Mean Interception Times (s) for 200 Trials ......................................................................................... 65

Table 3.6. Summary of Simulation Cases ......................................................................................................... 67

Table 3.7. Target-Trajectory Information for Fig. 3.4 ....................................................................................... 68

Table 3.8. Target-Trajectory information for Fig. 3.5 ....................................................................................... 70
Table 3.9. Target-Trajectory Information for Fig. 3.6. ................................................................. 71
Table 3.10. Target-Trajectory Information for Fig. 3.7. ................................................................. 72
Table 4.1. Summary of Experiment Cases. ......................................................................................... 84
Table 4.2. Target-Trajectory Information for Experiment 1. ............................................................ 86
Table 4.3. Interception Times for Experiment 1. ................................................................................. 86
Table 4.4. Target-Trajectory Information for Experiment 2. ............................................................ 87
Table 4.5. Interception Times for Experiment 2. ................................................................................. 89
Table 4.6. Target-Trajectory Information for Experiment 3. ............................................................ 91
Table 4.7. Interception Times for Experiment 3. ................................................................................. 91
List of Figures

Fig. 2.1. Mean System Time Percent Improvement of NN over FCFS under the base-case scenario. .................................................................................................................. 28

Fig. 2.2. (a) Mean System Time comparisons for $N = 1$. .......................................................... 29

Fig. 2.3. Mean travel time between two consecutively served customers under the NN policy. 30

Fig. 2.4. Percent Improvement of DNN over NN in Base-case scenario. .................................... 32

Fig. 2.5. Mean System Time Increase from City over Base Case for the NN Policy. ............. 35

Fig. 2.6. Mean System Time Increase from City over Base Case for the DNN Policy.............. 35

Fig. 2.7. Percent Improvement of the DNN policy over the NN policy in the City Scenario..... 36

Fig. 2.8. Locations of vehicle stations for standard hours when $N = 10$. .............................. 39

Fig. 2.9. Locations of vehicle stations during morning hours when N = 10. .......................... 39

Fig. 2.10. Mean System Time Decrease with anticipatory behaviour in NN. ....................... 41

Fig. 2.11. Mean System Time Decrease with anticipatory behaviour in DNN....................... 41

Fig. 2.12. Percent Improvement of DNN over NN in the City Scenario with Anticipatory Behaviour.................................................................................................................. 42

Fig. 2.13. Percent Improvement of DNN over NN in the City Scenario with Anticipatory Behaviour and Capping.................................................................................. 45

Fig. 3.1. Tree structure of $F$. ........................................................................................................ 50

Fig. 3.2. Flowchart of Proposed Methodology ............................................................................. 52

Fig. 3.3. Pursuer Trajectories for (a) One-Stage Horizon and (b) Two-Stage Horizon Simulations. .................................................................................................................. 66

Fig. 3.4. Simulation Trajectories for a fixed number of pursuers and targets. ..................... 68

Fig. 3.5. Simulation Trajectories for a dynamic number of targets disengaged on interception. 69

Fig. 3.6. Simulation Trajectories for a dynamic number of targets with pursuers consumed on interception. ........................................................................................................... 71

Fig. 3.7. (a): Simulation Trajectories up to and including Time A. ............................................ 72

Fig. 4.1. Model of the Setup’s Layout. ....................................................................................... 78

Fig. 4.2. Physical Setup with Pursuer Robots.............................................................................. 78

Fig. 4.3. Colour Marker Search – Modified from [59]. .............................................................. 82

Fig. 4.4. Close-up image of 3 Pursuer Miabots on the Workspace. ......................................... 83

Fig. 4.5. Pursuer Trajectories for the Experiment-1 Simulation.............................................. 85
Fig. 4.6. Pursuer Trajectories for Experiment 1. ................................................................. 86
Fig. 4.7. Pursuer Trajectories for Experiment-2 Simulation. ............................................... 88
Fig. 4.8. Pursuer Trajectories for Experiment 2. ................................................................. 88
Fig. 4.9. Pursuer Trajectories for Experiment-3 Simulation. .............................................. 90
Fig. 4.10. Pursuer Trajectories for Experiment 3. ............................................................... 90
Nomenclature and Acronyms

Latin Letters

\(a_p\)  
Acceleration Command for Advanced Predictive Guidance

\(a_{PN}\)  
Acceleration Command for Proportional Navigation

\(a_T\)  
Target acceleration

\(\dot{a}_T\)  
Target Jerk

\(\mathcal{A}\)  
Euclidean Square Region environment of the dial-a-ride problem

\(A\)  
Area of \(\mathcal{A}\)

\(B\)  
Blue Colour

\(C\)  
Alternate Representation of \(X\) for Rolling Horizons

\(Cb\)  
Chrominance-blue

\(Cr\)  
Chrominance-red

\(D\)  
Matrix of all Vehicle-Customer Distances

\(D\)  
Distance Travelled to Reach a Customer

\(D\)  
Distance between a pixel and a pre-defined colour

\(D_h\)  
Percentage of Customer Demands that follow rush-hour trends

\(D_u\)  
Percentage of Customer Demands that are uniform across \(\mathcal{A}\)

\(E[D]\)  
Expected Travel Distance to Reach a Customer

\(E[S]\)  
Expected Service Time in Base-case Scenario

\(E[S^2]\)  
Second Moment of Service Time

\(E[S_c]\)  
Expected Service Time in City-like Conditions

\(E[T^*]\)  
Expected Mean System Time of the Optimal Policy

\(E[T_{DNN}]\)  
Expected Mean System Time of the DNN Policy

\(E[T_{FCFS}]\)  
Expected Mean System Time of the FCFS Policy

\(E[T_{NN}]\)  
Expected Mean System Time of the NN Policy

\(F\)  
Matrix of all Pursuer-Target Interception Times

\(G (X)\)  
Interception Times of the Pairings selected in \(X\)

\(G\)  
Green Colour

\(h\)  
Total Interception Times for Pursuers to Intercept all Assigned Targets
$H_e$  Percentage of Customer Drop-offs in the High-demand Area
$H_s$  Percentage of Customer Pick-ups in the High-demand Area
$K$  Number of Unassigned and Assigned Customers
$l$  Dimension of the smallest pattern marker in pixels
$m$  Number of Available Pursuers
$n$  Number of Not-intercepted Targets
$N$  number of vehicles in a fleet
$N_{H}$  Number of Horizontal Lines of Vehicle Stations
$N_{V}$  Number of Vertical Lines of Vehicle Stations
$N$  Navigation Ratio
$p$  Position Vector
$p_e$  Closing Distance between a Pursuer and Target
$P$  Pursuer States Data Structure
$P_h$  Percentage of $A$ that represents the High-demand Area
$P_r$  Percentage of Vehicles Routed to Stations in the High-demand Area
$r$  Percentage of Time a Vehicle is Busy
$R$  Red Colour
$R_{TP}$  Euclidean Distance between a Pursuer-Target Pair
$s$  Depth of the Rolling Horizon
$S$  Service Time of a Customer
$t_f$  Duration of interception
$t_{go}$  Time-to-go until interception
$t_{ij}$  Interception Time of Target $j$, by Pursuer $i$
$T$  Target States Data Structure
$T$  System Time of a Customer
$v$  Velocity Vector
$v_e$  Closing Velocity
$v$  Velocity of fleet vehicles
$v_R$  Magnitude of the Closing Velocity
$W$  Wait Time of a Customer
$\bar{W}_{N,\lambda}$  Mean Wait Time Estimate for $N$ and $\lambda$
X Matrix of Pursuer-Target Pairings
Y Matrix of Vehicle-Customer Pairings
Z Objective Function

**Greek Letters**

<table>
<thead>
<tr>
<th>Greek Letter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>Significance Level</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>System Time Estimate Constant</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Customer Arrival Rate</td>
</tr>
<tr>
<td>$\dot{\lambda}$</td>
<td>Rate of change of the line-of-sight</td>
</tr>
<tr>
<td>$\lambda_C$</td>
<td>Mean Customer Arrival Rate in City-like Conditions</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Percentage of Time a Vehicle is Assigned or Busy</td>
</tr>
<tr>
<td>$\delta_{N,\lambda}$</td>
<td>Standard Deviation Estimate of the Wait Times for $N$ and $\lambda$</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Target Maneuver Frequency</td>
</tr>
</tbody>
</table>

**Acronyms**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>APGL</td>
<td>Advanced Predictive Guidance Law</td>
</tr>
<tr>
<td>CCD</td>
<td>Charge-Coupled Device</td>
</tr>
<tr>
<td>CI</td>
<td>Confidence Interval</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DARP</td>
<td>Dial-a-Ride Problem</td>
</tr>
<tr>
<td>DNN</td>
<td>Dynamic Nearest Neighbour</td>
</tr>
<tr>
<td>DRPM</td>
<td>Dynamic Re-Pairing Methodology</td>
</tr>
<tr>
<td>DTRP</td>
<td>Dynamic Travelling Repairman Problem</td>
</tr>
<tr>
<td>EKF</td>
<td>Extended Kalman Filter</td>
</tr>
<tr>
<td>FCFS</td>
<td>First-Come-First-Served</td>
</tr>
<tr>
<td>FOV</td>
<td>Field of View</td>
</tr>
<tr>
<td>GHz</td>
<td>Gigahertz</td>
</tr>
<tr>
<td>GT1</td>
<td>Ghost Target 1</td>
</tr>
<tr>
<td>GT2</td>
<td>Ghost Target 2</td>
</tr>
<tr>
<td>GT3</td>
<td>Ghost Target 3</td>
</tr>
<tr>
<td>GT4</td>
<td>Ghost Target 4</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>GT6</td>
<td>Ghost Target 6</td>
</tr>
<tr>
<td>MTIP</td>
<td>Multi-Target Interception Problem</td>
</tr>
<tr>
<td>MTSP</td>
<td>Multiple Travelling Salesperson Problem</td>
</tr>
<tr>
<td>NN</td>
<td>Nearest Neighbour</td>
</tr>
<tr>
<td>OANL</td>
<td>Obstacle Avoidance Navigation Law</td>
</tr>
<tr>
<td>P1</td>
<td>Pursuer 1</td>
</tr>
<tr>
<td>P2</td>
<td>Pursuer 2</td>
</tr>
<tr>
<td>P3</td>
<td>Pursuer 3</td>
</tr>
<tr>
<td>P4</td>
<td>Pursuer 4</td>
</tr>
<tr>
<td>P5</td>
<td>Pursuer 5</td>
</tr>
<tr>
<td>P6</td>
<td>Pursuer 6</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PDP</td>
<td>Pickup and Delivery Problem</td>
</tr>
<tr>
<td>PN</td>
<td>Proportional Navigation</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>T1</td>
<td>Target 1</td>
</tr>
<tr>
<td>T2</td>
<td>Target 2</td>
</tr>
<tr>
<td>T3</td>
<td>Target 3</td>
</tr>
<tr>
<td>T4</td>
<td>Target 4</td>
</tr>
<tr>
<td>T5</td>
<td>Target 5</td>
</tr>
<tr>
<td>T6</td>
<td>Target 6</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>VRP</td>
<td>Vehicle Routing Problem</td>
</tr>
</tbody>
</table>
1. Introduction

1.1. Motivation

Teams of autonomous agents can be used in numerous applications where the objective at hand may be too complex for one agent to solely achieve efficiently, such as multi-target interception, and customer pick-up and delivery problems (e.g., [1]-[3]). In such situations, the objective at hand can often be divided into tasks that are achievable by an individual agent, and when completed in tandem, allow the team to achieve the desired objective. The agents in these teams are expected to operate in dynamic environments, where the numbers of agents and tasks may vary over time, and as a result, need to be able to adapt the division of tasks amongst agents to changes in the environment. Additionally, the locations of the tasks cannot always be assumed to be known a priori, but, are observable in real-time. In such scenarios, to maximize the team’s ability to achieve the desired objective: (i) the optimal on-line one-to-one pairing of tasks to agents needs to be determined, and for situations in which task location is variable with time, (ii) autonomous agent-motion planning needs to be determined simultaneously.
The pairing of agents with tasks has commonly been referred to as task allocation (e.g., [4]-[6]). For situations in which the desired objective is to be achieved in a time-efficient manner, the optimality of the pairings is essential to ensure each agent completes its task(s) successfully and ultimately that the desired objective is achieved. However, one notes that globally optimal task allocations (i.e., re-pairings) would not be achievable due to the real-time, dynamic nature of the problem. Thus, the computational complexity of the problem would necessitate the acceptance of relevant near-optimal pairings, determined on-line, as achieved subject to time constraints.

The allocation of multiple agents to multiple tasks is similar to the Multiple Traveling Salesperson Problem (MTSP), in which multiple salespeople (agents) are optimally assigned to multiple cities (tasks) to minimize total travel time [7]. Methodologies proposed in the literature for the MTSP with static cities have been mostly off-line solutions that determine the optimal salesperson–city sequence before the salespeople begin traveling [8], [9]. Naturally, such strategies with a one-time determination of pairings, would not allow the salespeople to modify their assignments while traveling so as to react to the introduction or removal of tasks from the environment. Therefore, due to their inherent inability to adapt to changes in the environment during travel, an on-line task-allocation methodology which extends these off-line solutions would be necessary.

Based on the above discussion, the shortcomings of current on-line task-allocation methodologies for two applications of the multi-agent, multi-task allocation problem will be addressed in this Thesis. These applications are: (i) the on-line dial-a-ride problem, and (ii) the interception of multiple highly-maneuverable targets.

The on-line dial-a-ride problem (DARP) involves the allocation of vehicles (agents) to customers, which require transportation from a pick-up to drop-off location (tasks) with minimum wait time. The locations and arrival times of these demands are not known a priori or in advance of their arrival, but are revealed over time. While research into on-line methodologies for this problem has gained traction over the last decade, current on-line methodologies for this problem are limited in their ability to adapt to new demands as they arrive. As a result, few methodologies exist where vehicles are able to change their allocated customer while they are travelling towards them, instead of changing their allocations once they have dropped off a
customer [10]. The ability to change allocations while travelling poses the potential to decrease customer wait times, and as a result increase the number of demands served during a specified time period.

The interception of multiple targets (tasks) by a team of robotic pursuers (agents) is a problem that has been greatly investigated, where pursuers are required to reach the locations of the targets to intercept them, usually, in minimum time. Herein, this will be referred to as the Multi-Target Interception Problem (MTIP). Methodologies to address these problems commonly account for targets which are stationary or travelling along pre-defined trajectories [5], [11], [12]. These approaches do not readily apply to scenarios in which targets are highly-maneuverable and travel along a priori unknown trajectories as they do not account for the unpredictability in target motion. Additionally, the problem of intercepting a dynamic number of highly-maneuverable targets by a dynamic number of pursuers has not been vigorously investigated.

Naturally, due to the on-line nature of the problem and the time-varying nature of the number of agents and tasks, solution strategies to these problems would necessitate the consistent re-evaluation of agent-task pairings to ensure optimality is maintained throughout objective completion. Therefore, computationally efficient methods of determining pairings are necessary to ensure pairings can be determined and executed by the agents before they become non-optimal due to changes in the environment that occurred since the start of optimization.

For situations in which task location varies over time, agent motion-planning would require a methodology that can easily adapt the motions of its agents to their allocated tasks. For the dial-a-ride problem, this involves adapting the motions of the vehicles to head towards their assigned static customers, which can be solved by changing the heading of the vehicles upon receiving a new allocation. However, for the interception problem in which pursuer motions must adapt to target maneuvers as well as potential, frequent, changes in pursuer-target pairings the problem is less trivial. The use of navigational guidance in this endeavour would allow for real-time re-pairing due to minimal computational burden in its execution [13], and allow computational efforts to be directed towards the solution of the pairing problem. Furthermore, navigational guidance yields pursuer motion-trajectories that are time optimal for the interception of highly maneuverable targets [14]. Thus, a motion-planning methodology that
incorporates navigational guidance would best complement an on-line solution methodology to the task-allocation problem for multi-target interception.

Based on the discussion above, the on-line dial-a-ride problem and multi-target interception problem require task-allocation methodologies which are able to adapt the allocations of the agents to changes in the number and behaviour of the tasks in their environments within given time constraints. Additionally, methodologies which address the multi-target interception problem need to be able to adapt the allocations to variable numbers of pursuers which require time-optimal motion planning. In this context, the focus of this research has been on the development of two unique task-allocation methodologies, one for the dial-a-ride problem, which allows the vehicles to service customer demands in the environment in minimum average time, and one for the multi-target interception problem, which allows a team of pursuers to intercept all targets in the environment in minimum total time.

1.2. Literature Review

A discussion of the pertinent literature on task allocation methodologies for the Dial-a-Ride Problem (DARP), Multi-Target Interception Problem (MTIP), and Pursuer Motion-Planning is provided below.

1.2.1. Task-Allocation for the Dial–a–Ride Problems

The DARP is a subclass of the Vehicle Routing Problem (VRP) which strives to serve a set of customers’ demands with a fleet of vehicles in an effective and efficient manner. In such problems, the set of customer demands is allocated to a fleet of vehicles such that an optimal route (or tour) around those demands is determined for each vehicle such that an objective for the fleet is optimized [15]. The Pick-up and Delivery Problem (PDP) is the class of the VRP to which the DARP subclass belongs. In PDPs, customer demand locations arrive in pairs where the customer requests an entity to be picked up at an origin location and delivered to a destination location [10]. This type of problem imposes additional constraints on the VRP solution, as the constraint that the drop-off must occur after the pick-up limits the number of possible routes. PDPs are applicable to courier services, goods transportation fleets, emergency vehicle routing and taxi-cab dispatching. Within these applications, PDPs have been investigated for the minimization of, route makespan [16], customer wait times [17], and routing costs [18].
Off-line solutions for the DARP have been vigorously investigated in the literature given the premise that product distribution systems and courier service demands can be called in to the dispatching centre a day in advance [15]. This application of the DARP allows for a full off-line optimization of the demands to determine an optimal route pre-execution. Much research effort has been dedicated to developing methodologies to address the single and multiple capacity vehicle cases of this problem while ensuring customers are picked up and dropped off within their respective desired time windows [19]. For situations in which vehicles have multiple capacity and can pick-up additional customers while transporting an initial customer to its destination, the development of methodologies to ensure the maximum ride time of a customer is within acceptable bounds has additionally been of interest. Due to the NP hard nature of the problem, most methods to address this problem have been heuristics. Classical and modern meta-heuristics used to address off-line instances of these problems are detailed in [20].

However, due to the real-time nature of many dial-a-ride systems, on-line solutions have become necessary to dynamically adapt off-line solutions to stochastic events that occur during route execution. The justification for focusing on developing methodologies to adapt off-line solutions to dynamic events is that a subset of requests is often known in advance and, therefore, the solutions only need to adapt to the unexpected demands that are revealed over time [15]. Stochastic events for such systems can comprise a new customer demand, as well as a vehicle breakdown, customer cancellations, or customer no-show. The most commonly investigated of these events is the arrival of an unexpected customer during execution, at which point vehicle routes are re-optimized from the unexpected customers arrival onward [21]. However, re-planning the entire route of each vehicle may not be solvable due to computation time and, as a result, solutions which pre-compute several possible scenarios based on anticipated future requests have been developed [22]. To address the computational time limits of such solutions, much research has been devoted to insertion algorithms, which determine a near-optimal placement for the customer within time constraints [23], [24].

Re-optimization of the entire routing problem can yield more optimal results than insertion techniques when dynamic events require adaptations to the sequence of demands in the original route. These can be performed post-insertion algorithms to get a more optimal routing sequence [25], or immediately from the point of the unexpected demand [26]. While for small numbers of vehicles and demands it may be feasible to re-solve the entire routing problem, for large numbers
it is not, and heuristics are required to ensure a solution is found in a reasonable amount of time. Tabu search appears to be the most popular heuristic used in vehicle routing [20], while other meta-heuristics such as simulated annealing, genetic algorithms, and guided local search are also widely used [27].

The purely dynamic version of the VRP, where no demands are known \textit{a priori}, is discussed in [28] in terms of the Dynamic TSP. Here, customer demands arrive according to a Poisson process, where a server travels to demands and spends a stochastic amount of time at each while aiming to either maximize the number of demands served, or minimize the time each demand spends in the system. Based on this work, policies have been greatly developed for a variant of the purely dynamic VRP called the Dynamic Travelling Repairman Problem (DTRP), where a repairman services a demand on site for a stochastic amount of time instead of transporting a customer from an origin to destination as in the DARP. The focus of this research in [29], [30] has been on minimizing the mean system time, which consists of the time the demand waits to be served, plus the time it takes to serve the demand. In these papers, the lower bound on mean system time for single and multiple vehicle fleets for an unknown optimal policy was obtained and compared to several policies. The most common policy discussed is First-Come-First-Served (FCFS), in which demands are serviced in the order they enter the system. Several variants of the original FCFS policy have been presented including Partitioning, Stochastic Queue Median, and Sectoring Policy with the aim of reducing the mean system time to address all demands. While these policies are promising for light traffic (few demands) scenarios, they become increasingly unstable as the traffic level increases. For heavy traffic levels, the Nearest Neighbour (NN) policy, where vehicles are allocated to the closest waiting demand upon completion of service, provides the minimum mean system time.

While the above policies have shown promise in reducing the mean system time of \textit{a priori} unknown demands, they have the potential to be greatly improved through the incorporation of diversion strategies. In diversion strategies, the allocation between a vehicle and demand is not fixed until the demand is addressed, and can be changed (or re-allocated) if a more optimal vehicle-demand pairing arises from new information in the system, such as a new demand entering the system, or a vehicle becoming idle (having finished addressing a demand). The availability of real-time global positioning information about fleet vehicles has made real-time dynamic routing decisions feasible [31]. The knowledge of where the vehicle is at the time of a
demand allows the re-optimization of allocations to consider the vehicles next demand in the optimization where it previously could not. Diversion strategies have not received much attention in the literature, but have been demonstrated to provide substantial benefits for two variants of the on-line DARP: truck-load carrier problems [32]-[34], and courier services [35].

1.2.2. Task-Allocation for Multi-Target Interception Problems
MTIPs are an application of the MTSP where multiple pursuer agents are allocated to and intercept (move to the same position as) multiple targets according to a metric to be optimized, such as time or distance. The MTIP can be seen as an extension of the DARP where instead of a constant number of vehicles moving towards a dynamic number of static customer pick-up locations, a variable number of pursuers are required to intercept a dynamic number of highly-maneuverable targets. MTIPs can be subdivided, similar to DARPs, into off-line and on-line solution strategies and their applications, which are highly dependent on the variability in the numbers of targets and pursuers, as well as the amount of a priori information about them. Applications and extensions of the MTIP include military combat [4], multi-target search and rescue [36], and multi-target tracking [37].

There are three common interception objectives for the MTIP which dictate the type of pursuer-target pairings to be determined. The first interception objective is simultaneous interception, where each pursuer attempts to intercept its allocated target at the same time, and is most commonly investigated for UAVs and combat applications [38]. The second objective, time-optimal interception, is where a team of pursuers attempts to intercept all targets as fast as possible. This area has received the majority of attention in the literature, and is the type of interception problem addressed in this Thesis and [39], [40]. For the first and second objectives, pursuer-target pairings are formulated in a one-to-one manner, where one pursuer is paired with and intercepts one target at a time. The third objective, containment, where evasive targets are contained within a specified area by a group of pursuers, often pairs multiple pursuers with each target so that the pursuers can surround and contain the target [41]. However, regardless of the objective, the optimal determination of optimal pursuer-target pairings is necessary to ensure its successful completion.

Off-line solutions to the MTIP determine pursuer-target pairings only once before the onset of pursuit and, then, are executed by the pursuers without any changes. These methods perform
pursuer-target pairings only once either globally at the start of travel [38], or locally in search scenarios once a pursuer has detected a nearby target [42]. Such solutions are readily applicable to static environments where there is extensive \textit{a priori} knowledge of target behaviour, such as the location of static targets, or targets moving along pre-defined trajectories. In [40], a brute-force enumeration was used to pair pursuers to targets with pre-defined trajectories for simulated interception in minimum total and average time. Satisficing decision theory was utilized in [5] to determine appropriate pairings off-line for a simultaneous UAV-target interception problem where the static position of targets are known \textit{a priori}. Polynomial interpolation and curve fitting is used in [11] to determine the path along which a pursuer would intercept two targets with pre-defined trajectories in minimum time. For such static scenarios a one-time determination of pursuer-target allocations may suffice, however, for situations which involve dynamic environments as well as \textit{a priori} unknown target behaviour, off-line methodologies would not suffice as the optimality of the initial pairing combination could not be guaranteed to hold throughout pursuit.

For scenarios where target motion is not known \textit{a priori} an optimal sequence of target interceptions cannot be planned off-line, warranting on-line solution methodologies to address the MTIP. As with the DARP, the on-line MTIP is a combinatorial optimization problem which requires an efficient method of determining satisfactorily optimal pairings within given time constraints. When targets are highly-maneuverable and no information about their behaviour is known \textit{a priori}, on-line solutions are necessary to be able to adapt the optimal pairing solution to changes in the environment such as unexpected obstacles, target pop-ups and target motion throughout the interception. Due to the combinatorial nature of the problem as discussed above with respect to the DARP, and the need to consistently re-evaluate the pairings based on target motion, heuristics are again the most common method of determining satisfactorily optimal pairings on-line within given time constraints. A branch-and-bound solution to an on-line optimization problem for highly-maneuverable targets is presented in [43]. In [4], an on-demand interception system is presented where weapons are assigned to targets one at a time as the target’s interception deadline approaches, allowing for those of greatest interest to hold precedence. In [6], tasks are assigned to robots according to an automaton such that tasks are assigned in a specific order, and only to robots capable of completing them. An algorithm is
presented for on-line re-assignment of UAVs containing and intercepting evading targets using a differential game theory framework in [41].

The aforementioned techniques, however, typically assume a constant number of pursuers and targets throughout the pursuit. On-line assignment methods, however, may need to deal with scenarios in which the numbers of pursuers and targets varies dynamically over the course of the pursuit. While some attention has been given to problems in which the number of targets is dynamic, very little has been done for dynamic numbers of pursuers in centralized task-allocation systems. In [44], a centralized auction method is used to allocate a fixed number of pursuers to static targets according to an NN policy, where targets pop-up at random locations. In [45], a distributed on-line solution strategy is presented for a group of pursuers to address static targets that appear at pre-defined locations at random intervals in time. Each pursuer periodically determines its optimal assignment independently using data compiled from all pursuers. However, while these techniques account for a variable number of static targets, they do not account for variability in the motion of these targets, nor in the number of pursuers during pursuit.

1.2.3. Pursuer Motion-Planning

Unlike the methods to solve the DARP, which do not deal greatly with the motion planning of vehicles since their demands are at static locations, in the MTIP much research is devoted to the development of efficient pursuer motion-planning methodologies that complement the task-allocation methodologies used. This is due to the fact that in such dynamic scenarios, the efficiency of task allocation is closely dependent on the ability of the pursuers to accomplish their tasks in a time-optimal manner. As a result, motion-planning methodologies which can move the pursuer towards its allocated target, while accounting for its ability to maneuver, in the least amount of time are necessary.

Centralized multi-target optimal interception methodologies in the literature, include path planning through cell decomposition [39], polynomial interpolation trajectory generation [11], and the development of waypoints through Voronoi graphs [38]. However, these approaches, while well suited to applications where targets are static or move along pre-defined trajectories, are not readily applicable to targets with a high degree of maneuverability. Furthermore, they are subject to significant computational burden which may decrease their ability to respond to target
maneuvers in a time-efficient manner, as well as take away processing power from the determination of optimal task-allocations.

Guidance theories have been used to navigate interceptors toward targets with highly-maneuverable motions, first in missile-guidance applications [46]-[48], and more recently in mobile robotics [13], [14], [49], [50]. In navigation guidance, a pursuer is accelerated to move towards an estimated interception point on the collision triangle between the pursuer and target [51]. As the motion of the target changes, this estimated interception point also changes, varying the acceleration which is applied to the pursuer, and allowing it to efficiently navigate towards the estimated interception location. These techniques are well suited to dynamic multi-target interception problems, as they can adapt to the targets’ movements quickly due to the minimal computational burden, allowing for effective continuous on-line re-planning of the pursuers’ movements.

Given the dynamic environment of the MTIP, obstacle avoidance needs to be incorporated into the motion-planning of the pursuers to ensure there are no collisions between them. For scenarios in which the trajectories or locations of targets are fixed, the avoidance of dynamic obstacles in the workspace has been addressed solely using the navigation of the pursuers [52] [53]. Namely, pursuer-target pairings are determined initially off-line, while the paths the pursuers traverse in order to intercept their targets are re-planned on-line to account for changes in the environment, such as obstacles and pop-up threats [5]. However, these approaches do not account for deviations in the pursuers’ trajectories required during navigation to avoid these obstacles in the determination of the optimal task-allocation solution. Therefore, it is possible that due to such deviations, a pursuer may become better suited to intercept a different target. As a result, the only manner in which to ensure that on-line motion planning with obstacle avoidance does not affect the optimality of the pairing solution, is to have both determined on-line.

1.3. Research Objectives

The objective of this proposed research is the development of two on-line task re-allocation methodologies, one for the DARP and one for the MTIP, which are capable of re-allocating agents to tasks for minimum task completion time in a real-time implementable manner.
For the DARP, it is envisioned that the developed methodology would be able to determine pairings to minimize customer system times regardless of traffic intensity, and customer demand arrival patterns. This methodology would not require any *a priori* information on arrival patterns or locations, however it is assumed that once a customer demand enters the system its location would be made available.

For the MTIP, it is envisioned that the optimal on-line task-allocation methodology would be able provide a team of autonomous pursuers with the ability to intercept multiple targets in a time efficient manner within 2D or 3D dynamic environments. The proposed generic multi-pursuer multi-target re-pairing methodology will account for possible variations of the interception problem that multi-agent teams may encounter. For example, after an interception, a pursuer can *disengage* from its incapacitated target and continue to its next allocated target, or can be *consumed* during the interception. The methodology will not require any *a priori* information about the number of pursuers and targets, nor about the motions of the targets. However, it is assumed that the locations and velocities of the targets and pursuers will be able to be determined in real-time through the use of external sensors.

This Thesis presents the successful applicability of the proposed task allocation methodologies to the DARP through simulations in the Automod environment, and the MTIP through simulations in the C++ environment, and through MTIP experimentation with three mobile robots. The simulation testing environments for both problems assumed perfect instantaneous information about tasks’ and agents’ locations and velocities. During the experimental phase the task-allocation and motion planning methodologies for the MTIP were tested against system lag and noise.

### 1.4. Thesis Organization

This Thesis presents two time-optimal on-line task re-allocation methodologies for a team of agents to address *a priori* unknown tasks with minimum mean task-completion time for the DARP and minimum total task-completion time for the MTIP. The structure of this thesis is as follows:

- **Chapter 1** has presented the motivation and background information about the problems considered.
Chapter 2 presents the problem formulation of the dial-a-ride problem addressed, and introduces the novel Dynamic Nearest Neighbour (DNN) policy. It compares the functionality of the DNN policy to the FCFS and NN policies under uniform and city-like environments. Finally the functionality of the DNN policy when customer wait-time caps and vehicle anticipatory behaviour are incorporated is presented. The improvements of DNN over FCFS and NN for all conditions are demonstrated through simulation examples.

Chapter 3 presents the problem formulation of the multi-target interception problem addressed, and introduces the novel generic Dynamic Re-Pairing Methodology. The architecture of the methodology is presented along with implementation options that allow the methodology to function generically, or be customized to any specific instance of the MTIP. The functionality of the re-allocation method in reducing the total interception time as compared to fixed assignments is presented, along with re-pairing examples demonstrating the methodologies ability to determine pairings for minimum total interception time.

Chapter 4 presents the experimental results of the Dynamic Re-Pairing Methodology. The experimental setup is presented along with the vision system, communication system and robotic vehicles. The functionality of the re-allocation methodology to adapt to changes in target maneuver as well as numbers of pursuers and targets is again confirmed through experimental examples.

Chapter 5 presents the conclusions of the Thesis and recommendations for future work.
2. Dynamic Nearest Neighbour Policy

The on-line dial-a-ride problem (DARP) involves the allocation of vehicles (agents) to customers, who demand transportation from a pick-up to drop-off location (tasks) with minimum wait time, as these customer demands are revealed over time. While many applications of the DARP exist, such as the dispatching of emergency vehicles, and transportation networks for the physically handicapped, the application specifically addressed in this thesis is that of a taxi-cab dispatching system, although the policy presented can be utilized for any dynamic DARP. This system utilizes real-time information about the dynamic environments in which the vehicles operate to make decisions about vehicle to customer-demand allocations on-line in real-time. In this context, the first task-allocation problem addressed in this Thesis consists of the development of a policy to allocate, and re-allocate, a fixed number of vehicles to a dynamic number of customer demands as they arrive in time, with no a priori knowledge of the number or location of customer demands to arrive, for a lower mean system time than existing policies.

This chapter presents the Dynamic Nearest Neighbour (DNN) policy that can be used to re-allocate an a priori unknown and dynamic number of customer demands to vehicles on-line to
minimize the mean system time of the customers. Section 2.1 constructs the problem definition of the task-allocation problem. Section 2.2 presents the proposed DNN Policy, its anticipatory behaviour policy, and efforts to reduce the longest customer wait times. Finally, simulation results demonstrating DNNs ability to re-allocate vehicles to customer demands for a decreased mean system time as compared to two well-established policies, First-Come-First-Served (FCFS) and Nearest Neighbour (NN), are presented in Section 2.3.

2.1. Problem Definition

The objective of this work is the development of a policy to allocate and re-allocate (or route and re-route) vehicles to customer demands on-line, as these demands arrive over time, to minimize the mean system time of the customers in the system. No a priori information about the customer locations or time of arrival of their demands is available before the demand enters the system.

These customer demands appear randomly in a 2D square region, \( A \), of area \( A \) and are served by a fleet of \( N \) vehicles (or taxis) moving at a constant velocity \( v \) throughout service. Customer demands arrive in time according to a Poisson process with a rate of \( \lambda \), where a demand consists of a call placed by the customer to the system to request service. Each customer’s demand is independent, with uniformly distributed pick-up (origin) and drop-off (destination) locations.

However, given the nature of the taxi-cab dispatching application of the DARP, the policy developed must be able to account for variations in customer arrival patterns over the course of a day. This stems from the fact that the majority of taxi-cab dispatching companies are located in cities where the location and quantity of demands varies over the course of the day due to transportation trends in morning and evening rush hours as customers move from the suburbs to downtown area, and vice versa. As a result, the policy developed must not only work for the idealistic setting discussed above where customer demands are located uniformly in space with a constant arrival rate over the day, but also for realistic settings, such as cities.

The system time of a customer is defined as:

\[
T = W + S ,
\]  

(2.1)
where \( T \) represents the system time of the customer, which is comprised of, \( W \), the wait time of the customer, and \( S \), the service time of the customer. The \textit{wait time} of a customer is defined as the elapsed time from the instant the customer’s pick-up call arrives in the system until the instant they are picked up by the vehicle, and the \textit{service time} of the customer is the elapsed time from the instant the vehicle has picked-up the customer at their origin location until the instant the vehicle arrives at the customer’s desired destination.

The service time of a customer is stochastically the same across all policies where the service of a customer is not pre-empted. This means, that while a vehicle is transporting a customer (i.e. the customer is in the vehicle) it cannot be re-allocated to another customer, forcing the vehicle to drop-off the customer at an intermediate location between their initial pick-up and drop-off locations and begin moving towards an alternate customer. The problem addressed in this Thesis does not deal with pre-emption of service, and as a result, the service time of a customer is the time it takes to traverse the Euclidean distance between two uniformly and independently distributed (pick-up and drop-off) points, \( X_1 \) and \( X_2 \), in a square area, \( A \). The first two moments of the service time are, then, defined as \[ 54 \] \[ 55 \]:

\[
E[S] = \frac{E[||X_1 - X_2||]}{v} = \frac{c_1 \sqrt{A}}{v}, \tag{2.2}
\]
\[
E[S^2] = \frac{E[||X_1 - X_2||^2]}{v^2} = \frac{c_2 A}{v^2}, \tag{2.3}
\]

where \( c_1 \approx 0.52 \) and \( c_2 = 1/3 \). As a result, the expected service time of all customers in the system is constant, where its expected value is given by Eq. (2.2) based on system parameters.

As outlined in \[ 29 \], with respect to a variant of the DARP, the Dynamic Travelling Repairman Problem (DTRP), the system time under consideration may exhibit some similarities to the system time of the classical Markovian/General/N-server \((M/G/N)\) queue that has the same arrival (call) process and the same service time distribution as that of \( S \), the service time of the customer in the DARP system. In these queues, arrivals occur according to a mean rate of \( \lambda \), and are served by \( N \) servers, where the service time distribution is arbitrary \[ 55 \]. However, while in the classical \( M/G/N \) queue wait time is generated only through servers (vehicles) serving other customers, in the DARP the wait time consists of this component as well as the time it takes for a vehicle to move from its previous drop-off location to a new customer. Therefore, any policy
which minimizes the amount of time that vehicles spend while driving in-between customers, minimizes the mean system time of the customers.

2.2. **Dynamic Nearest Neighbour Policy (DNN)**

The DNN policy proposed in this Thesis improves upon the performance of the NN policy through re-pairing a vehicle to a closer customer whenever one becomes available, regardless of whether the vehicle is travelling towards a previously assigned customer or idle. For the dynamic DARP addressed in this thesis, a vehicle can be idle, assigned, or busy, and a customer can be unassigned, assigned, or a customer-in-service. Vehicles and customers cannot exist in any state other than those described above and those which will be defined later in this Section. A vehicle is denoted as *assigned* when it is paired with a customer and is en-route towards their pick-up location. A vehicle is *busy* when it has already picked up a customer and is travelling towards their drop-off location. Otherwise, the vehicle is assumed to be *idle*. Similarly, a customer who is paired with a vehicle is denoted as an *assigned* customer, and as a *customer-in-service* when travelling in a vehicle to their destination. All others, who placed calls but have not been paired with a vehicle, are denoted as *unassigned* customers.

Under the NN policy [30], when a new customer demand enters the system, they are assigned to the nearest available idle vehicle, if one exists. Otherwise, the new customer becomes an unassigned customer and must wait for service until they become the nearest customer to a vehicle that has become idle. Similarly, upon dropping off a customer, the idle vehicle is assigned to the unassigned customer with the shortest distance from the vehicle’s current location, if one exists. Otherwise, the vehicle remains idle.

As can be seen from the NN policy, once a vehicle has been assigned to a customer, it cannot be re-assigned until it delivers its respective customer to their destination. For instance, if a new customer who is closer to an assigned vehicle than its original customer enters the system, the vehicle cannot be reassigned to this new customer even if deemed beneficial. Similarly, if no unassigned customers exist and a vehicle becomes idle that is closer to an assigned customer than its originally assigned vehicle, the assignment cannot change, and the newly idle vehicle must remain idle until a new customer arrives. Thus, instances may arise when the re-pairing of a vehicle could lead to a lower individual system time of the closer customer; however, since this is not allowed under the NN policy, the outcome can be a potential increase in mean system
time. This premise leads to the development of the DNN policy, which extends upon the NN policy to allow for re-pairings in vehicle-customer assignments, where re-pairing would lead to a decrease in a customer’s individual system time, and ultimately the mean system time of all customers served in the system. One may note that an assigned (or unassigned) customer would not know that they are in fact paired (or not paired yet) with a vehicle until it shows up at their pick-up location, and as a result any changes in the vehicle assigned to pick them up would be invisible to them.

To ensure such re-pairings are possible, the DNN policy ensures that: (i) each assigned customer is paired with the vehicle it is closest to, provided that there is no other customer who is closer to that vehicle, and (ii) each assigned vehicle is paired with the closest customer waiting, provided that there is no other vehicle which is closer to that customer. The task-allocation problem at hand is one of combinatorial nature – real-time, re-pairing of vehicles and customers as events occur in the system. These events can be a vehicle becoming idle or a new customer demand entering the system.

In this context, let us first define a \((K \times N)\) binary matrix \(Y\) that represents the vehicle-customer pairings, where:

\[
y_{ij} = \begin{cases} 
1 & \text{if Customer } i \text{ is paired with Vehicle } j, \\
0 & \text{otherwise;}
\end{cases} \tag{2.4}
\]

subject to

\[
\sum_{i=1}^{K} \sum_{j=1}^{N} y_{ij} = \min (K, N) , \tag{2.5}
\]

\[
\sum_{j=1}^{N} y_{ij} = \begin{cases} 
1 & \text{if Customer } i \text{ is paired, } \forall i , \\
0 & \text{otherwise;}
\end{cases} \tag{2.6}
\]

and

\[
\sum_{i=1}^{K} y_{ij} = \begin{cases} 
1 & \text{if Vehicle } j \text{ is paired, } \forall j , \\
0 & \text{otherwise;}
\end{cases} \tag{2.7}
\]

Above, \(K\) is the number of unassigned and assigned customers and \(N\) is the number of not-busy vehicles in the system at the time of the re-pairing. Each vehicle is paired with a unique customer at any event update. The vehicle-customer pairings matrix \(Y\) is resized according to the changes in \(K\) and \(N\) so that only those vehicles which are not busy and those customers which are unassigned or assigned are considered.

In order to determine the instantaneous optimal pairings, let \(D\) represent the \((K \times N)\) matrix of non-negative Euclidean distances between Customer \(i\) and Vehicle \(j\) at the time of re-pairing,
where the indices in \( D \) correspond to those in \( Y \). Then, to ensure each vehicle is paired with the nearest customer, the DNN policy minimizes the following objective function at each re-pairing:

\[
\min_{i=1}^{K} \sum_{j=1}^{N} (y_{i,j} \times d_{i,j}).
\]  

\( (2.8) \)

### 2.2.1. DNN Algorithm

As discussed in the above policy description, the objective function re-evaluation is triggered at each major event update in the system. These events consist of, Event \( A \), a new customer demand entering the system, and Event \( B \), a vehicle becoming idle having reached its customer drop-off point. The occurrence of these events triggers a new solution of Eq. \( (2.8) \), where one must re-solve the model in Eqs. \( (2.4) – (2.8) \), which may consequently change some or all of the previous customer-vehicle assignments if they are no longer deemed optimal.

At minor event updates, such as a vehicle reaching its assigned customer’s pick-up location, Event \( C \), new assignments are not made; instead, the assigned vehicle becomes a busy vehicle and its assigned customer becomes a customer-in-service. This results in a decrease in \( N \) and \( K \) for the next re-evaluation of the model, but does not trigger such a re-evaluation at the event’s occurrence.

The computational complexity of seeking globally optimal re-pairing at any instant, by searching through all possible pairing solutions, would make the problem intractable for even modest numbers of customers and vehicles. Therefore, the following DNN Algorithm was developed to determine near-optimal pairings for a fleet of vehicles which lead to a lower mean system time than other policies. For \( N = 1 \), the results of this algorithm obtain results identical to that of a brute-force enumeration, and for \( N > 1 \), results similar to those of a brute-force enumeration but with less computational burden and processing time.

The DNN Algorithm is as follows:

**If Event \( A \) occurs: A new customer appears**

1. (a) If an idle or an assigned vehicle exists, determine the closest idle or assigned vehicle that is strictly closer (not equidistant) to the new customer than the customer to which it is currently assigned:
(i) If this vehicle is an assigned vehicle, unassign it from its previously assigned customer and proceed to Step 2;

(ii) If this vehicle is an idle vehicle, proceed to Step 3.

(b) If assigned vehicles exist, but none are strictly closer to the new customer than their currently assigned customer, proceed to Step 4.

(c) If all vehicles are busy, proceed to Step 4.

2. Assign the vehicle to the new customer. The new customer becomes an assigned customer and the previously assigned customer becomes unassigned. Return to Step 1, and treat the now unassigned customer as a new customer.

3. Assign the vehicle to the new customer. The vehicle and the new customer become an assigned vehicle and an assigned customer, respectively. Exit algorithm and await new trigger.

4. The new customer is denoted as an unassigned customer and starts waiting in a queue. Exit algorithm and await new trigger.

If Event B occurs: A vehicle becomes idle

5. (a) If an assigned or unassigned customer exists, determine the closest unassigned customer or assigned customer whose vehicle is strictly farther away (not equidistant) than the currently idle vehicle:

   (i) If the customer is an assigned customer, unassign them from their previously assigned vehicle and proceed to Step 6;

   (ii) If the customer is an unassigned customer, proceed to Step 7.

(b) If assigned customers exist, but none are strictly farther away from their currently assigned vehicle than the idle vehicle, proceed to Step 8.

(c) If there are only customers-in-service, proceed to Step 8.

6. Assign the newly idle vehicle to the assigned customer. The idle vehicle becomes an assigned vehicle and the previously assigned vehicle becomes unassigned. Return to Step 5, and treat the unassigned vehicle as a new idle vehicle.

7. Assign the newly idle vehicle to the unassigned customer. The vehicle and the new customer become an assigned vehicle and an assigned customer, respectively. Exit algorithm and await new trigger.

8. Keep the vehicle as idle at its current location. Exit algorithm and await new trigger.

If Event C occurs: A vehicle arrives at its assigned customer’s pick-up location

2.2.2. Anticipatory Behaviour

So far with the DNN algorithm, vehicles that have become idle remain at their last customer’s drop-off location until they are next called for service. However, the time a customer spends waiting for service could be further minimized if the vehicles, when not in use, were routed to be distributed about the area in a manner respective of the demands to arrive. Moving vehicles in an anticipatory behaviour of the demands to arrive provides a decrease in individual customer system times by decreasing the distance a vehicle must travel to reach the customer’s pick-up location. The benefits of anticipatory behaviour are not only evident at times when demands are uniformly distributed in the area, but even more so at times when city-like transportation trends occur. In real life, such vehicles would usually return to some stations, or spots, to wait for new customers.

The proposed modified DNN policy aims at distributing vehicles uniformly over the area through the use of cab stops (or vehicle wait stations) according to the anticipated customer arrival patterns. This is the most common anticipatory behaviour practice for taxi-cab systems in real-life. However, it can be noted that anticipatory behaviour methods other than vehicle stations can be used to provide such a distribution of vehicles about the area. With this policy, all the vehicle “knows” is that it has been assigned to move to a location; where this location is, if it calls a vehicle station, or whether it is fixed or moving does not matter.

In this Thesis, the issue of the optimal number of vehicle stations and how to choose their optimal locations is not addressed. It is only assumed that the number of vehicle stations needs to be approximately equal to the number of vehicles, \( N \), (to decrease the expected distance to reach a customer demand the most) and that their distribution across the area needs to be dynamic and representative of the variations in arrival patterns. When the arrival rate is constant and the demand locations are uniformly distributed, this number of vehicle stations should be uniformly distributed about the entire area. However, when dealing with city-like conditions where the distribution of the location of demands may vary over the course of a day due to rush hour trends, the location of vehicle stations must reflect these trends. It is expected that rush hour trends result in pick-ups that are in an area of high demand, and drop-offs in an area of low demand, following that of a suburbs to downtown and vice versa daily commute. Therefore, when the pick-up locations of customers tend towards one section of the area (high-demand
area), one has to have more stations in the high-demand area, and fewer in the low-demand area, so as to ensure the distribution of vehicles is representative of the arrival patterns.

2.2.2.1. **Anticipatory DNN Policy**

Given a distribution of vehicle stations, the modified (anticipatory) DNN policy is as follows: At times when the distribution of demands is expected to be uniform across the area, all idle vehicles are directed to the closest vehicle station to them that has the fewest vehicles assigned to it. This creates an additional state in which the vehicles can exist, which is $V_{Sassigned}$, or vehicle station assigned, to represent the vehicle being assigned to a vehicle station. Herein, a $V_{Sassigned}$ vehicle refers to those vehicles waiting at a station as well as those that are en-route to a station. When an event trigger forces a re-evaluation of the model, if a $V_{Sassigned}$ vehicle that was en-route to the vehicle station is re-assigned to a customer demand, the vehicle immediately diverts its motion from heading towards the vehicle station to that of its newly assigned customer.

The assignment of vehicles to vehicle stations depends on the demand arrival pattern at the time of the assignment to the vehicle station. When the arrival pattern is uniform, the vehicle is routed to the closest vehicle station to its drop-off location with the least amount of $V_{Sassigned}$ vehicles. When the arrival pattern is not uniform across the area, the number of vehicles routed to the stations in either of the high- or low-demand areas is representative of the fraction of demands of the entire area expected in each of the high- or low-demand areas. In this case, it is first determined whether the vehicle should be routed to the high- or low-demand area, and then are assigned to the closest vehicle station that has the fewest vehicles assigned to it in that area.

Vehicles whose drop-off location is in the area of high demand are routed to their closest vehicle station in the high-demand area which has the fewest vehicles assigned to it. For vehicles whose drop-off locations are in the low-demand area, to determine which area a cab is to be routed to, a percent of cabs to be routed back to the high-demand area is first determined. Based on previous customer arrival patterns, or general demand expectation, the percentage of customer demands that follow the rush hour trend and the percentage of those whose pick-up and drop-off locations are still independently and uniformly distributed across the area first needs to be determined. From this information, the number of cabs to be routed back to the area of high
demand from the low-demand area can be determined. The number of cabs routed back to the high-demand area is:

\[ P_r = \frac{H_s - H_e}{H_e}, \quad (2.9) \]

\[ H_s = (P_h \cdot D_u) + D_h \quad (2.10) \]

\[ H_e = ((1 - P_h) \cdot D_u) \quad (2.11) \]

\[ D_u + D_h = 1 \quad (2.12) \]

where, \( P_r \) represents the percentage of vehicles to be routed from the low-demand area to vehicle stations in the high-demand area after becoming idle, \( H_s \) represents the percentage of customers whose pick-up locations are in the high-demand area, \( H_e \) represents the percentage of customers whose drop-off locations are in the high-demand area, \( P_h \) represents the percentage of the entire area \( A \) that comprises the high-demand area, \( D_u \) represents the percentage of all customer demands that are distributed uniformly across \( A \), and \( D_h \) represents the percentage of all customer demands that occur following the rush hour trends, starting in the high-demand area and ending in the low-demand area.

This incorporation of anticipatory behaviour to the DNN policy, results in modifications being made to Steps (1), and (8) of the policy as follows:

1. (a) If an idle, VSassigned or assigned vehicle exists, determine the closest idle, VSassigned, or assigned vehicle that is strictly closer (not equidistant) to the new customer than the customer to which it is currently assigned:

   (i) If this vehicle is an assigned vehicle, unassign it from its previously assigned customer and proceed to Step 2;

   (ii) If this vehicle is an idle or VSassigned vehicle, proceed to Step 3.

(b) If assigned vehicles exist, but none are strictly closer to the new customer than their currently assigned customer, proceed to Step 4.

(c) If all vehicles are busy, proceed to Step 4.

8. (a) If there is no anticipatory behaviour, keep the vehicle as idle at its current location. Exit algorithm and await new trigger.
(b) If customer demand is uniform across $\mathcal{A}$, assign the idle vehicle to the closest vehicle station that has the least amount of vehicles assigned to it. The idle vehicle becomes a $V$Salloined vehicle. Exit algorithm and await new trigger.

(c) If customer demand is not-uniform across $\mathcal{A}$, and the vehicle’s drop-off was in the high-demand area, assign the idle vehicle to the closest vehicle station in that area that has the least amount of vehicles assigned to it. The idle vehicle becomes a $V$Salloined vehicle. Exit algorithm and await new trigger.

(d) If customer demand is not-uniform across $\mathcal{A}$, and the vehicle’s drop-off was in the low-demand area:

(i) If a random probability generated is less than $P_r$, assign the idle vehicle to the closest vehicle station in the high-demand area that has the least amount of vehicles assigned to it. The idle vehicle becomes a $V$Salloined vehicle. Exit algorithm and await new trigger.

(ii) If a random probability generated is greater than $P_r$ assign the idle vehicle to the closest vehicle station in the low-demand area that has the least amount of vehicles assigned to it. The idle vehicle becomes a $V$Salloined vehicle. Exit algorithm and await new trigger.

2.2.3. Customer Wait-Time Limits

While the DNN policy attempts to decrease the mean system time through re-routing vehicles to customers which are located closer to them, it does not assure that some customers would not experience excessive wait times. Given the nature of re-routing vehicles to those closer creates some customer outliers in areas that are not highly traversed, for example along the edges, or in the corners, of $\mathcal{A}$. Therefore to attempt to address these demands in a timely manner as well, a customer wait-time limit is incorporated into the DNN policy. The optimality of the actual time limit on wait times employed is beyond the scope of this Thesis, however its dependence on the number of customer demands arriving, $\lambda$, and the number of the vehicles to serve them, $N$, is noted. As a result, a value for this limit must be selected which is dependent on $\lambda$ and $N$ such that it is not so low that the majority of customers are past it, and not so high that customers do not reach it.

The proposed DNN policy is modified herein such that the use of this limit creates two classes of customer demands, normal and high-priority. If a customer’s waiting time reaches this limit, the status of this customer is prioritized. Once a change in a customer’s status is noted, the closest vehicle to that customer, which was previously assigned to another customer, is
redirected to this high-priority customer and serves them as an *absolute* assignment. This means that once a vehicle is assigned to a high-priority customer, there is no possibility of it being re-paired with another customer until that customer has been dropped off. This re-assignment, naturally, would lead to a re-evaluation of all assignments at that point in time, and could lead to the re-pairing of the remaining assigned vehicles and customers.

The DNN policy is modified such that a new event, Event $D$, is introduced which represents a customer reaching the wait time cap. The DNN Algorithm is modified as follows to incorporate this event, with Steps (1) and (5) modified to ensure the priority assignment is absolute:

1. (a) If an *idle*, $VS_{\text{assigned}}$ or *assigned* vehicle which is not paired with a priority customer *exists*, determine the closest idle, $VS_{\text{assigned}}$ or assigned vehicle that is *strictly closer* (not equidistant) to the new customer than the customer to which it is currently assigned:
   
   (i) If this vehicle is an *assigned* vehicle, unassign it from its previously assigned customer and *proceed to Step 2*;
   
   (ii) If this vehicle is an *idle* vehicle, *proceed to Step 3*.

   (b) If *assigned* vehicles *exist*, but none are *strictly closer* to the new customer than its currently assigned customer, *proceed to Step 4*.

   (c) If all vehicles are *busy*, *proceed to Step 4*.

5. (a) If an *assigned* or *unassigned* customer *exists*, determine the closest unassigned customer or assigned non-priority customer whose vehicle is *strictly farther away* (not equidistant) than the currently idle vehicle:
   
   (i) If the customer is an *assigned* customer, unassign them from their previously assigned vehicle and *proceed to Step 6*;
   
   (ii) If the customer is an *unassigned* customer, *proceed to Step 7*.

   (b) If *assigned* customers *exist*, but none are strictly farther away from their currently assigned vehicle than the idle vehicle, *proceed to Step 8*.

   (c) If there are only *customers-in-service*, *proceed to Step 8*.

**If Event $D$ occurs:** A customer reaches the wait-time limit

10. *Assign* the priority customer to the vehicle which is closest to it. The new customer becomes an *assigned priority-customer*, the vehicle *assigned with priority* and the previously assigned customer *unassigned*. *Return to Step 1*, and treat the now unassigned customer as a new customer.

The complete DNN policy including all Events A-D can be found in Appendix A.
2.3. **Simulations**

2.3.1. **Methodology**

A number of simulations were conducted to examine the performance of the DNN policy to decrease the mean system time of the customers in the system. To demonstrate its performance, the DNN policy was compared to the FCFS and NN policies as the optimal policy for the DARP is unknown.

The lower bound on an optimal policy for the DTRP, and likewise the DARP, has been defined in [30], however it is not readily tractable. The difference of the DTRP from the DARP is that a repair-person travels to the customer’s location and spends a random amount of service time there, whereas in the DARP a customer is transported from that location to a drop-off location. After the service is over in both problems, the server travels to another customer if there is any waiting for service. Two theorems to obtain two lower-bounds for the optimal mean system time, \( E[T^*] \), one for the light-traffic (\( \lambda \to 0 \)) and the other for the heavy-traffic cases (\( r \to 1, \quad r \equiv \lambda E[S]/N \)) are presented. The difference that in the DARP a service vehicle will be at a different location (drop-off location of the customer) at the end of its service time does not affect the proofs of the theorems, which render them applicable to this problem as well. In particular, Theorem 2 in [30], for the heavy-traffic regime states that for some constant \( \gamma \geq 2/(3\sqrt{6\pi}) \approx 0.266 \),

\[
E[T^*] \geq \gamma^2 \frac{\lambda A}{N^2 v^2 (1 - r)^2} - \frac{E[S](1 - 2r)}{2r}.
\]  

(2.13)

The fact that the optimal policy, let alone the actual value of \( \gamma \), is unknown makes the assessment of the performance of a policy with respect to the optimal policy difficult. As a result, in this Thesis, as has been completed in the literature, the simulations presented herein will compare the relative performance of DNN to alternative policies via an extensive numerical study.

The First-Come-First-Served (FCFS) policy is the easiest policy to implement, according to which customers are served in the order that they arrive in the system. When a vehicle drops-off its customer, it is assigned to the unassigned customer (if there is any) who has been waiting the longest. If multiple vehicles are idle, the newly arrived customer is serviced by a random vehicle.
Under the Nearest Neighbour (NN) policy, when a new customer demand enters the system, they are assigned to the nearest available idle vehicle, if one exists. Otherwise, the new customer becomes an unassigned customer and must wait for service until they become the nearest customer to a vehicle that has become idle. Similarly, upon dropping off a customer, the idle vehicle is assigned to the unassigned customer with the shortest distance from the vehicle’s current location, if one exists. Otherwise, the vehicle remains idle.

In [29], the FCFS and NN policies in the DTRP are compared considering only a single vehicle. Via numerical examples, it is demonstrated that the NN policy outperforms the FCFS policy (and other policies investigated) as \( r \to 1 \), and that the FCFS policy cannot handle the higher arrival rates that the NN policy can handle. As a result, while an initial comparison to the FCFS policy will be included in this section for completeness, all comparisons of the DNN policy will be completed with respect to the NN policy as it is already demonstrated to be better than the FCFS policy. For the numerical simulation examples presented herein, for the FCFS policy, performance at \( \rho = 2\lambda E[S]/N = 0.1, 0.2, 0.4, 0.6, 0.8, 0.9, 0.95 \) will be evaluated and for the NN and DNN policies, since they can handle higher arrival rates than the FCFS policy, in addition to those considered for the FCFS policy, \( \rho = 0.99, 1, 1.05, 1.1, 1.2, 1.3, 1.4, 1.5 \) will be evaluated. For each \( \rho \), four values of \( N \) will be evaluated, \( N = 1, 10, 20, \) and 100.

Four simulation cases will be presented to demonstrate the performance of the DNN policy, as outlined in Table 2.1. First, the DNN, NN, and FCFS policies performance will be analysed via numerical examples for the single and multi-server case under constant arrival rates, herein termed the \textit{base-case scenario}. Subsequently the DNN and NN policies will be evaluated under realistic settings, such as city-like environments where customer arrival rates and distributions vary over the course of the day, termed the \textit{city scenario}. Under these environments, the DNN and NN policies’ relative performance with the addition of anticipatory behaviour and wait-time limits will be compared, along with the benefits of utilizing such methods with any policy.

<table>
<thead>
<tr>
<th>Simulation Cases Studied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation Cases</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>
2.3.2. Simulation Environment

The simulation models were created using the Automod (Version 11.1) software package. In all the simulated experiments, the area of the square service region, $A$, was set to unity. A warm-up period of at most 18,000 time units ensured reaching steady-state operating conditions. Once in steady-state, in each of 20 replications, running the system for 28,800 time units resulted in sufficiently tight Confidence Intervals (CIs) around the mean system time estimates. The largest 95% CI half-width to mean system time ratios were obtained when $N = 1$ at the highest arrival rate considered. This ratio was 3.5% for the NN and DNN policies, and 12.2% for the FCFS policy. At other arrival rates, when $N = 1$, the ratio was less than 1%. For $N > 1$, 95% CI half-width to mean system time ratios did not exceed 1% for any arrival rate, and decreased with increasing numbers of vehicles.

In the simulations, the customer arrival process, and the pick-up and drop-off locations of the customers were synchronized by using common random numbers in corresponding replications of alternative policies in order to reduce the variation in simulation outputs, and so that alternatives could be compared more accurately. Simulation models were run on a Windows-based computer with a 2.8 GHz CPU and 2GB RAM, with the DNN policy simulation taking longer to run than the FCFS and NN policies. The computation times increased with the arrival rate and $N$. When $N=100$ and $\rho = 1.5$, it took roughly 4.5 minutes to run each of the 20 replications of the DNN simulation model in the base-case scenario, and 12.5 minutes for the cases to be discussed in Section 2.3.6.

In the simulation, the clock advances to the earliest of the events discussed in Section 2.2, and the updates the locations of vehicles at the event. Service vehicles (taxis) once assigned to a customer or vehicle station immediately begin moving along the shortest Euclidean distance at velocity $v$ between their current and destination location, where $v = 1$. Unless otherwise specified, idle vehicles remain at their drop-off location until they are given an assignment. Re-evaluation of pairings occurs only when an event is triggered, and is not re-evaluated between event triggers.

2.3.3. Case 1: Analysis of DNN, NN, and FCFS in Base-case scenario

The analysis of the base-case scenario consists of comparing the behaviours of the three policies with customers arriving at a constant arrival rate with uniform and independent distributions of
customer pick-up and drop-off locations. First, the FCFS and NN policies will be analysed in Section 2.2.3.1, followed by the NN and DNN policies in Section 2.2.3.2.

2.3.3.1. **Base Case Analysis of the FCFS and NN Policies**

The relative performances of the NN and FCFS policies is presented in Fig. 2.1 for the arrival rates and numbers of vehicles discussed above: the x-axis shows $\lambda/N$, and the y-axis shows the percent improvement of the NN policy over the FCFS policy, defined as $(E[T_{FCFS}] - E[T_{NN}])/E[T_{NN}] \%$ when both policies have the same $\lambda$ and $N$. In all the examples, the NN policy yields lower mean system times. The minimum, median and maximum percent improvements of the NN policy as compared to the FCFS policy are given in Table 2.2. For $N=1$, the improvement monotonically increases with higher arrival rates, while for $N > 1$, the improvement first declines up until $\lambda/N = 0.769$, after which the improvement starts increasing, mainly because the FCFS policy approaches the critical $\rho$, and its mean system time explodes. Moreover, the NN policy performs better as $N$ increases until $\lambda/N = 0.769$, which can be seen in Fig. 2.2 (a)-(d).

![Image](image.png)

**Fig. 2.1.** Mean System Time Percent Improvement of NN over FCFS under the base-case scenario.
Table 2.2. Percent Improvement of the NN policy over the FCFS policy.

<table>
<thead>
<tr>
<th>N</th>
<th>Min (%)</th>
<th>Median (%)</th>
<th>Max (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.16</td>
<td>15.69</td>
<td>343</td>
</tr>
<tr>
<td>10</td>
<td>29</td>
<td>40.01</td>
<td>108</td>
</tr>
<tr>
<td>20</td>
<td>41</td>
<td>49.22</td>
<td>82</td>
</tr>
<tr>
<td>100</td>
<td>68</td>
<td>72.25</td>
<td>75</td>
</tr>
</tbody>
</table>

Fig. 2.2. (a) Mean System Time comparisons for $N = 1$.

Fig. 2.2. (b) Mean System Time comparisons for $N = 10$.

Fig. 2.2. (c) Mean System Time comparisons for $N = 20$.

Fig. 2.2. (d) Mean System Time comparisons for $N = 100$.

In order to study the different behaviour of the improvement curves for FCFS and NN when $N = 1$ and $N > 1$, the mean travel time of vehicles was estimated between the last dropped-off customer and the next picked-up customer. Since the service times are stochastically the same in any policy, the better performance of the NN policy can only be attributed to lower mean travel times between two consecutive customers served. Let this travel time be denoted by the random variable $D$. Under the FCFS policy, $D$ has the same distribution as $S$ since the vehicle’s
last customer’s drop-off location and the next customer’s pick-up location are generated independently of each other and uniformly distributed. Consequently, $D$ has the same first two moments also given by Eqs. (2.2) and (2.3). Therefore, when $v = 1$, under the FCFS policy, $E[D] \approx 0.52$ and is invariant of $\lambda/N$. However, under the NN policy, Fig. 2.3 shows how $E[D]$ changes with $\lambda/N$.

![Fig. 2.3. Mean travel time between two consecutively served customers under the NN policy.](image)

As can be seen in Fig. 2.3, the larger the size of the fleet of vehicles, the smaller $E[D]$ is, which explains why the NN policy performs better compared to the FCFS policy with more vehicles. However, while $E[D]$ decreases monotonically for $N = 1$, which explains the monotonical increase in the NN policy’s relative performance in Fig. 2.1 for a single vehicle, for $N > 1$ it increases up until a $\lambda/N$ value, different for each $N$, after which it tends to decrease. The left-most vertical dashed line in Fig. 2.3 shows $\lambda/N = 0.913$ corresponding to $\rho = 0.95$, the highest arrival rates considered for the FCFS policy in Fig. 2.1, and as can be seen in this figure $E[D]$ does not decrease for any $N > 1$ up until this point. As a result, the slight decline in the performance of the NN policy until $\lambda/N = 0.769$ is due to $E[D]$ increasing for NN while $E[D]$ for FCFS remains constant. However, after this point the FCFS policy approaches the
critical load and its mean system time increases so much that even though $E[D]$ continues to increase for a while for the NN policy, its relative performance starts improving exponentially.

The right-most vertical dashed line in Fig. 2.3 is the point after which $E[D]$ becomes the same for all $N > 1$ and they tend to 0. A potential explanation for this behaviour of $E[D]$ for $N > 1$, is that with low arrival rates there are typically several idle vehicles when a new customer arrives. Since the nearest vehicle is selected for service, $E[D]$ will be less than 0.52. As $\lambda$ increases there are fewer idle vehicles on average and, thus, it increases. Once $\lambda$ becomes large enough, the controlling factor shifts from the number of idle vehicles to the number of unassigned customers in the system. At this point, each newly available vehicle has multiple customers to choose from, and hence, $E[D]$ once again begins to decrease. It should also be noted that as $N$ increases, the critical value of $\lambda/N$ also increases.

### 2.3.3.2. Base Case Analysis of the NN and DNN Policies

The relative performance of the DNN policy is compared herein with the NN policy. In Fig. 2.4, the $x$-axis shows $\lambda/N$, whereas the $y$-axis shows the percent improvement of the DNN policy over the NN policy defined as $(E[T_{NN}] - E[T_{DNN}])/E[T_{DNN}] \%$ when both policies have the same $\lambda$ and $N$. In all the examples, the DNN policy yields lower mean system times except when $N=1$, for $\lambda/N = 1.442308$. For this arrival rate the 95% CI is [-0.69%, 0.72%], and since it contains 0 both policies are deemed to be statistically not differentiable. The minimum, median and maximum percent improvements of the DNN policy as compared to the NN policy are given in Table 2.3.

<table>
<thead>
<tr>
<th>$N$</th>
<th>Min (%)</th>
<th>Median (%)</th>
<th>Max (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.02</td>
<td>3.09</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>0.22</td>
<td>13.22</td>
<td>19.64</td>
</tr>
<tr>
<td>20</td>
<td>0.11</td>
<td>13.21</td>
<td>23.94</td>
</tr>
<tr>
<td>100</td>
<td>0.025</td>
<td>2.04</td>
<td>25.90</td>
</tr>
</tbody>
</table>

As can be seen in Fig. 2.4, for each $N$ the improvement curves are hill-shaped. As $N$ increases, higher $\lambda/N$ values are needed for the improvement curve to start inclining sharply, and similarly for the performance to subsequently peak and decline. The $E[D]$ curves for DNN are similar to those in Fig. 2.3 for NN, but the values are less than or equal to those of the NN
policy for the same $N$ and $\lambda/N$ values. When $\lambda/N$ is small or large, the $E[D]$ for both policies becomes almost the same, whereas for intermediate values, it is strictly smaller under the DNN policy. This trend can be seen in the shape of the improvement curves in Fig. 2.4, where for low and high $\lambda/N$ values the percent improvement of DNN over NN is minimal, and for intermediate values is great. This is because, at low arrival rates there are many idle vehicles when a new customer arrives and, as a result, it is extremely unlikely for vehicle re-assignments to occur. As a result, at these arrival rates there is practically no difference in the behaviour of the two policies.

![Fig. 2.4. Percent Improvement of DNN over NN in Base-case scenario.](image)

However, as the arrival rate increases the number of vehicle re-assignments also increases, as with more customers entering the system, the probability of a customer arriving closer to a vehicle than its currently assigned customer is greater. The greater the arrival rate, the more re-assignments occur in the DNN policy and the average $E[D]$ decreases more than the average under the NN policy, and a difference between the two policies can be seen. At some point, the improvement reaches a maximum and, then, begins to decline because of the impact of a second factor. With higher arrival rates, there are many unassigned customers awaiting service. It is likely that any newly idle vehicle will be located very close to a currently unassigned customer, and, as a result, the chances of a beneficial vehicle re-assignment occurring from a new customer
arriving closer than this already assigned customer becomes minimal. As the traffic intensity continues to increase, fewer and fewer re-assignments occur and the improvement declines to the point where the performance of both policies becomes similar.

2.3.4. Case 2: Analysis of DNN and NN in City-like Environments

In order to evaluate the performance of the DNN policy in realistic environments for taxi-cab dispatching, the policies relative performance to NN is compared under a city scenario. First, a description of the scenario used in these experiments is provided in Section 2.3.4.1, and subsequently a discussion on the performance of both policies and their relative performance under these conditions is presented in Section 2.3.4.2.

2.3.4.1. City Scenario

To emulate a city-like environment, the simulations considered a cyclic 24-hour time pattern with arrival rates and customer demand distributions fluctuating over the course of the day.

The arrival rates are varied over the course of the day in three distinct phases, normal hours, quiet hours, and rush hours. The intervals from 6:00am to 7:00am, 9:00 am to 5:00 pm and 7:00 pm to 12:00 am are considered normal hours during which customer demands arrive according to \( \lambda/N \), corresponding to \( \lambda/N \) in the base-case scenario. The interval from 12:00 am to 6:00 am is considered quiet hours, during which the customer arrival rate is set to half of the arrival rate of the normal hours, \( \lambda/2N \), to simulate few customers needing service during sleeping hours. Finally, during the intervals from 7:00am to 9:00am and 5:00pm to 7:00pm the arrival rate is doubled, \( 2\lambda/N \), to simulate rush hours as residents of the city travel to and from work.

In order to further model city-like transportation patterns, where residents live in the suburbs and work downtown, customer pick-up and drop-off locations are modified to emulate this pattern. During the intervals of 6:00 am to 10:00 am and 4:00 pm to 8:00 pm, the square service area, \( \mathcal{A} \) was considered as divided into two halves vertically, with the halves representing the downtown of a city and its suburbs, respectively. The former interval is considered as morning hours, during which \( D_h = 50\% \) of the customers have pick-up locations in the suburbs and drop-off locations downtown, and the remaining \( D_u = 50\% \) of the customers have pick-up and drop-off locations that are independently and uniformly distributed as they
were in the base-case scenario. Similar to morning rush hours, the interval from 4:00 pm to 8:00 pm is considered *evening hours*, during which \( D_h = 50\% \) of the customers have pick-up locations downtown and drop-off locations in the suburbs. The remaining \( D_u = 50\% \) of the customers have pick-up and drop-off locations that are independently and uniformly distributed as they were in the base-case scenario. During the remaining time intervals 12:00am to 6:00 am, 10:00am to 4:00pm, and 8:00pm to 12:00am, *standard hours* occur and all customers have pick-up and drop-off locations that are independently and uniformly distributed as they were in the base-case scenario.

To illustrate a 24-hour day in the city scenario defined herein, Table 2.4 is presented to show the fluctuation in arrival rates and customer arrival distributions at each hour.

### Table 2.4. Hourly Changes in Customer Arrival Rates and Demand Distributions in the City Scenario.

<table>
<thead>
<tr>
<th>time</th>
<th>12am</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda/N )</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( D_u )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( D_h )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Column values for \( \lambda/N \) are multipliers of \( \lambda/N \), and column values for \( D_u \) and \( D_h \) are percentages of the total number of customer demands in that time period.

#### 2.3.4.2. Performance of DNN and NN in the City Scenario

To properly evaluate the relative performance of DNN and NN in city-like conditions, the performance of each policy in the city scenario, as compared to the base-case scenario, must first be analysed. The arrival rate of customer demands during normal hours is equal to the arrival rates \( \lambda \) considered in the base-case scenarios. Given that the arrival rate is \( \lambda/2 \) for six quiet hours, \( 2\lambda \) for four rush hours each day, and \( \lambda \) for the remaining 14 hours, the mean arrival rate is \( \lambda_c = 25\lambda/24 \), as compared to \( \lambda \) in the base-case scenario. As a result, there are more customer demands to be addressed in the city scenario as compared to the base-case scenario. In addition to having slightly higher overall arrival rates, Eqs. (2.2) and (2.3) are no longer valid as a result of half the customers being routed from one half of the unit square to the other for 8 hours every day. From simulation results, it is determined that the mean time a customer rides in a vehicle increases from \( E[S] = 0.52 \), for the base-case scenario, to \( E[S_c] = 0.55 \) for the city scenario. This issue also increases the mean driving time in-between customers, however its mean value
depends on the customer arrival rate. As a result, the mean system time under both policies in the
city scenario increased with respect to the base-case scenario. Table 2.5 lists the minimum,
median and maximum of these increases for both policies, with these increases demonstrated
graphically for all $\lambda$ investigated in Fig. 2.5 for the NN policy and Fig. 2.6 for the DNN policy.

| Table 2.5. Mean System Time increase in City scenario as compared to the Base-case scenario. |
|---|---|---|
| N | Min (%) | Median (%) | Max (%) |
| NN Policy  | 1 | 4.67 | 35.04 | 120.74 |
|  | 10 | 4.74 | 33.05 | 56.66 |
|  | 20 | 4.83 | 34.78 | 61.19 |
|  | 100 | 4.95 | 42.99 | 68.73 |
| DNN Policy  | 1 | 4.56 | 35.99 | 120.55 |
|  | 10 | 4.74 | 38.34 | 58.22 |
|  | 20 | 4.81 | 38.90 | 63.66 |
|  | 100 | 4.96 | 37.25 | 72.39 |

![Fig. 2.5. Mean System Time Increase from City over Base Case for the NN Policy.](image)

![Fig. 2.6. Mean System Time Increase from City over Base Case for the DNN Policy.](image)

Given that the DNN policy loses its superiority over the NN policy with higher offered
load, namely, $\lambda E[S]$, lower improvements in the relative performance of the DNN policy due to
higher $\lambda E[S_r]$ were expected as compared to those in Fig. 2.4. In Fig. 2.7, the $x$-axis shows
$\lambda/N$ with $\lambda$ as the arrival rate during normal hours, and the $y$-axis shows the percent
improvement of the DNN policy over the NN policy defined as

\[
\left(\frac{E[T_{NN}]-E[T_{DNN}]}{E[T_{DNN}]}\right)\% 
\]

when both policies have the same number, $\lambda$ and $N$. As expected, the percentage improvements in Fig. 2.7 are much lower than those in Fig. 2.4,
however, the same trend in relative performance is visible. As with the base-case scenario, when $N$ and $\lambda$ are greater, the difference between the two policies increases further, whereas as $\lambda_c E[S_c] \to 1$ or $\lambda_c E[S_c] \to 0$, the two policies tend to be identical. The minimum, median and maximum percent improvements of the DNN policy as compared to the NN policy are given in Table 2.6.

**Table 2.6.** Percent Improvement of the DNN policy over the NN policy in the City Scenario.

<table>
<thead>
<tr>
<th>N</th>
<th>Min (%)</th>
<th>Median (%)</th>
<th>Max (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2</td>
<td>2.72</td>
<td>3.5</td>
</tr>
<tr>
<td>10</td>
<td>0.22</td>
<td>6.96</td>
<td>10.2</td>
</tr>
<tr>
<td>20</td>
<td>0.12</td>
<td>6.52</td>
<td>10.4</td>
</tr>
<tr>
<td>100</td>
<td>0.02</td>
<td>9.69</td>
<td>13.07</td>
</tr>
</tbody>
</table>

**Fig. 2.7.** Percent Improvement of the DNN policy over the NN policy in the City Scenario.

### 2.3.5. Case 3: Analysis of DNN and NN in City-like Environments with Anticipatory Behaviour

In all previous simulation cases, the limiting assumption that an idle vehicle remains at the drop-off location of its last customer if it has not been assigned to a new customer is made. However,
as discussed in Section 2.2.2, sending these idle vehicles to wait stations strategically located in the area would minimize the mean driving time to new customers and, as a result, the individual wait times of the customers. In order to evaluate the performance of the DNN and NN policies, and their relative performance under these conditions, a method of anticipatory behaviour is employed under the city scenario to determine the benefits of routing these vehicles to wait stations in anticipation of customer arrival. First, a description of the determination of vehicle station locations for these simulations is provided in Section 2.3.5.1, and subsequently a discussion on the performance of both policies under these conditions is presented in Section 2.3.5.2.

2.3.5.1. Placement of Vehicle Stations for Anticipatory Behaviour

As discussed previously in Section 2.2.2.1, the issue of the optimal number of vehicle stations and how to choose their optimal locations is not addressed in this thesis. It is only assumed that the number of vehicle stations needs to be approximately equal to the number of vehicles, $N$, (to decrease the expected distance to reach a customer demand the most) and that their distribution across the area needs to be dynamic and representative of the variations in arrival patterns over the course of the day.

For the simulations discussed herein, the vehicle stations are distributed over the service region according to the number of vehicles in the fleet, $N$. Due to the square-shaped region, $A$, each station is located at the intersection of fictitious vertical and horizontal lines that segment the region. In Table 2.7 the numbers of stations used in these simulations are given for the different numbers of vehicles considered, $N$. The number of vertical ($N_V$) and horizontal ($N_H$) lines are also given (in parenthesis) in this table. When $N_V$ and $N_H$ are not equal to the square root of the number of vehicle stations, they were chosen to be divisors of the number of locations that are closest to one another.

<table>
<thead>
<tr>
<th>$N$</th>
<th>Low Demand</th>
<th>Uniform Demand</th>
<th>High Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 (2,1)</td>
<td>1 (1,1)</td>
<td>2 (2,1)</td>
</tr>
<tr>
<td>10</td>
<td>6 (2,3)</td>
<td>9 (3,3)</td>
<td>16 (4,4)</td>
</tr>
<tr>
<td>20</td>
<td>12 (4,3)</td>
<td>20 (4,5)</td>
<td>30 (6,5)</td>
</tr>
<tr>
<td>100</td>
<td>48 (6,8)</td>
<td>100 (10,10)</td>
<td>144 (12,12)</td>
</tr>
</tbody>
</table>
In order to determine the coordinates of the vehicle stations, let us assume that the \((x, y)\) coordinates of the lower left corner of the service area are at the origin, (0, 0). The vertical lines, \(x\), and horizontal lines, \(y\), are located at:

\[
x = \frac{1}{2}N_V + \frac{i}{N_V}, \quad i = 0, \ldots, N_V - 1, \quad \text{and} \quad (2.14)
\]
\[
y = \frac{1}{2}N_H + \frac{i}{N_H}, \quad i = 0, \ldots, N_H - 1. \quad (2.15)
\]

Under the city scenario, the distribution of customer demands varies over the course of the day. As such, three types of distributions of customer demands occur to emulate this in the simulation: standard hours, morning hours, and evening hours as discussed in Section 2.3.4.1. During standard hours, the number of cab vehicle stations listed for uniform demand in Table 2.7 are located across \(\mathcal{A}\). In such cases, it does not matter which one is \(N_V\) or \(N_H\), since the service area is a square and there is uniform demand. During morning and evening hours one half of the area is the high-demand area and the other half low demand, where a vertical line at \(x = 0.5\) represents the boundary between downtown and the suburbs. During these times, half the numbers of vehicle stations listed under the low-demand area and half the numbers under the high-demand area are located in their respective halves, as the numbers provided in Table 2.7 are for a full unit square. The position of these vehicle stations is determined by which half of the vehicle stations fall on the appropriate side of \(x = 0.5\), as determined by Eqs. (2.14) and (2.15).

To demonstrate this concept, an example is provided in Figs. 2.8 and 2.9 for the \(N=10\) case. In both of the examples, the bold vertical line at \(x = 0.5\) represents the boundary between downtown and the suburbs, where downtown is located on the left side of this line, and the suburbs on its right. The distribution of vehicle stations for standard hours is provided in Fig. 2.8. For \(N = 10\), under uniform demand, 9 stations are placed uniformly across \(\mathcal{A}\), with \(N_H = N_V = 3\). The distribution of vehicle stations for morning hours is provided in Fig. 2.9, with the distribution for evening hours being the exact mirror distribution about \(x = 0.5\). For \(N = 10\), for the low-demand area \(6/2 = 3\) vehicle stations are located on its respective side of \(x = 0.5\), with \(N_V = 2\), and \(N_H = 3\). Similarly, for the high-demand area, \(16/2 = 8\) vehicle stations are located on its respective side, with \(N_V = 4\), and \(N_H = 4\).
Fig. 2.8. Locations of vehicle stations for standard hours when $N = 10$.

Fig. 2.9. Locations of vehicle stations during morning hours when $N = 10$.

To clarify how vehicles are assigned to these vehicle stations in simulation, an example to demonstrate the implementation of the anticipatory behaviour methodology in Section 2.2.1 for morning hours is discussed. $D_h = 50\%$ of all customer demands originate in the area of high
demand (suburbs) and are dropped off in the area of low demand (downtown). The remaining $D_u = 50\%$, arrive equally likely in either the suburbs or downtown, and get dropped off equally in the suburbs or downtown. Consequently, if idle vehicles would remain wherever they dropped off their last customers, 75% of the fleet would end up in downtown and 25% in the suburbs.

It is, thus, logical that the dispatching policy at hand would attempt to match the 75% arrival rate from the suburbs by having 75% of idle vehicles located in the suburbs and not remaining downtown. This is achieved by sending 2/3 of the idle vehicles that complete their trips downtown back to the suburbs (i.e., 50% of the fleet that is downtown) to complement the 25% of the fleet which is already in the suburbs. The same strategy is employed for evening hours, with the terms downtown and suburbs reversed.

2.3.5.2.  Analysis of the NN and DNN policies under the City Scenario with Anticipatory Behaviour

To properly evaluate the relative performance of the DNN and NN policies with anticipatory behaviour in the city scenario, the performance of each policy under these conditions, as compared to the city scenario, must first be analysed. In order to complete such an evaluation, the NN policy needs to be modified to include anticipatory behaviour similar to that of DNN.

The way in which vehicles are assigned to vehicle stations is the same in both anticipatory policies. The difference between these anticipatory policies is that, where in DNN assigning a $V_{S\text{assigned}}$ vehicle to a new customer demand would trigger a re-evaluation of the entire set of vehicle assignments, in the NN policy, this re-assignment from $V_{S\text{assigned}}$ to the new customer demand would not change the assignments of any other vehicles in the fleet.

The minimum, median and maximum decreases in mean system time due to anticipatory NN/DNN dispatching compared to the city scenario with no anticipatory behaviour are given in Table 2.8, with these decreases demonstrated graphically for all $\lambda$ investigated in Fig. 2.10 for the NN policy and Fig. 2.11 for the DNN policy. In each case, the decrease in mean system time due to anticipatory behaviour diminishes as the arrival rate increases. This is expected since with higher offered loads, there are some unassigned customers present and vehicles do not become idle after dropping-off their customers. As a result, these vehicles are not directed to the vehicle stations, but instead to these customers; thus, the benefit of anticipatory behaviour diminishes.
Table 2.8. Mean System Time decrease in the City Scenario with Anticipatory Behaviour.

<table>
<thead>
<tr>
<th>N</th>
<th>Min (%)</th>
<th>Median (%)</th>
<th>Max (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN Policy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>10.23</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>2.98</td>
<td>9.38</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>3.47</td>
<td>9.47</td>
</tr>
<tr>
<td>100</td>
<td>0.25</td>
<td>4.93</td>
<td>9.86</td>
</tr>
<tr>
<td>DNN Policy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>9.85</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>3.46</td>
<td>9.06</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>4.66</td>
<td>8.94</td>
</tr>
<tr>
<td>100</td>
<td>0.72</td>
<td>5.18</td>
<td>8.77</td>
</tr>
</tbody>
</table>

Fig. 2.10. Mean System Time Decrease with anticipatory behaviour in NN.

Fig. 2.11. Mean System Time Decrease with anticipatory behaviour in DNN.

For $N = 1$, the maximum decrease with both NN and DNN policies is observed at $\lambda/N = 0.096$, which was the smallest value considered. At the $\alpha = 5\%$ significance level, no statistical difference was evident with either policy after $\lambda/N = 0.913$, as with this higher load, the vehicle cannot be anticipatorily routed as frequently, if at all. However, for $N = 10$ the maximum decrease is attained at $\lambda/N = 0.385$ with both policies, for $N = 20$ at $\lambda/N = 0.385$ with NN, and $\lambda/N = 0.577$ with DNN, and for $N = 100$ at $\lambda/N = 0.577$ with both policies, the third and fourth smallest normal hour arrival rates considered. The benefit of anticipatory behaviour becomes more pronounced when there is a fleet of vehicles instead of a single vehicle, as is evident by the trends in the mean system time decrease curves in Figs. 2.10 and 2.11.
In Fig. 2.12, the relative performances of the NN and DNN policies are compared. The minimum, median and maximum percent improvements of the DNN policy as compared to the NN policy are given in Table 2.9. For intermediate $\lambda$, for $N > 1$, the relative performance increases by 1 to 2% as compared to the city scenario. However, in general, the relative performances are similar to the results summarized in Figure 2.7. As a result, the decrease in mean system time under both policies due to anticipatory dispatching is similar, which preserves the relative superiority of the DNN policy over the NN policy.

### Table 2.9. Percent Improvement of DNN over NN in the City Scenario with Anticipatory Behaviour.

<table>
<thead>
<tr>
<th></th>
<th>Min (%)</th>
<th>Median (%)</th>
<th>Max (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.96</td>
<td>3.8</td>
</tr>
<tr>
<td>10</td>
<td>0.2</td>
<td>7.39</td>
<td>11.6</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>6.73</td>
<td>12.2</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>9.90</td>
<td>14.3</td>
</tr>
</tbody>
</table>

**Fig. 2.12.** Percent Improvement of DNN over NN in the City Scenario with Anticipatory Behaviour.
2.3.6. Case 4: Analysis of DNN and NN in the City Scenario with Anticipatory Behaviour and Capping of Customer Wait Times

Although the DNN policy decreases the mean system time, it does not assure that some customers would not experience excessive wait times. Such long waiting times were observed during simulation, especially, since the policy favours swapping longer trips for shorter trips. In order to examine the severity of the issue, the simulation output of the case discussed in Section 2.3.5 was analysed for each $\lambda$ and $N$, and the longest wait times from each of the 20 replications was determined and compared with the mean wait time estimate ($\bar{W}_{N,\lambda}$, with $N$ vehicles and $\lambda$ as the arrival rate during normal hours). In Table 2.10, the minimum, median and maximum values obtained from this analysis are presented.

<table>
<thead>
<tr>
<th>N</th>
<th>Min (%)</th>
<th>Median (%)</th>
<th>Max (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>629</td>
<td>1,811</td>
<td>3,474</td>
</tr>
<tr>
<td>10</td>
<td>684</td>
<td>1,767</td>
<td>4,134</td>
</tr>
<tr>
<td>20</td>
<td>956</td>
<td>1,842</td>
<td>2,980</td>
</tr>
<tr>
<td>100</td>
<td>1,062</td>
<td>2,083</td>
<td>3,589</td>
</tr>
</tbody>
</table>

As can be seen from this analysis, the longest wait times were significantly greater than the mean wait times for all $\lambda$ and $N$. As discussed in Section 2.2.3, the optimality of the actual time limit on wait times employed is beyond the scope of this Thesis. However, to minimize these longest wait times, after estimating the standard deviation ($\delta_{N,\lambda}$) and the mean wait times ($\bar{W}_{N,\lambda}$) from the simulation output, $\bar{W}_{N,\lambda} + 6\delta_{N,\lambda}$ was selected as the limit. This limit was selected as it was not so low that the majority of customers were past it, and not so high that customers did not reach it.

Once a customer’s wait time reaches this limit, as discussed in Section 2.2.3, for DNN it becomes a high-priority customer and is immediately given an absolute assignment with the closest assigned vehicle. As a result, a re-evaluation of the DNN model is performed at the occurrence of the event and the assignments of other vehicles may change. To analyse the benefits of this capping policy, and for fair comparison in terms of the policies relative performance, the NN policy was modified as well, such that when a high-priority customer is
noted, as soon as a vehicle becomes idle, it is routed immediately to the closest high-priority customer with an absolute assignment.

After the capping policy was implemented, the longest wait times from 20 replications were recorded and compared with those obtained when no limit was used. In Table 2.11, the reductions in longest wait times attained after implementing the capping policy are given for both policies at each \( N \). It is difficult to outline a general trend, but it may be noted that more reduction occurs under the DNN policy with \( N > 1 \). This may be due to the possibility of redirecting a previously assigned vehicle to a high-priority customer as soon as it is prioritized, in contrast to the NN policy, where the high-priority customer has to wait until a vehicle drops off its customer and there are no other high-priority customers closer to this vehicle. Additionally, it is observed that the reduction in maximum wait times due to capping tends to decrease as \( N \) increases.

<table>
<thead>
<tr>
<th>( N )</th>
<th>Min (%)</th>
<th>Median (%)</th>
<th>Max (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>1</td>
<td>0</td>
<td>113.63</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0</td>
<td>49.28</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0</td>
<td>37.16</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0</td>
<td>13.42</td>
</tr>
<tr>
<td>DNN</td>
<td>1</td>
<td>0</td>
<td>91.98</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>5.74</td>
<td>58.88</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>18.22</td>
<td>52.58</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>11.86</td>
<td>39.7</td>
</tr>
</tbody>
</table>

However, while imposing a wait-time limit may decrease the longest wait times, its effect on the mean system time must be analysed. For \( N = 1 \), in about half of the \( \lambda \)s analysed, the increase in mean system times were statistically significant at the \( \alpha = 5\% \) significance level; whereas, for \( N > 1 \), the difference in mean system times were not significant most of the time for both policies. When the change in mean system times was statistically significant, the highest increases are presented in Table 2.12 for each \( N \). As a result, it is concluded that at these wait-time limits, with no or little increase in mean system times, implementing wait-time limits reduces the longest wait times significantly.
Chapter 2: Dynamic Nearest Neighbour Policy

Table 2.12. Greatest Increases in Mean System Time as a result of using Wait-Time Limits.

<table>
<thead>
<tr>
<th></th>
<th>N = 1</th>
<th>N = 10</th>
<th>N = 20</th>
<th>N = 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN Policy</td>
<td>5.59%</td>
<td>0.67%</td>
<td>0.588%</td>
<td>0.00%</td>
</tr>
<tr>
<td>((\lambda/N = 1.346154))</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNN Policy</td>
<td>4.13%</td>
<td>0.73%</td>
<td>0.48%</td>
<td>0.07%</td>
</tr>
<tr>
<td>((\lambda/N = 1.346154))</td>
<td>((\lambda/N = 0.951923))</td>
<td>((\lambda/N = 0.961538))</td>
<td>((\lambda/N = 0.576923))</td>
<td></td>
</tr>
</tbody>
</table>

Given that the mean system times hardly increased, or were statistically not different from the estimates found in Section 2.3.5, the relative performances of the two policies with the wait-time limits is expected to be similar to those in Fig. 2.12. The relative performances presented in Fig. 2.13 for this case attest to the similarity. The minimum, median and maximum percent improvements of the DNN policy as compared to the NN policy are given in Table 2.13. As a result, the use of wait-time limits decreases the longest wait time under both policies while increasing the mean system time minimally. Thus, the relative superiority of the DNN policy over the NN policy is preserved.

Fig. 2.13. Percent Improvement of DNN over NN in the City Scenario with Anticipatory Behaviour and Capping.
Table 2.13. Percent Improvement of DNN over NN in the City Scenario with Anticipatory Behaviour and Capping.

<table>
<thead>
<tr>
<th>N</th>
<th>Min (%)</th>
<th>Median (%)</th>
<th>Max (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.36</td>
<td>4.35</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>7.48</td>
<td>11.52</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>6.61</td>
<td>11.85</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>9.75</td>
<td>14.27</td>
</tr>
</tbody>
</table>

2.4. Summary

The proposed Dynamic Nearest Neighbour (DNN) policy presented in this Chapter, is capable of determining vehicle-customer pairings as demands become available in an on-line manner. Through the use of vehicle re-routing, and the re-evaluation of all assignments as new information becomes available, the DNN policy has the ability to reduce individual customer system times by re-assigning a vehicle to a new demand closer to it than its previous assignment. The incorporation of anticipatory behaviour through vehicle wait stations, and wait-time limits further decrease the policies mean system time. As can be seen from the simulation examples provided, the DNN Policy has been successfully implemented, and demonstrated to assign and re-assign vehicles for the DARP with lower mean system time than both the FCFS and NN policies. The policy’s ability to maintain this relative performance was proven to be maintained regardless of vehicle number, N, arrival rate, λ, and dynamic changes in the arrival rate and locations of customers under city-like conditions. The benefits of the use of anticipatory behaviour and wait-time limits in conjunction with dispatching policies were demonstrated to decrease the mean system time of customer demands for both the NN and DNN policies, while maintaining their relative performance.
3. Dynamic Re-Pairing Methodology

For systems in which tasks are distributed across multiple autonomous agents (pursuers), the optimal coordination of the agents’ actions is necessary to ensure the successful completion of the common objective. For the Multi-Target Interception Problem (MTIP) addressed herein, these tasks are the individual interceptions of a target by a pursuer. These agents utilize real-time information about the dynamic environments in which they operate to make decisions about task allocation and motion-planning on-line in real-time. In this context, the second task-allocation problem addressed in this Thesis consists of time-optimally re-allocating dynamic numbers of pursuers to targets of interest based on changes in the \( a \ priori \) unknown numbers of pursuers and highly-maneuverable targets.

This Chapter presents the Dynamic Re-Pairing Methodology (DRPM) that can be used to pair and navigate pursuers to targets on-line for time-optimal interception. Section 3.1 constructs the problem definition of the task-allocation optimization problem. Section 3.2 presents the proposed DRPM, and its four customizable modules: Task-Allocation, Navigation, Image-Acquisition and Data Management. Subsequently, implementation options for the DRPM are discussed in
Section 3.4. Finally, simulation results demonstrating the DRPMs ability to re-allocate pursuers to targets for minimum total interception time are presented.

3.1. Problem Definition

The objective of this work is the development of a novel generic methodology for the interception of a set of mobile targets by a dynamic team of robotic pursuers in minimum (total) time. Targets are assumed to appear in and disappear from the environment at random locations in space and time. Similarly, the team of pursuers may be augmented with additional pursuers, or lose some during the pursuit. The dynamic optimal allocation of tasks is centralized, and pursuers execute their assigned tasks in an autonomous manner.

The task-allocation problem at hand is one of combinatorial nature – real-time, continuous re-pairing of pursuers and targets. In this context, let us first define an \((m \times n)\) binary matrix \(X\) that represents the pursuer-target pairings, where:

\[
x_{ij} = \begin{cases} 1 & \text{if Pursuer } i \text{ is paired with Target } j, \\ 0 & \text{otherwise}; \end{cases} \quad (3.1)
\]

subject to \(\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} = \min (m, n), \quad (3.2)\)

\[
\sum_{j=1}^{n} x_{ij} = \begin{cases} 1 & \text{if Pursuer } i \text{ is paired, } \forall i, \\ 0 & \text{otherwise}; \end{cases} \quad (3.3)
\]

and \(\sum_{i=1}^{m} x_{ij} = \begin{cases} 1 & \text{if Target } j \text{ is paired, } \forall j, \\ 0 & \text{otherwise}; \end{cases} \quad (3.4)
\]

Above, \(m\) is the number of available pursuers and \(n\) is the number of not-intercepted targets at the time of the optimization. Each pursuer is paired with a unique target at any time instant. All pairings are periodically re-evaluated based on the latest state of the dynamic environment. The pursuer-target pairings matrix \(X\) is resized according to the changes in \(m\) and \(n\) so that only those pursuers and not-intercepted targets in the environment are considered.

In order to determine the instantaneous optimal pairings, let \(F\) represent the \((m \times n)\) matrix of interception times for each pursuer-target pairing, where:

\[
f_{ij} = t_{ij}, \quad \forall i, j. \quad (3.5)
\]
Above, \( t_{ij} \) represents the interception time of Target \( j \) by Pursuer \( i \). Furthermore, let \( G(X) \) represent the interception times for the pairings selected in \( X \) where:

\[
g_{ij} = x_{ij} \times f_{ij}, \quad \forall i, j, \tag{3.6}
\]

such that only the interception times for the pairings chosen in \( X \), by the search engine, are considered during the optimization. Total interception time for a group of pursuers is assumed to be equal to the maximum of the estimated individual interception times for all the pairs at any given time instant, which needs to be minimized:

\[
\min Z = \min \max \{ G(X) \}. \tag{3.7}
\]

Using a rolling horizon, the above objective can be extended such that one is able to look ahead past the first interception to optimize subsequent interceptions. Namely, in a multi-stage horizon, each pursuer can be paired with a string of queued targets, where the depth of the horizon represents the number of targets the pursuer will intercept in a sequential manner. Consequently, the binary \( X \) matrix is extended to represent the horizon stage in which each target is intercepted:

\[
x_{ijk} = \begin{cases} 
  1 & \text{if Pursuer } i \text{ is paired with Target } j \\
  0 & \text{at Horizon Stage } k,
\end{cases} \tag{3.8}
\]

subject to

\[
\sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{s} x_{ijk} = \min \left( (m \times s), n \right), \tag{3.9}
\]

\[
\sum_{j=1}^{n} x_{ijk} = \begin{cases} 
  1 & \text{if Pursuer } i \text{ is paired,} \\
  0 & \text{otherwise};
\end{cases} \quad \forall i, k, \tag{3.10}
\]

and

\[
\sum_{i=1}^{m} \sum_{k=1}^{s} x_{ijk} = \begin{cases} 
  1 & \text{if Target } j \text{ is paired,} \\
  0 & \text{otherwise};
\end{cases} \quad \forall j. \tag{3.11}
\]

Above, \( s \) represents the depth of the rolling horizon, i.e., the number of stages considered by the optimization algorithm, to a maximum of \( n \) stages, such that there are never more stages than the number of not-intercepted targets.

For a multi-stage horizon, \( F \) similarly represents the interception times for the pursuer-target pairings in \( X \). However, since interception times for the second and subsequent stages are dependent on the targets intercepted in the previous stages, the interception times need to be represented by a tree structure, Fig. 3.1. The first level of the tree defines the pursuer’s identity,
the second level identifies the target intercepted during the 1\textsuperscript{st} stage, the third level identifies the target intercepted during the 2\textsuperscript{nd} stage, and so on up to the (s+1)\textsuperscript{th} level for an s-stage horizon. Each node on the tree represents the interception time for the pursuer-target(s) combination. In order to allow for pursuers to be unpaired at certain horizon stages, each node has (n+1) branches, such that there is one branch for each target, and one extra branch to represent the pursuer not being paired with a target at that horizon stage. This is represented as Node 0 for the group which contains the same value as its parent node and has no children nodes.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{tree_structure.png}
\caption{Tree structure of F.}
\end{figure}

In order to allow for easier traversal of F, let the \((m \times s)\) matrix be an alternate representation of X:

\[ c_{ik} = \sum_{j=1}^{n} x_{ijk} \times j \quad \forall \ i, k \quad , \]  

(3.12)

where \( c_{ik} \) represents the identity of the target being pursued by Pursuer \( i \) at Horizon-Stage \( k \). Furthermore, let the \((1 \times m)\) vector represent the total interception times for each pursuer to intercept all the targets it is paired with for all \( s \) horizon stages, respectively, as:

\[ h_i = f_{i,C_{ik},C_{ik+1},\ldots,C_{ils}} \quad k \text{ starts at } 1, \forall \ i \quad . \]  

(3.13)

Therefore, in its more generic form, (7) can be expressed as:

\[ \min Z = \min \max( h(X) ) \quad . \]  

(3.14)
3.2. **Dynamic Re-pairing Methodology**

In order to determine time-optimal pairings for interception according to the problem formulation presented above, the proposed Dynamic Repairing Methodology (DRPM) is presented as a generic solution methodology to address the most complex multi-pursuer, multi-target interception problem, that of a dynamic number of pursuers and a dynamic number of highly-maneuverable targets. One of the primary novelties of this methodology is its modular architecture, where the user can make application-specific implementation decisions within each module without affecting the functionality of the entire system. This allows for a plug-and-play type environment. Possible implementation options are discussed further in Section 3.3, including different metrics and optimization methods for task allocation, and pursuer-motion guidance techniques. An overall schematic diagram of the proposed methodology is shown in Fig. 3.2: rectangles represent functions within the modules, oblongs represent data structures, and diamonds represent decision points. All data transferred between functions or modules are represented as arrows moving between components, where the label on the arrow indicates the data structure being transferred.

The proposed architecture utilizes four modules: the Task-Allocation, Navigation, Information-Acquisition, and Data-Management modules, respectively. Data sharing occurs directionally across all modules by passing through the Data-Management module. The Task-Allocation and Navigation modules write onto and read from the Data-Management module at specified update rates. The latest states of the pursuers and targets within the workspace are estimated using the Information-Acquisition module and written to the Pursuer/Target States data structures in the Data-Management module. Once these data are available, the Task-Allocation module begins to repeatedly calculate the matrix of (estimated) interception times ($F$), from which the optimization search engine determines and updates the Data-Management module with the best available pursuer-target pairings ($X$) for utilization by the Navigation module. Each pursuer, then, independently accesses the Navigation module such that the guidance method generates acceleration commands for it to time-optimally intercept its allocated target. A motion controller converts the acceleration commands into motion commands that meet the non-holonomic constraints of the pursuers.
Fig. 3.2. Flowchart of Proposed Methodology
3.2.1. **Data-Management Module (Blackboard)**

This module is responsible for ensuring that all other modules in the system have access to the most recent data about the pursuers and targets in the workspace by creating a common point for data sharing across the system. In order to minimize data storage redundancies in the system, a blackboard approach is proposed. Four specific data structures are updated onto and also read from the blackboard. Each time a data structure is updated, the previous value is erased and replaced with new data, such that only the current information is available to the modules, Table 3.1.

<table>
<thead>
<tr>
<th>Module</th>
<th>Purpose</th>
<th>Data Structures</th>
<th>Interception Status</th>
<th>[P]</th>
<th>[T]</th>
<th>[X]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information-Acquisition</td>
<td>Update the locations and numbers of Pursuers and Targets</td>
<td>Write Write Write N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task-Allocation</td>
<td>Update the Pairings Matrix</td>
<td>Read Read Write</td>
<td>Read Read Read Write</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Navigation</td>
<td>Guide Pursuers towards their Targets</td>
<td>Read Read Read Read</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The blackboard is updated with new pursuer (P) and target (T) state information every $\Delta t_{state}$ by the Information-Acquisition module in the pursuer state, Table 3.2, and target state, Table 3.3, data structures. The state information for the pursuers and targets comprises the respective positions and translational velocities with respect to a fixed coordinate system, represented by $p$ and $v$ respectively, where $p_{Pi}$ and $v_{Pi}$ are the position and velocity of the $i^{th}$ pursuer, and, $p_{Tj}$ and $v_{Tj}$ are the position and velocity of the $j^{th}$ target. New pairings (X) are updated on the blackboard by the Task-Allocation module (at maximum) every $\Delta t_{state}$ to the Pairings Matrix, Table 3.4. This data is read by the Navigation module so that each pursuer can determine the appropriate guidance command to intercept its allocated target.

<table>
<thead>
<tr>
<th>Table 3.2. Pursuer State Data Structure [P].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pursuer no.</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>$\vdots$</td>
</tr>
<tr>
<td>$m$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3.1. Data Exchange Rights on the Data-Management Module (Blackboard).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
</tr>
<tr>
<td>Information-Acquisition</td>
</tr>
<tr>
<td>Task-Allocation</td>
</tr>
<tr>
<td>Navigation</td>
</tr>
<tr>
<td>Purpose</td>
</tr>
<tr>
<td>Update the locations and</td>
</tr>
<tr>
<td>numbers of Pursuers and</td>
</tr>
<tr>
<td>Targets</td>
</tr>
<tr>
<td>Update the Pairings Matrix</td>
</tr>
<tr>
<td>Guide Pursuers towards their Targets</td>
</tr>
<tr>
<td>Data Structures</td>
</tr>
<tr>
<td>Interception Status</td>
</tr>
<tr>
<td>Write</td>
</tr>
<tr>
<td>Write</td>
</tr>
<tr>
<td>Write</td>
</tr>
<tr>
<td>N/A</td>
</tr>
<tr>
<td>Read</td>
</tr>
<tr>
<td>Read</td>
</tr>
<tr>
<td>Read</td>
</tr>
<tr>
<td>Write</td>
</tr>
<tr>
<td>Read</td>
</tr>
<tr>
<td>Read</td>
</tr>
<tr>
<td>Read</td>
</tr>
<tr>
<td>Data Structures</td>
</tr>
<tr>
<td>[P]</td>
</tr>
<tr>
<td>[T]</td>
</tr>
<tr>
<td>[X]</td>
</tr>
</tbody>
</table>
Chapter 3: Dynamic Re-Pairing Methodology

Table 3.3. Target State Data Structure [T].

<table>
<thead>
<tr>
<th>Target no.</th>
<th>Position ($p_{T_j}$)</th>
<th>Velocity ($v_{T_j}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.4. Pairings Matrix Data Structure [X].

<table>
<thead>
<tr>
<th>Pursuer no.</th>
<th>Target no.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>$m$</td>
<td>$n$</td>
</tr>
</tbody>
</table>

By having the pairings updated as frequently as, if not faster than, the state information of the pursuers and targets is made available, it is ensured that pursuers receive relevant pairings. The value of $\Delta t_{state}$ is selected by the user depending on their interception requirements for the application at hand.

The third set of data on the blackboard is the Interception Status, which allows the Data-Management module to act as a control centre for the system – namely, to start and stop the execution of the Navigation and Task-Allocation modules. While the Interception Status is false, ($n > 0$ – i.e., while there are still not-intercepted targets in the workspace) the Data-Management module updates the Navigation and Task-Allocation modules to alert them to start executing, or, at the end of each of their module executions that they need to repeat execution. Once the Interception Status is set to true ($n = 0$), the Data-Management module updates the Navigation and Task-Allocation modules to alert them that the current interception pursuit has ended, and that they no longer need to continue pairing and navigating pursuers.

In order to demonstrate how a blackboard is utilized, a simplified example is discussed in Appendix B for a 2-pursuer, and 3-target scenario.
3.2.2. **Information-Acquisition Module**

This module determines the states of the pursuers/targets within a designated workspace. It comprises a sensory system capable of determining the positions and velocities of all pursuers and targets. This module must update the Data-Management module with this state information and the Interception Status every $\Delta t_{\text{state}}$. Any sensory system which can attain the required state information and update the Data-Management module within the specified update rate can be used for this module.

3.2.3. **Task-Allocation Module**

This module is responsible for determining the one-to-one pursuer-target interception pairings. The two key components of this module are: (i) the calculation of estimated interception times ($F$), and (ii) the optimization search engine.

The calculation of interception times, for each pursuer to intercept each target in the workspace, is based on the specific guidance method used. This component reads the pursuer/target state information from the Data-Management module, then calculates and sends the matrix of interception times ($F$) to the multi-stage optimization search engine to determine the pairings.

The optimization search engine can employ any integer optimization method selected by the user, but must be able to determine an optimal pairing set that meets the criteria outlined in the Problem Formulation within the required update rate, $\Delta t_{\text{state}}$, such that new pairings are available to the pursuers before target state information is updated. In order to ensure the Pairings Matrix ($X$) is never left unfilled, and subsequently pursuers are not left idle during a pursuit, a random set of pairings is initially written onto the Data-Management module, and is subsequently updated with the optimal pairings being rewritten every time a ‘better’ pairing combination is found. An optimization iteration ends when either (i) the algorithm finishes searching through all stages of the multi-stage optimization, or (ii) the pursuers need to be updated with new motion commands ($\Delta t_{\text{state}}$ has been reached). Once the pairings have been updated and new state information is obtained, the Task-Allocation module restarts with the new state information, keeping the previous pairing solution as a starting point upon which to find a ‘better’ solution.
As discussed in the Problem Definition, Section 3.1, a rolling horizon approach is used in the determination of the optimal pairings. The multi-stage optimization searches sequentially through the first, second, and subsequent stages of the optimization to determine the optimal pairing solution. The optimization searches through a maximum of \( n \) stages, where \( n \) is the number of not-intercepted targets in the workspace. The actual number of stages searched depends on the time allowed for the search. Each time a better solution is found during the first stage, it is updated to the Pairings Matrix \( (X) \) as discussed above. If a better solution is found during the second or subsequent stages, only the pursuer-target pairings which have interceptions during the first stage of the optimization are updated to the Pairings Matrix for pursuer execution. Any pursuer not allocated to intercept a target during the first stage of the optimization can be allocated a fictitious task.

3.2.4. Navigation Module

This module navigates each pursuer towards its allocated target. While the Interception Status on the Data-Management module is set to \textit{false}, each pursuer executes this module, such that each pursuer can be navigated concurrently, rather than sequentially, towards its allocated target. Furthermore, due to the possible parallelism of the trajectory-planning process, the computational burden on the system is not tangibly increased by a greater number of pursuers.

There are two components which are required for this module that can be implemented as desired by the user: the Guidance Method, and the Motion Controller.

The guidance method reads the target states from the Data-Management module and determines the required interception accelerations for the pursuers autonomously every \( \Delta t_{motion} \), which is significantly less than \( \Delta t_{state} \). This faster update rate allows the pursuers to better adapt their motions to intercept highly-maneuverable targets, since pursuers know their current location and can adapt their motion to any update in pairings determined between target state updates. These accelerations are, then, sent to the pursuer’s motion controller for conversion into motion commands that are executable by the pursuers. Any guidance method and motion controller which can determine the appropriate interception accelerations and motion command within the specified \( \Delta t_{motion} \) update rate can be used in this module.
3.3. **Dynamic Re-Pairing Methodology Implementation**

The proposed implementation methodology for the interception of mobile targets by a dynamic team of robotic pursuers addresses several issues related to pursuer/target pairing, target tracking, and pursuer-motion trajectory planning, each of which are encompassed in the different modules of the DRPM. Possible implementation options for each issue are presented below; however, any approach which would meet the specific criteria of the corresponding module in the DRPM is sufficient for use. This allows the solution methodology to be customized by the user, as desired, to the specific application at hand.

3.3.1. **Pursuer/Target Pairing (Task-Allocation Module)**

The objective of the Task-Allocation module is to determine pursuer/target pairings for minimum total interception time based on the pursuer/target state information acquired by the Information-Acquisition module. To complete this objective, three main components are needed: (i) a metric to evaluate the optimality of individual pairings, (ii) a search method to determine the optimal set of pairings, and (iii) a method to deal with inequalities between the number of pursuers and targets.

3.3.1.1. **Interception Time Metric**

In order to evaluate the optimality of individual pairings for time-optimal interception, the minimization of the overall total interception time for the team of pursuers is required. As defined in the problem definition, the total interception time of the group of pursuers and targets is controlled by the pairing with the longest interception time. As a result, time-to-intercept is selected as the appropriate metric. The individual pursuer-target interception times must be estimated for all pursuer-target combinations, such that an optimal combination of pursuer-target pairings can be selected by the optimization search engine. However, one must note that for highly-maneuverable targets, actual interception times cannot be calculated in real-time ahead of interception due to uncertainty in target motion. Thus, since the pairing optimization algorithm is computationally intensive, a simplified estimate of the interception time is used.

The simplest method of estimating an interception time is to assume a direct correlation between the distance/range between a pursuer-target pair and the time for that pursuer to intercept its target:
Chapter 3: Dynamic Re-Pairing Methodology

\[ \mathbf{p}_C = \mathbf{p}_{Tj} - \mathbf{p}_{Pi} \]  \hspace{1cm} (3.15)

\[ t_{ij} \propto R_{TP} = \|\mathbf{p}_C\| \]  \hspace{1cm} (3.16)

where \( \mathbf{p}_C \) is closing distance, \( t_{ij} \) is the interception time of Target \( j \) by Pursuer \( i \) at a given time instance, and \( R_{TP} \) is the Euclidean distance between the \( i^{th}-j^{th} \) pursuer-target pair.

While for the majority of the time the above formulation will aid in determining optimal pairings, it does not take into account the specific (instantaneous) velocities of the targets and pursuers. As a result, the time it takes for a pursuer located close to, but moving away from, a target to change its course and head towards the target is not included in the estimation. Therefore, the use of a modified estimate of the interception times is proposed, and is calculated according to the Advanced Predictive Guidance Law (APGL) described in [56]:

\[ t_{ij} = \frac{R_{TP}}{v_R} \]  \hspace{1cm} (3.17)

where \( v_R \) is the magnitude of the closing velocity, \( \mathbf{v}_c \), projected along \( R_{TP} \) defined as:

\[ \mathbf{v}_c = \mathbf{v}_{Tj} - \mathbf{v}_{Pi} \]  \hspace{1cm} (3.18)

\[ v_R = -\frac{\mathbf{p}_C \cdot \mathbf{v}_C}{\|\mathbf{p}_C\|} \]  \hspace{1cm} (3.19)

In order to ensure that pairings are stable over time, especially when dealing with cases of multiple potential targets with near-equal estimated interception times or when obstacle avoidance is used to prevent collisions between pursuers in the environment, a simple filter is implemented to prevent the inefficient switching of allocations (i.e., re-pairings). This filter is incorporated into the calculation of the interception times of individual pairings, such that a previous pairing would be favoured until a pairing with a certain percent improvement in interception time becomes available. Therefore, a flip-flop in pairings would not occur for a pursuer every time a near-equidistant set of targets is present.

As the pursuer-target state information is updated on-line, the matrix of interception times, \( \mathbf{F} \), is also calculated on-line as new information becomes available. The interception times for all pursuer-target pairing combinations must be estimated in order for the optimization to determine
the best set of pairings. As a result, a computationally efficient method of estimating these interception times, such as those presented above, is required so that the majority of the computational effort can be spent on finding the optimal pairing solution.

3.3.1.2. Optimization Search Engine

As outlined in Section 3.2.3, the optimization search engine can employ any integer optimization method selected by the user that is able to determine an optimal pairing set within the required update rate, $\Delta t_{\text{state}}$. The computational complexity of seeking globally optimal re-pairing at any instant, by searching through all possible pairing solutions, would make the problem intractable for even modest numbers of pursuers and targets. Therefore, an integer-optimization method can be utilized which is capable of providing results similar to those of a brute-force enumeration within the required update rates of the module.

In this Thesis, Simulated Annealing, [57], was selected as it has the ability to avoid settling in local minima in favour of locating the global minimum by using a time-varying probabilistic measure to determine which pairing permutations should be searched. For single-stage rolling-horizon examples, Simulated Annealing provides optimal results with a sufficiently fast update rate.

Naturally, the computational burden would significantly increase with the depth of the rolling horizon. As a result, the search engine needs to return an optimal pairing solution to the blackboard each time a better one is found, regardless of whether it is found within the same horizon stage, or a subsequent one. Since these results are written onto the blackboard as they become available and used as needed, if at the end of a stage the optimization pairings are not yet required, the search engine writes these pairings to the blackboard, and moves on to the next stage. The optimization stops only when the pursuers access the blackboard to inquire about their newest pairings.

Furthermore, in order to ensure that pairings are stable over time, i.e., to prevent the inefficient switching of allocations, especially, when dealing with cases of multiple potential targets with near-equal estimated interception times, a simple filter is implemented as discussed above. A previous pairing, then, would be favoured until a pairing with a certain percentage
improvement in interception time becomes available, and maintains the stability of the system such that pursuers can actually intercept their allocated targets.

3.3.1.3. **Coping with Inequalities in the Numbers of Pursuers and Targets**

As one of the primary objectives of this work is to develop a generic task-allocation methodology, the corresponding implementation technique must also be able to cope with inequalities in the numbers of pursuers and targets.

In the case of an excess number of pursuers, allocating unpaired pursuers to either real or fictitious targets in the workspace could move them in an *anticipatory behaviour* of intercepting one of the remaining targets in the workspace. Herein, such an approach was deemed to be beneficial due to the inherent uncertainty in the movements of the targets. Allocating an unpaired pursuer to a real target in the workspace can be achieved by pairing it with one that the pursuer can intercept in the least estimated amount of time. Having more than one pursuer paired with a target may yield a higher probability that the target will be intercepted in a time-optimal manner; however, this can bias pursuers towards specific targets. Thus, herein, the generation and use of fictitious targets, called “ghost targets”, which can guide a pursuer towards a convergence point of all the not-intercepted targets’ motions is proposed.

Ghost targets can be located at a weighted distance amongst the remaining targets in the workspace, represented by:

\[
p_{G_i} = \sum_{j=1}^{n} \left( \frac{t_i}{\sum_{j=1}^{n} t_i} \times p_{T_j} \right),
\]

(3.19)

where, for Pursuer \(i\), \(p_{G_i}\) is the location of its corresponding ghost target.

Assigning an unpaired pursuer to intercept a ghost target does not increase the total interception time for the team since a pursuer is classified as ‘unpaired’ only after the optimization search cycle. Namely, the estimated interception time of a ghost target by the unpaired pursuer is not factored into the calculation of the total interception time of the group of targets.

In the case of excess targets, any target unpaired after the optimization search is left not-pursued. Since this methodology only deals with one-to-one pairings, there exist few options to
Chapter 3: Dynamic Re-Pairing Methodology

ensure such targets do not escape the pursuers. One alternative is to create a priority status for the targets, such that high-priority targets will be included within the set of paired pursuers-targets during the optimization.

3.3.2. Target Tracking (Information-Acquisition Module)
The on-line task-allocation methodology proposed in this paper requires up-to-date target pose information at acceptable rates, which can be attained via an external centralized sensory system or a distributed system. The exact type of sensory-system required would depend on the application at hand, and may include an overhead centralized vision system, a signal-localization system, or on-board cameras on pursuer robots, as selected by the user.

Namely, the Information-Acquisition Module requires frequent and consistent analysis of instantaneous snapshots of the environment throughout the pursuit. From these snapshots, pursuers and targets must be localized with respect to an absolute coordinate system such that interception times can be estimated as accurately as possible.

In order to accurately identify targets, a mechanism first to determine which objects in the environment are indeed targets, and second to distinguish between different targets, must be selected by the user. When using a vision system a common method of identifying targets can be the use of unique visual markers or features located on the targets, including colour or pattern coding. When using signal localization, on the other hand, often each target is set to emit a different frequency signal so that each target has a unique signature.

3.3.3. Pursuer Motion Trajectory-Planning (Navigation Module)
For guiding pursuers to targets of interest in a time-optimal manner, navigation guidance methods are recommended due to their ability to adapt, readily, to changes in target maneuver. These navigation commands generated would need to be converted to appropriate motion commands in order to physically maneuver the pursuers within their environments.

3.3.3.1. Guidance Method
Guidance theories have been used as a means to efficiently navigate interceptors towards highly-manoeuvrable targets in minimal computation burden. In navigation guidance, a pursuer is accelerated to move towards an estimated interception point on the collision triangle between the pursuer and target [51]. As the motion of the target changes, this estimated interception point
also changes, varying the acceleration which is applied to the pursuer, and allowing it to efficiently navigate towards the estimated interception location. Therefore, at a required update rate, the target’s velocity, acceleration, and jerk must be estimated to determine an appropriate acceleration command for the pursuer. These three terms can be estimated, for example, by utilizing an Extended Kalman Filter (EKF) based on the estimated target position and velocity received from a sensory system (e.g., [13], [48]).

3.3.3.1.1. Proportional Navigation Guidance Law
A typical guidance method, Proportional Navigation (PN), can be used as an effective means of navigating a pursuer towards a target [58]. PN determines an acceleration command for the pursuer which is inversely proportional to the square of the time-to-go until interception and proportional to the Zero-Effort-Miss (defined by the terms in the bracketed portion of Eq. 3.20) below [58]:

\[
a_{PN} = \frac{N}{t_{go}^2} [p_c + v_c \cdot t_{go}],
\]

where \( N \) is a navigation ratio (usually in the order of 3 to 5), and \( t_{go} \) is the time-to-go until interception. However, as can be noted from the above equation, the PN law does not directly take target maneuver into account when determining an appropriate acceleration command for the pursuer, thus, its application is not ideal for highly-maneuvering targets.

3.3.3.1.2. Advanced Predictive Guidance Law
The Advanced Predictive Guidance Law (APGL) can be used as a more effective means of navigating a pursuer towards a highly-maneuverable target [56]. The APGL extends the fundamentals of PN to better adapt to highly-maneuverable targets by accounting for target maneuver. In the APGL, the predicted intercept between each pursuer and target can be individually calculated on-line by integrating the non-linear pursuer and target acceleration equations forward in time at each guidance update. This is achieved by trying to yield a zero-miss distance while minimizing:

\[
\int_0^{t_F} a_p^2(t) \, dt,
\]

where \( a_p(t) \) is the pursuer acceleration needed to intercept the target at time instant \( t \), and \( t_F \) is the total duration of the interception. \( a_p(t) \) is defined by [56]:
where $t_{go}$ is the time-to-go until interception from $t$ to $t_F$, $\dot{\lambda}$ is the rate of change of the line of sight, $\omega$ is the target maneuver frequency, $\mathbf{a}_T$ is the target acceleration, and $\mathbf{a}_J$ is the target jerk. Since the APGL explicitly accounts for the target acceleration and jerk, it is better able to account for changes in target motion in a time-optimal manner.

### 3.3.3.2. Obstacle Avoidance

As the pursuers operate in an environment where the movements of targets are highly maneuverable, an obstacle-avoidance method should be implemented to ensure that no collisions occur amongst the pursuers. Any obstacle-avoidance method that minimizes deviation from the pursuer’s desired direction of travel in order to avoid these dynamic obstacles would be acceptable. To account for such issues, the Obstacle Avoidance Navigation Law (OANL) presented in [56] is recommended. The OANL, which uses the Collision-Cone approach to identifying obstacles to be avoided [13], is specifically designed to work with the APGL to steer a pursuer toward its target around obstacles in its path. As such, when a pursuer navigates toward a target, all other pursuers in the workspace are treated as obstacles that need to be avoided.

The use of obstacle avoidance with the guidance law does not affect the optimality of pairings as they are continuously being re-evaluated on-line. Hence, if a deviation in a pursuer’s initial path occurs during obstacle avoidance that is significant enough to result in the optimal pairing of this particular pursuer to an alternate target, the optimization search engine will return this alternate set of pairings.

### 3.3.3.3. Motion Controller

Acceleration commands generated by the guidance method need to be verified and translated into executable motion commands for the pursuers by means of a motion controller. First, the motion controller needs to ensure the acceleration commands generated by the guidance method are achievable by the pursuers by directly considering the kinematic model of the pursuers including the necessary non-holonomic constraints. If the acceleration command is feasible, it is converted into an appropriate motion command. Otherwise, the maximum feasible acceleration that can be achieved while complying with the non-holonomic constraints of the pursuer, which provides the least directional deviation from the acceleration requested, can be selected to be
converted into the motion command. For ground vehicles, such as the ones used in our experiments, the kinematic model of a differentially-driven wheeled mobile robot can be used to directly convert acceleration commands derived in Eq. (3.22) into motion commands consisting of right and left wheel speeds.

### 3.4. Simulations

A number of simulations were conducted to examine the performance of the proposed on-line task-allocation methodology for the time-optimal interception of a group of highly maneuverable targets. The first set of simulations, presented in Section 3.4.2, demonstrates the effectiveness of the on-line re-pairing method to minimize total interception time, the second set, presented in Section 3.4.3, demonstrates the benefits of using a rolling-horizon approach, and the third set, Section 3.4.4, presents the method’s ability to cope with variable numbers of pursuers and highly-maneuverable targets.

#### 3.4.1. Methodology

For simulations presented in Sections 3.4.3, and 3.4.4 where the trajectories of the pursuers and targets over the course of pursuit can be seen in graphical format, target trajectories are denoted by dashed lines, and those of pursuers by solid lines. The colour of the pursuer trajectory denotes the target with which it is paired. The starting positions of the pursuers and targets at the onset of pursuit are denoted by hollow circles, and the interception of a target by a pursuer is shown by a hollow square. A pursuer/target appearing during the pursuit is denoted by a solid circle, a pursuer/target disappearing during the pursuit is denoted by a solid triangle, and the instantaneous location of a ghost target by a ‘+’.

Interception requirements and velocity limitations on the pursuers and targets varied depending on the simulation example, and thus are presented individually in the sections below.

#### 3.4.2. Effectiveness of the On-line Re-pairing Methodology

In order to demonstrate the ability of the DRPM to determine optimal re-pairings on-line, when compared to a randomized static pairing system, simulations were conducted for various numbers of pursuers and targets, which remained constant throughout the pursuit. The mean interception times of these cases, with 200 random trials each, are presented in Table 3.5.
### Table 3.5. Mean Interception Times (s) for 200 Trials.

<table>
<thead>
<tr>
<th>Number of Pursuers</th>
<th>Number of Targets</th>
<th>With Random Static Pairing (s)</th>
<th>With On-line Re-pairing (s)</th>
<th>Percentage Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>9</td>
<td>15.80 (0.45)</td>
<td>7.50 (2.60)</td>
<td>52%</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>6.68 (0.15)</td>
<td>3.87 (1.20)</td>
<td>42%</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>23.50 (4.23)</td>
<td>10.50 (2.47)</td>
<td>55%</td>
</tr>
</tbody>
</table>

For the static random allocation system, a pursuer is allocated to a target until its interception, without the ability to be re-paired en-route, and only re-paired with a new target (which is not currently allocated to another pursuer) once its initial task is completed. For the on-line re-pairing system, the DRPM re-evaluated the pairings every 150ms and re-allocated targets as necessary, often diverting pursuers from a previous target to another which had become more optimal to intercept during pursuit. All targets were given random trajectories and both pursuers’ and targets’ starting locations were randomized. The maximum speed of the pursuers was limited to 200 mm/s and the maximum speed of the targets was limited to 100 mm/s, and a target was said to be intercepted when there was less than 10 mm relative distance between the target and pursuer. The significantly decreased mean interception times presented in Table 3.5, which concur with the similar benefits proven for the dial-a-ride problem in Chapter 2, reflects the clear benefit of on-line re-pairing to decrease total interception time.

#### 3.4.3. Benefits of Using a Rolling Horizon

Through the use of a rolling horizon, the search engine would have the ability to look past the first interception between a target and pursuer to subsequent interceptions, thus, potentially minimizing the total interception time of not only the first set of interceptions, but multiple sets of interceptions. The simulation example herein presents the differences in pairing combinations for a rolling horizon with one and two stages for a 2-pursuer (P1, P2) – 2-target (T1, T2) case, where no pursuers or targets appear or disappear during the pursuit, Fig. 3.3. The maximum speed of the pursuers was limited to 200 mm/s and the maximum speed of the targets was limited to 100 mm/s, and a target was said to be intercepted when there was less than 10 mm relative distance between the target and pursuer. 

In the one-stage horizon case, Fig. 3.3 (a), the optimization search engine determines pairings for the first horizon depth only. Possible pairing combinations are either {(P1-T2), (P2-T1)}, or {(P1-T1), (P2-T2)}. Whereas, in the two-stage horizon case, Fig. 3.3 (b), the
optimization search engine determines pairings for the first- and second-stage horizons, leading to a larger set of possible pairings: \{(P1-T2-0), (P2-T1-0)\}, \{(P1-T1-0), (P2-T2-0)\}, \{(P1-T1-T2), (P2-0-0)\}, \{(P1-T2-T1), (P2-0-0)\}, \{(P1-0-0), (P2-T1-T2)\}, or \{(P1-0-0), (P2-T2-T1)\}, where 0 represents no pairing during that horizon stage. No pursuer can be left unpaired during the first horizon, and then paired during a subsequent horizon. Pursuers left unpaired by the optimization, are allocated to ghost targets for this simulation example, as outlined in Section 3.3.1.3.

For the one-stage horizon, each pursuer is paired with a unique target; P1 is paired with and intercepts T2, and P2 is paired with and intercepts T1, yielding an interception time of 4.9 s. However, in the two-stage horizon case, a faster time (4.0 s) is achieved by having the search engine look ahead through 2 stages of interceptions, and conclude that it would be faster to have P1 intercept both targets, and leave P2 unpaired. While P2 is unpaired, it is assigned to a ghost target (dotted red line between T1 and T2), so as to move it in an anticipatory behaviour towards both targets in case of a change in trajectory of the targets.

### 3.4.4. Re-pairing Examples of Proposed Methodology

In order to demonstrate the ability of the proposed solution methodology to adapt to varying numbers of pursuers and highly-maneuverable targets during pursuit and account for different types of interceptions, four simulation examples are presented herein, Table 3.6. In order to simulate the physical limitations of the pursuers used in experimentation, their maximum speed and lateral acceleration were limited to 20 mm/s and 20 mm/s², respectively. A one-stage rolling horizon with the APGL guidance method and interception-time metric was utilized. A target was
said to be successfully intercepted when there was less than 30 mm relative distance in both x and y directions between the centroids of the pursuer and target. Pairings were continually updated by the optimization search engine at a frequency of five updates per second.

One objective of the generic methodology used in the DRPM, is to be able to deal with different types of interception scenarios. To demonstrate this, examples for both interception types discussed in Section 1.3 are presented. The two interception types are disengagement, when a pursuer disengages from its assigned target once it intercepts it and moves on to intercept another target, and consumption, when a pursuer is consumed by the target upon interception and is not available to intercept any more targets.

<table>
<thead>
<tr>
<th>Simulation Case</th>
<th>Varying Numbers of Pursuers</th>
<th>Varying Numbers of Targets</th>
<th>Type of Interception</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No</td>
<td>No</td>
<td>Disengagement</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Yes</td>
<td>Disengagement</td>
</tr>
<tr>
<td>3</td>
<td>Yes</td>
<td>Yes</td>
<td>Consumption</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>Disengagement</td>
</tr>
</tbody>
</table>

### 3.4.4.1. Case 1: Fixed Numbers of Pursuers and Targets throughout Pursuit

In this simulation example, a fixed number of pursuers is paired with and intercepts a fixed number of targets, where the pursuers disengage from the targets upon interception. The purpose of this example is to demonstrate the ability of the pursuers to adapt their motions to their pairings and the motion trajectories of their allocated targets in a time-efficient manner. The trajectories of the pursuers and targets can be seen in Fig. 3.4, with the trajectory information of the targets located in Table 3.7.

As can be seen in Fig. 3.4, first, Pursuer 1 (P1) is paired with Target 3 (T3), Pursuer 2 (P2) with Target 1 (T1), and Pursuer 3 (P3) with Target 2 (T2).

- At Time A, P2 intercepts T1, and is re-paired with T4.
- At Time B, P3 intercepts T2, and is re-paired with a ghost target, GT3, at the weighted distance between the current positions of T3 and T4.
- At Time C, P2 intercepts T4, and since there remains only one not-intercepted target, all ghost locations collapse onto the position of T3.
- At Time D, P3 enters the scene and is paired with T5, while all other pairings remain
unchanged.

- At Time E, P1 intercepts T3.

### Table 3.7. Target-Trajectory Information for Fig. 3.4.

<table>
<thead>
<tr>
<th>Target Identity</th>
<th>Motion Type</th>
<th>Max Speed (mm/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sinusoidal</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Circular</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>Sinusoidal</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>Circular</td>
<td>14</td>
</tr>
</tbody>
</table>

![Fig. 3.4. Simulation Trajectories for a fixed number of pursuers and targets.](image)

### 3.4.4.2. Case 2: Dynamic Numbers of Targets with Fixed Number of Pursuers

In this simulation example, a fixed number of pursuers is paired with and intercepts a dynamic number of targets, where the pursuers disengage from the targets upon interception. The purpose of this example is to demonstrate the ability of the pursuers to adapt their pairings to changes in the increase and decrease of targets throughout pursuit. The trajectories of the pursuers and targets can be seen in Fig. 3.5, with the trajectory information of the targets located in Table 3.8.

As can be seen in Fig. 3.5, P1 is paired with T4, P2 with T2, P3 with T3, and P4 with T1.

- At Time A, T5 appears, and is left un-paired as there are no un-paired pursuers available...
to be paired with it.

- At Time B, P3 intercepts T3, and is re-paired with T5.
- At Time C, P1 intercepts T4, and is re-paired with a ghost target, GT1, at the weighted distance between the current positions of T1, T2, and T5 according to Equation 3.19.
- At Time D, T1 disappears, and P4 is re-paired with a ghost target, GT4, and P1 is re-paired with GT2, which are at the weighted distance between the current positions of T2 and T5.
- At Time E, T6 appears, and P4 is re-assigned to T6. As a result of the addition of the new target, P1 is re-paired with GT3, which is at the weighted distance between the current positions of T2, T5, and T6.
- At Time F, P2 intercepts T2, and P3 intercepts T5 near-simultaneously, and as such there exists only one target remaining. The locations of all ghost targets collapse onto the location of T6, which is then pursued by all pursuers.

![Simulation Trajectories for a dynamic number of targets disengaged on interception.](image)
Table 3.8. Target-Trajectory information for Fig. 3.5.

<table>
<thead>
<tr>
<th>Target Identity</th>
<th>Motion Type</th>
<th>Max Speed (mm/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sinusoidal</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>Random Circular</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>Random Linear</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>Random Linear</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>Sinusoidal</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>Linear</td>
<td>10</td>
</tr>
</tbody>
</table>

3.4.4.3. Case 3: Dynamic Numbers of Targets with Pursuers Consumed on Interception

In this simulation example, pursuers are paired with and intercept a dynamic number of targets, where the pursuers are consumed by the targets upon interception (and subsequently disappear). The purpose of this example is to demonstrate the ability of the DRPM to deal with consumption-type interceptions and re-pair appropriately based on the changes in the numbers of targets and pursuers.

Due to the nature of a consumption-type interception, it naturally addresses a dynamic number of pursuers. As a pursuer intercepts a target and is consumed, the number of pursuers and targets in the environment decreases, and all remaining pursuers’ pairings need to adapt to this change. The trajectories of the pursuers and targets can be seen in Fig. 3.6, with the trajectory information of the targets located in Table 3.9.

As can be seen in Fig. 3.6, first, P2 is paired with T2, P3 with T3, and P4 with T1, and P5 with T4. Since there are more pursuers than targets, P1 is paired with a ghost target, GT1.

- At Time A, T4 is intercepted, and consumes P5. As a result of the loss of the target, P1 is re-paired with a new ghost-target, GT2.
- At Time B, T5 appears. P4 is re-paired with T5, and P1 is re-paired with T1.
- At Time C, T1 disappears, and P1 is re-paired with GT3.
- At Time D, T6 appears, and P1 is re-paired with T6.
- At Time E, P2 intercepts T2 and is consumed on interception.
- At Time F, P3 intercepts T3, and is consumed on interception.
• At Time G, P1 intercepts T6, and is consumed on interception.

<table>
<thead>
<tr>
<th>Target Identity</th>
<th>Motion Type</th>
<th>Max Speed (mm/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sinusoidal</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>Random Circular</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>Random Linear</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>Linear</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>Sinusoidal</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>Linear</td>
<td>10</td>
</tr>
</tbody>
</table>

Fig. 3.6. Simulation Trajectories for a dynamic number of targets with pursuers consumed on interception.

3.4.4.4. Case 4: Dynamic Numbers of Pursuers and Targets in a Disengagement Pursuit

In order to demonstrate the ability of the proposed solution methodology to adapt to the numbers of pursuers and targets increasing and decreasing during a pursuit, a detailed simulation example is shown in Fig. 3.7. In this simulation, both pursuers and targets appear and disappear randomly during the pursuit.

The example starts with five pursuers and five targets, with the trajectories and speeds of all targets given in Table 3.10. The simulation graphs shown in Fig. 3.7 have been segmented to demonstrate the changes in pursuers’ pairings and trajectories from the onset of the pursuit up
Chapter 3: Dynamic Re-Pairing Methodology

until each of the defining moments of the pursuit, where an event forcing a change in pairings occurs.

<table>
<thead>
<tr>
<th>Target Identity</th>
<th>Motion Type</th>
<th>Max Speed (mm/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Random</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Random</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>Random</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>Sinusoidal</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>Sinusoidal</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>Circular</td>
<td>14</td>
</tr>
</tbody>
</table>

As can be seen in Fig. 3.7. (a), initially, P1 is paired with T1, P2 with T2, P3 with T3, P4 with T4, and P5 with T5.

- At Time A, T1 turns 90° counter-clockwise, T2 turns 45° counter-clockwise, and T3 turns 120° counter-clockwise. The change in directions of these targets is demonstrated by the thin red lines extending from the location at which they turn, into the direction of their new headings. These changes in direction cause the pairings of P1 and P2 to switch, such that P1 is now paired with T2 and P2 with T1, as the switch yields a potentially shorter total interception time.

**Fig. 3.7. (a):** Simulation Trajectories up to and including Time A.
• At Time B, Pursuer 6 (P6) enters from the left side of the workspace; however, since existing pairings are deemed to be optimal, P6 is allocated to a Ghost Target (GT6) located at the weighted centroid location of the targets in the scene.

![Simulation Trajectories up to and including Time B.](image)

**Fig. 3.7. (b):** Simulation Trajectories up to and including Time B.

• At Time C, P3 intercepts T3 and it is re-allocated to a Ghost Target (GT3), and all other pairings remain the same.

![Simulation Trajectories up to and including Time C.](image)

**Fig. 3.7. (c):** Simulation Trajectories up to and including Time C.
Chapter 3: Dynamic Re-Pairing Methodology

- At Time D, P2 intercepts T1, and P5 disappears simultaneously. At this point in time, P2 is re-allocated to T2 and P1 is re-allocated to T5; all other pairings remain the same.

Fig. 3.7. (d): Simulation Trajectories up to and including Time D.

- At Time E, Target 6 (T6) appears. P6 is re-paired with T6, and all other pairings remain. Since P6 now has a ‘real’ target to intercept, GT6 disappears.

Fig. 3.7. (e): Simulation Trajectories up to and including Time E.
• At Time F, P4 intercepts T4. P4 is then re-allocated to a Ghost Target (GT4), and all other pairings remain the same.

Fig. 3.7. (f): Simulation Trajectories up to and including Time F.

• At Time G, P1 intercepts T5, and T2 disappears simultaneously. At this point in time, only one target remains in the workspace, T6, and as a result all ghost target locations collapse onto the location of T6, and it is pursued by all pursuers.

Fig. 3.7. (g): Simulation Trajectories up to and including Time G.
The simulation ends when P6 intercepts T6 at Time H.

![Simulation Trajectories](image)

**Fig. 3.7. (h):** Simulation Trajectories up to and including Time H.

### 3.5. Summary

The proposed Dynamic Re-Pairing Methodology presented in this Chapter, is capable of determining pursuer-target pairings for minimum total interception time in dynamic environments in an on-line manner. Through the use a modular approach to allow computational effort to be focused on the determination of the optimal pairing solution, navigation guidance is an efficient motion-planning methodology for adapting readily to changes in pursuer-target pairings and target maneuver. The on-line re-evaluation of pairing optimality and re-allocation due to changes in target maneuver has demonstrated a reduction in total interception time as compared to random static assignment systems. The use of a rolling-horizon to look past the first interception to multiple subsequent interceptions has presented an additional reduction in interception time. As can be noted through the pursuers’ trajectories of the above presented examples, the proposed solution methodology is efficient in determining pursuer-target re-pairings on-line in response to changes in pursuer/target number and target maneuver.
4. Dynamic Re-Pairing Methodology – Experiments

In order to examine the real-time performance of the proposed solution methodology in autonomous vehicle pursuit, numerous experiments were conducted, three of which are presented in detail in this Chapter. These experiments demonstrate the ability of the Dynamic Re-Pairing Methodology (DRPM) to work under non-ideal conditions (as were presented in Chapter 3) that are more representative of realistic environments. Section 4.1 describes the experimental setup, including details on the vision system and pursuer robots. Section 4.2 presents the pursuer pairings and trajectories of three experiments and their corresponding simulation results. Conclusions are presented in Section 4.3.

4.1. Experimental Set-up

The experimental set-up comprised a bounded workspace with an overhead CCD camera attached to a PC. The physical layout of this setup is presented in model form in Fig. 4.1, with an actual image of the setup provided in Fig. 4.2. The software for the experiments, running on a 2.79 GHz Intel Core 2 Duo CPU, consisted of three primary modules developed in a Visual C++ multi-threaded application: (i) image acquisition and processing, (ii) optimization of pursuer-target pairings, and (iii) trajectory planning and communication, respectively, along with a blackboard for common data sharing. These modules corresponded to
those of the Image-Acquisition, Task-Allocation, Navigation, and Data-Management modules of the DRPM respectively.

An analog overhead CCD camera, mounted 2000 mm above the workspace, captured images of the workspace at a rate of approximately 30 frames per second, and transferred them to the frame grabber in the PC. The workspace was 1500mm x 2000mm and was illuminated by two halogen lights mounted on either side of the camera. The image-processing module analyzed the images to extract the positions of the pursuer and target robots. The optimization module, determined the pursuer-target pairings. The acceleration commands were calculated for all pursuers by the trajectory-planning and communication module and broadcast to them via a Bluetooth module.

![Fig. 4.1. Model of the Setup’s Layout.](image1)

![Fig. 4.2. Physical Setup with Pursuer Robots.](image2)

The details of the experimental setup can be found in [59], with the vast majority repeated here for completeness. Modifications that were made to the setup since its use in [59] have also been incorporated.

### 4.1.1. Vision System

#### 4.1.1.1. Hardware

The vision system comprises a JVC TK-870U CCD camera with a Cosmicar C60607 wide-angle lens and a Matrox Meteor II frame-grabber. The CCD camera has a resolution of (640 × 480) pixels and a standard composite colour output. The Cosmicar C60607 wide angle lens has a focal length of 6 mm and provides a Field-of-View (FOV) of (3200 mm × 2400 mm). The images
taken by the camera are transferred to the Matrox Meteor II frame-grabber through a co-axial cable from the composite output. The Meteor II is installed into the PCI slot of the PC. Matrox provides a set of Visual C++ libraries that allow access to the memory of the Meteor II to extract image data.

Image-to-world coordinate transformation is done through a camera-calibration model. The overhead camera was calibrated using the same off-line, non-coplanar calibration technique that was used in [60]. This technique, computes the intrinsic and extrinsic parameters of a camera, and provides a set of equations for coordinate transformation. The intrinsic parameters include focal length, lens distortion, scale factor uncertainty, and 2D camera-image centre coordinates. The extrinsic parameters are the six translation and rotation parameters defining the 3D rigid body transformation from world-to-image coordinate system. A large calibration board, on which evenly spaced circular markers were printed, was used to provide the calibration model with known coordinate points in the workspace. The circles are 80 mm in diameter. The calibration board is 1157 mm × 1118 mm in size and contains 5 rows × 5 columns of circles. It was placed at various locations in the workspace and the pixel location of each coordinate point (circle centroid) was taken. The model parameters were then calculated using these coordinate points.

4.1.1.2. Software
The pursuer robots are colour coded for identification, and need to be distinguishable by the Image-Acquisition module to ensure correct task allocation. The raw image taken of the workspace contains three channels of data, indicating the intensities of the Red (R), Green (G) and Blue (B) colours in each pixel. This is more commonly known as the RGB colour space. Each channel contains 8 bits of information. Thus, 24-bits of information are embedded into one pixel, representing more than 16 million different colours. However, there are only four colours that are used by the identification markers: red, blue, yellow, and pink. Therefore, the 24-bits of information must be categorized into any of the four colours or a non-colour. This is called the thresholding process.

A transformation of the entire raw-image to the YCbCr colour space is performed because of its ability to separate the luminance or brightness of a pixel from its hue or colour [61]. This property makes it possible to place less importance on the change in brightness of an object, as
when the object is affected by shadows but, rather, focus on the colour of the object itself. The $YCbCr$ colour space, like the $RGB$ space, is also 3-dimensional. The first dimension, $Y$, describes the luminance, or brightness, of the pixel. The second, $Cb$, which stands for Chrominance-blue, describes the hue of the pixel in terms of its intensity of the blue colour. Lastly, $Cr$, which stands for Chrominance-red, refers to the hue of the pixel in terms of the red colour. The transformation is performed by the following equations, [60]:

\[
Y = 0.299R + 0.587G + 0.114B ,
\]

\[
Cb = \frac{(R-Y)}{1.772} , \text{ and}
\]

\[
Cr = \frac{(R-Y)}{1.402} ,
\]

where $Y$ has a range of $[0, 255]$, and $Cb$ and $Cr$ both have a range of $[-127.5, 127.5]$.

As mentioned above, four colours are used by the identification markers: blue, yellow, and red to distinguish the pursuer, and pink on each robot to determine its heading/orientation. Before thresholding can be carried out, these colours must be assigned a $YCbCr$ value. An image of identification markers containing these four colours is first taken. From the pixel values, the mean is found for each colour. The mean is now the defined $YCbCr$ value of that colour. For example, the blue colour has a $Y$ value of 140, a $Cb$ value of 10.0, and a $Cr$ value of -8.0. The four identification colours are, hereafter, referred to as the predefined colour set. When an image is examined, the Euclidean distances, in the $YCbCr$ colour space, between each pixel in the image and the predefined colour set are calculated. Since it is desirable to lessen the effects of uneven lighting or shadows, a weighted Euclidean distance is used. Less weight is placed on the $Y$ channel because it refers to the brightness of the pixel, while more weight is put on the $Cb$ and $Cr$ channels because they refer to the hue of the pixel. Through experimentation in [59], it was found that the weights in the following equation perform well to lessen the effects of shadows:

\[
D = \sqrt{0.15 (Y_{pixel} - Y_{colour})^2 + 0.425 (Cb_{pixel} - Cb_{colour})^2 + 0.425 (Cr_{pixel} - Cr_{colour})^2} ,
\]

where $D$ is the weighted Euclidean distance, $Y_{pixel}$, $Cb_{pixel}$, and $Cr_{pixel}$, are measured $YCbCr$ values of the pixel, and $Y_{colour}$, $Cb_{colour}$, and $Cr_{colour}$ are the values of the predefined colour set. It was found that the pixels on the identification marker varied no more than $18.0^\circ$ weighted Euclidean
distance from the defined $YCbCr$ value. This value was, therefore, set as the threshold distance. Thus, if a pixel is within a threshold distance of 18.0" of a certain colour in the predefined colour set, then it is considered to be that colour. If the distance is more than this threshold for all the colours of the predefined colour set, the pixel is considered a non-colour. This thresholding operation effectively transforms each pixel to contain 2 bits of information, indicating that the pixel is blue, yellow, red, pink or a non-colour.

After the image has gone through the thresholding operation, the positions of the pursuer robots are determined. A search through the image of the work floor is performed to find the markers on the robots. In order to minimize the search time, sampling is done according to markers. The dimension of the smallest marker is used to achieve the smallest sampling rate, denoted here as $l$ pixels. Starting from pixel location (0, 0) in the work floor, every $0.5l$ pixels are sampled along the $X$ and $Y$ directions. If the sampled pixel has a colour in the predefined set, a search frame is placed over that pixel. The size of the search frame is dependent on the colour of interest, but it is twice that of the width of the marker. This search frame size ensures that the marker, regardless of its orientation, would be within the frame. Every pixel in the search frame is examined and if the number of pixels of a certain colour in the search frame exceeds a predetermined threshold, then, a marker of that colour is considered to be located in that search frame. The centroid of that colour blob is then calculated to sub-pixel accuracy using the Centroid Method [62]. This operation continues throughout the whole work floor to locate all colour markers. Fig. 4.3 illustrates this search process. With all the markers located, robot identification can be performed. Each of the three pursuers is identified by a colour-coded marker. The marker contains a coloured square in the centre of the marked pattern, and a pink strip in front of the pursuer as shown in Fig. 4.4. The rest of the pursuer is coloured in black. The vision program first searches for coloured square markers (in this case blue markers). Once a blue marker is found, the algorithm looks for a pink marker within a distance of the radius of the blue marker. If a corresponding pink marker is located, then, a pursuer has been successfully found. An imaginary line is then drawn between the centre of the blue square and the centroid of the pink pattern. This line indicates the heading of the pursuer.
4.1.2. Pursuer Robots

Three identical robotic vehicles were used in the implementation of this work, which act as pursuer robots. The vehicles are the MIABOT Pro mobile robots developed by Merlin Systems [63], Fig. 4.4. These vehicles are fully autonomous and have a differential-drive system that provides an ease of implementation and compactness. They have a base plate, a top plate, two DC-motors, two gear assemblies, and two ball casters. The motors are driven by $6 \times 1.5$ V (AA) cells through a low-resistance driver integrated circuit with a slow-acting current limit at about 5A. Maximum speed of an unloaded motor is in the range of 6-8000 rpm, with the motor shafts driving the wheels through an 8:1 gearing. The motors incorporate quadrature encoders giving 512 position-pulses per rotation, and the wheels are 52 mm in diameter. So, one encoder pulse corresponds to just under 0.04 mm of movement. The top plate is marked with a coloured pattern to provide the vision system with the position and orientation of the pursuer. The kinematic model of these pursuers used by the motion controller to convert guidance commands into executable left and right wheel speeds is presented in [56], and repeated in Appendix C for completeness.
Due to the limited number of (three) mobile robots, used as pursuers, the targets were ‘simulated’. Namely, the (physical) robots pursued ‘virtual’ targets, operating in a mixed real/virtual environment. Since the optimization search engine reads the location of the pursuers and targets from the blackboard, the method in which the target-state data is acquired, whether it is simulated by the computer or determined from the vision system, is not relevant to the optimization nor to the pursuers’ motions. Namely, a pursuer robot would not ‘know’ the difference between a physical or simulated target.

4.1.3. Communication System
The communication system of the MIABOT Pro robot consists of a bi-directional Bluetooth communications module, which provides a robust frequency hopping wireless communications protocol at 2.4GHz. A Bluetooth communications card incorporated within each robot enables the host PC to communicate with it by converting the Bluetooth link to logic-level serial signals connecting to the main board processor. A PC Bluetooth dongle is supplied that plugs into the USB port on the PC which can support wireless links with up to 7 robots at once. Each Bluetooth link is a dedicated, secure, two-way channel established exclusively between the two devices that appears, to PC applications, as a virtual COM port, which can be connected to like an ordinary serial port. At the robot end, it appears as logic-level serial signals. The PC dongle acts as a Bluetooth master device (which can establish links) while each robot is a separate slave device. A slave device can only be paired with one master at any one time. If radio contact is lost, the link will be automatically restored when radio contact is regained. However, whenever the vehicle or computer is powered off, the link must generally be re-established.
4.2. Results and Discussion

Simulations were first conducted to identify the expected behaviour of the pursuers under ideal conditions identical to those of the experimental scenario, with the same hardware/software and same data-update frequencies. The simulations and corresponding experimental results are presented below for three experiment cases, Table 4.1. The real-time experiments were repeated three times under identical conditions to examine the robustness of the motion guidance and pairing methods to system noise.

<table>
<thead>
<tr>
<th>Experiment Case</th>
<th>Varying Numbers of Pursuers</th>
<th>Varying Numbers of Targets</th>
<th>Type of Interception</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No</td>
<td>No</td>
<td>Disengagement</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Yes</td>
<td>Disengagement</td>
</tr>
<tr>
<td>3</td>
<td>Yes</td>
<td>Yes</td>
<td>Disengagement and Consumption</td>
</tr>
</tbody>
</table>

For the simulation graphs, all in-figure notations follow the same legend as those of the simulations presented in Chapter 3. For the simulation and experiment examples presented herein, where the trajectories of the pursuers and targets over the course of pursuit can be seen in graphical format, target trajectories are denoted by dashed lines, and those of pursuers by solid lines. The colour of the pursuer trajectory denotes the target with which it is paired. The starting positions of the pursuers and targets at the onset of pursuit are denoted by hollow circles, and the interception of a target by a pursuer is shown by a hollow square. A pursuer/target appearing during the pursuit is denoted by a solid circle, a pursuer/target disappearing during the pursuit is denoted by a solid triangle, and the instantaneous location of a ghost target by a ‘+’. However, only pursuers/targets entering or exiting the scene are marked on the experimental results so as not to impede the visibility of the three individual runs.

The maximum speed and lateral acceleration of the pursuer robots were limited to 20 mm/s and 20 mm/s², respectively for both the simulations and experiments. A target was said to be successfully intercepted when there was less than 30 mm relative distance in both x and y directions between the centroids of the pursuer and target. A one-stage rolling horizon was used for optimal pairing, with the pursuers navigating and determining interception times according to the Advanced Predictive Guidance Law methods discussed in Sections 3.3.1.1 and 3.3.3.1.2.
4.2.1. **Experiment 1: Constant Numbers of Pursuers and Targets**

The objective of this experiment is to demonstrate the re-pairing methodology’s ability for the pursuers to adapt their motions to their pairings, and the motion trajectories of their allocated targets, in a time-efficient manner when encountering system noise. The trajectories and speeds of the simulated targets are given in Table 4.2, with the interception times for each experiment run given in Table 4.3. The simulation of the experiment is shown in Fig. 4.5 and the corresponding experimental results in Fig. 4.6.

As can be noted from both Fig. 4.5 and Fig. 4.6, each pursuer is initially paired with the target it is closest to; namely, P1-T1, P2-T4, and P3-T3, while T2 is left not-pursued:

- At Time A, P2 intercepts T4, and is re-paired with T2.
- At Time B, P1 intercepts T1, and is re-paired with a Ghost Target, GT1, located at the weighted distance between the current positions of T2 and T3.
- At Time C, P2 intercepts T2, and as such there exists only one target remaining. The locations of all ghost targets collapse onto the location of T3, which is then pursued by all pursuers.
- At Time D, P3 intercepts T3.

![Fig. 4.5. Pursuer Trajectories for the Experiment-1 Simulation.](image-url)
Chapter 4: Dynamic Re-Pairing Methodology - Experiments

Fig. 4.6. Pursuer Trajectories for Experiment 1.

Table 4.2. Target-Trajectory Information for Experiment 1.

<table>
<thead>
<tr>
<th>Target Identity</th>
<th>Motion Type</th>
<th>Max Speed (mm/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Circular</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>Straight</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>Circular</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>Straight</td>
<td>16</td>
</tr>
</tbody>
</table>

Even though, the example given here depicts a stable assignment process of targets to pursuers, during the experiments, the proposed methodology re-evaluated pairings, on average, 96 times during the pursuits. The pairing and re-pairing results of the simulations and experiments were in complete agreement for both experiments.

Table 4.3. Interception Times for Experiment 1.

<table>
<thead>
<tr>
<th>Trial</th>
<th>Interception Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>38.4</td>
</tr>
<tr>
<td>2</td>
<td>34.9</td>
</tr>
<tr>
<td>3</td>
<td>37.5</td>
</tr>
</tbody>
</table>
4.2.2. **Experiment 2: Dynamic Numbers of Targets**

The objective of this experiment is to demonstrate the re-pairing methodology’s ability to cope with a pursuit during which targets randomly appear/disappear in/from the scene. The trajectories and speeds of the simulated targets are given in Table 4.4, with the interception times for each experiment run given in Table 4.5. The simulation of the experiment is shown in Fig. 4.7 and the corresponding experimental results in Fig. 4.8.

<table>
<thead>
<tr>
<th>Target Identity</th>
<th>Motion Type</th>
<th>Max Speed (mm/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sinusoidal</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>Circular</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>Random</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>Random</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>Random</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>Circular</td>
<td>11</td>
</tr>
</tbody>
</table>

As can be noted from both Fig. 4.7 and Fig. 4.8, each pursuer is initially paired with the target it is closest to; namely, P1-T1, P2-T2, and P3-T3, while T4 and T5 are left not-pursued:

- At Time A, T1 and T2 pass each other, and it becomes more optimal for P1 to intercept T2, and P2 to intercept T1, while all other pairings remain unchanged.
- At Time B, T6 enters the scene, and is left not-pursued as the previous pairings are deemed to be optimal.
- At Time C, P3 intercepts T3, and is re-paired with T4, while all other pairings remain unchanged.
- At Time D, P1 intercepts T2, and is re-paired with T6, while all other pairings remain unchanged.
- At Time E, T1 disappears, and P2 is re-paired with T5, while all other pairings remain unchanged.
- At Time F, P3 intercepts T4 and is re-paired with a Ghost Target (GT3) located at the weighted distance between the instantaneous locations of T5 and T6, while all other pairings remain unchanged.
- At Time G, P1 intercepts T6, and since subsequently there remains only one target not-intercepted, all ghost target locations collapse onto the position of T5.
- At Time H, P2 intercepts T5.

Fig. 4.7. Pursuer Trajectories for Experiment-2 Simulation.

Fig. 4.8. Pursuer Trajectories for Experiment 2.
Chapter 4: Dynamic Re-Pairing Methodology - Experiments

For the experiment example provided herein, even though a very stable task-allocation process of targets to pursuers is depicted, during the experiments, the proposed methodology re-evaluated pairings, on average, 270 times during the pursuit. The pairing and re-pairing results of the simulations and experiments were in complete agreement.

Table 4.5. Interception Times for Experiment 2.

<table>
<thead>
<tr>
<th>Trial</th>
<th>Interception Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>63.7</td>
</tr>
<tr>
<td>2</td>
<td>64.9</td>
</tr>
<tr>
<td>3</td>
<td>65.3</td>
</tr>
</tbody>
</table>

4.2.3. Example 3: Dynamic Numbers of Pursuers

While Experiment 2 dealt with cases of randomly appearing/disappearing targets, Experiment 3 considers the cases when pursuers enter or exit the scene. Additionally, both consumed and disengaged pursuers are considered post-interception. The trajectories and speeds of the simulated targets are given in Table 4.6, with the interception times for each experiment run given in Table 4.7. The simulation of the experiment is shown in Fig. 4.9 and the corresponding experimental results in Fig. 4.10.

As can be noted from both Fig. 4.9 and Fig 4.10, each pursuer is initially paired with the target it is closest to: P1-T1, P2-T2, while T3 and T4 are left not-pursued:

- At Time A, T5 enters the scene, and is left not-pursued as the original pairings are deemed to be optimal.
- At Time B, P2 intercepts T2, and is re-paired with T3, while all other pairings remain unchanged.
- At Time C, P1 intercepts T1, and is consumed upon interception.
- At Time D, P3 enters the scene and is paired with T5, while all other pairings remain unchanged.
- At Time E, P2 intercepts T3, and is re-paired with T4.
- At Time F, P3 intercepts T5, and only one target is left in the scene. Thus, T4 is pursued by both P2 and P3.
- At Time G, P2 intercepts T4.
Fig. 4.9. Pursuer Trajectories for Experiment-3 Simulation.

Fig. 4.10. Pursuer Trajectories for Experiment 3.
Table 4.6. Target-Trajectory Information for Experiment 3.

<table>
<thead>
<tr>
<th>Target Identity</th>
<th>Motion Type</th>
<th>Max Speed (mm/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Random</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Sinusoidal</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Random</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>Sinusoidal</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>Sinusoidal</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 4.7. Interception Times for Experiment 3.

<table>
<thead>
<tr>
<th>Trial</th>
<th>Interception Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>73.2</td>
</tr>
<tr>
<td>2</td>
<td>72.2</td>
</tr>
<tr>
<td>3</td>
<td>72.0</td>
</tr>
</tbody>
</table>

For this example, even though a very stable task-allocation process of targets to pursuers is depicted, during the experiments, the proposed methodology re-evaluated pairings, on average, 380 times during the pursuits. Again, the pairing and re-pairing results of the simulations and experiments were in complete agreement.

4.3. Summary

As can be seen from the three experiment examples provided, the DRPM has been successfully implemented, and demonstrated to be functional, for the task-allocation and navigation of pursuer robots in autonomous ground vehicle pursuit. The trajectories of the pursuers in simulation and experiment results are in complete agreement, demonstrating the methodology’s ability to determine appropriate pairings and pursuer motion-trajectories regardless of system noise. Additionally, the methodology’s ability to deal with two different types of interceptions, highly-maneuverable targets, and dynamic numbers of pursuers and targets is in agreement with the findings of Chapter 3, and demonstrated through the three experimental results provided herein.
A fundamental problem for teams of autonomous agents is the on-line, optimal allocation of tasks to those who are best suited to address them. The tasks may comprise transporting a customer from a pick-up to drop-off destination, or the interception of a highly-maneuverable target. The agents may operate in dynamic environments where the numbers of tasks to be addressed and their location may vary over time. Additionally, the number of agents may also vary.

The focus of this Thesis has been on developing task-allocation methodologies for the dial-a-ride problem (DARP) and the multi-target interception problem (MTIP) such that the determination of optimal agent-task pairings would be able to adapt to dynamic, and a priori unknown changes in the numbers and locations of tasks, as well as changes in the numbers of agents. These methodologies focus on the core premise of evaluating the full on-line re-optimization of agent-task pairings frequently, to account for, and adapt the pairings set to, changes as they happen.
5.1. Summary and Contributions

The contributions of this work are the development of two on-line repairing methodologies, discussed below.

5.1.1. Dynamic Nearest Neighbour Policy

The dynamic nearest neighbour (DNN) policy has been developed to improve upon the task-allocation methodology of the nearest neighbour (NN) policy by allowing for the efficient operation of multi-vehicle fleets through its ability to divert vehicles in an on-line manner. This policy, at every event update, re-optimizes all vehicle-customer pairings, such that vehicles are always paired with the customer to which they are closest. Namely, vehicles that are already en-route to assigned customers can be diverted to others when up-to-date real-time information makes new vehicle-customer pairings more favorable. Furthermore, the DNN policy is further augmented to distribute vehicles about the environment in an anticipatory behaviour of customer demands to arrive, as well as to minimize the longest customer wait times through the addition of wait-time limits.

Simulation results clearly indicate the superiority of the proposed DNN policy to the NN policy, and its ability to account for city-like traffic with no or little effect on the customers’ mean system times.

5.1.2. Dynamic Re-Pairing Methodology

The dynamic re-pairing methodology (DRPM) is a generic task-allocation solution methodology for the on-line coordinated interception of a variable number of targets with high maneuverability by a dynamic team of robotic pursuers in minimum total time. The solution approach utilizes modular software architecture to allow for the simultaneous determination of the pairing solution and pursuer navigation commands, with a common data sharing point to ensure the most up-to-date information is available for real-time decision making. This information comprises the number and position of pursuers and targets in the environment determined from an external sensory system. The optimization of pursuer-target re-pairings is achieved by a search engine in an on-line mode using a rolling-horizon concept for variable search depth, which allows for near-optimal pairings to be determined within given time constraints. Subsequently, each pursuer executes its interception task independently and
autonomously, where the Advanced Predictive Guidance Law is used for the time-optimal motion planning to intercept highly-maneuverable targets. The Obstacle Avoidance Navigation Law is also implemented to ensure no collisions occur in the environment.

Simulations and experiments have confirmed the proposed methodology to be efficient in determining optimal pairings and coordinating pursuers’ motions. Due to the generic nature of the DRPM, it can be used for autonomous air, surface, water, or space vehicles.

5.2. Recommendations for Future Work

The above section clearly outlines the contributions of this Thesis to the development of on-line task-allocation methodologies; however, there are still several issues to be addressed. This section details potential areas for research to further expand the relevance of the two methodologies developed to the real-life situations they address.

5.2.1. Dynamic Nearest Neighbour Policy

5.2.1.1. Realistic City-Traffic Conditions

The policy presented in this Thesis looks at its performance in simplistic city-like traffic, in which the customer demands are increased and directionalized to represent rush hours and decreased to represent night-time traffic. However, the vehicles in these simulations are assumed to move at a constant and fixed velocity at all times of the day, which is rarely the case in city-like traffic conditions. The evaluation of the policy, and its ability to determine optimal vehicle-customer assignments, under true city-like conditions where vehicles travel at different velocities, at different times of the day and over different regions of the city would further demonstrate its superiority over existing policies. As a result, determining if the policy still holds when not assuming a constant and linear relationship between distance and time would be necessary.

5.2.1.2. Dynamically Varying Numbers of Vehicles

In this Thesis, vehicle drivers are assumed to work 24 hours a day for an infinite number of days with no down time. This is again not highly representative of a real-life taxi-cab dispatching environment, in which vehicles may break down, drivers require breaks, and where the day is broken down into multiple shifts during which vehicles transition drivers. To verify its superiority over the NN policy under such situations, the policy would need to be modified to
include the ability to determine optimal vehicle-customer assignments when the number of vehicles varies dynamically and stochastically over the course of the day.

5.2.2. Dynamic Re-Pairing Methodology

5.2.2.1. Interception and Target Types Addressed
In this Thesis, the types of interception addressed were disengagement, where the pursuer would disengage the incapacitated target and continue to its next allocated target, or consumption, where the pursuer and target would be incapacitated post-interception. Additionally, only highly-maneuverable targets were addressed. In order to extend the DRPM to real-life combat applications, coordinated interception and evading targets may need to be addressed.

For this methodology to be functional in such applications, a method with which to anticipate the behaviour and resulting motion-trajectories of evading targets would need to be incorporated into the DRPM to account for this in the determination of pairings. Additionally, to address coordinated interception the navigation guidance laws used in the DRPM would need to be augmented to focus on intercepting an assigned target at a specific time, instead of in minimal time.

5.2.2.2. Dynamic and Realistic Environment
In order to assess the DRPM’s applicability to truly dynamic environments, the policy would need to be evaluated in a real-life setting with moving and stationary obstacles in addition to all the pursuers and targets. This incorporation of a planning mechanism in the navigation algorithm to avoid such obstacles, as well as a possible augmentation of the interception time calculation to account for the additional time required to avoid the obstacles, would be necessary. Additionally, depending on the application, whether it be surface, water- or air-borne, the ability to traverse the area between the pursuer and target (be it due to varying terrain topography, weather conditions, or no-fly zones) could additionally be accounted for. Due to the modularity of the DRPM, such additions could be made to the Navigation module, and appropriate interception time metric, without affecting the functionality of the rest of the methodology.

5.2.2.3. Hardware and Software Issues
The system implemented in these experiments had variable refresh rates across the different modules, with the slowest refresh rate being that of determining the navigation commands for the
robots, at approximately five times per second. However, this time could be significantly decreased through upgrading the development environment from Visual Studio 6.0 to one which could allow for more threads to run in parallel. This would allow each pursuer robot to operate on its own thread instead of one thread for all pursuers, and remove this bottleneck from the system.

Additionally, the Image-Acquisition module implementation could be optimized. Firstly, the CCD camera used had a resolution of 640x480 pixels, and limited contrast adjustment. The use of a better camera with greater resolution would allow for a more accurate determination of pursuer/target locations, and minimize the amount of noise in the system. Additionally, the vision-processing algorithm used to extract the position information from the image is computationally intensive as each pixel in the image must be analysed a minimum of one time, up to a maximum of the number of pattern markers on all pursuers (for this case six) times to determine the positions of the pursuers. A less computationally intensive algorithm would greatly decrease the processing time to determine the positions, and allow for more frequent pursuer-target pairings to be generated.

5.2.2.4. **Implementation**

In this Thesis, the algorithm has been verified to be functional for surface vehicles. However, while it has been developed to be applicable to surface, water, air, and space vehicles, it has not been validated to be functional for these. In order to gauge its efficacy and optimality in such applications, it would be beneficial to test it on such different vehicles through the modification of the motion controller.

5.3. **Final Concluding Statement**

In conclusion, this Thesis presents two novel on-line task-allocation methodologies, one for the dial-a-ride problem, and the other for the multi-target interception problem. The Dynamic Nearest Neighbour Policy improves upon the shortcomings of the Nearest Neighbour policy by allowing for a dynamic re-allocation of all vehicles to customers as new information, and more optimal pairings, become available. The Dynamic Re-Pairing Methodology, with its modular system architecture, allows users to adapt the methodology to any multi-target interception application at hand. The frequent re-optimization of the pursuer-target pairings in the DRPM allows for the pairings to adapt to changes in the numbers and motions of the pursuers and
targets in the system. Future work in this task-allocation research area could aid in extending the applicability of these methodologies to real-life, and real-time, dispatching and interception applications.
References


http://www.neuro.sfc.keio.ac.jp/~aly/polygon/info/color-space-faq.html


http://www.robotshop.com/ca/content/PDF/Miabot-Pro-User-Manual.pdf
Appendix A: Complete DNN Policy Algorithm

If Event A occurs: A new customer appears

(1) (a) If an idle, VSassigned or assigned vehicle which is not paired with a priority customer exists, determine the closest idle, VSassigned or assigned vehicle that is strictly closer (not equidistant) to the new customer than the customer to which it is currently assigned:

(i) If this vehicle is an assigned vehicle, unassign it from its previously assigned customer and proceed to Step 2;

(ii) If this vehicle is an idle vehicle, proceed to Step 3.

(b) If assigned vehicles exist, but none are strictly closer to the new customer than their currently assigned customer, proceed to Step 4.

(c) If all vehicles are busy, proceed to Step 4.

(2) Assign the vehicle to the new customer. The new customer becomes an assigned customer and the previously assigned customer becomes unassigned. Return to Step 1, and treat the now unassigned customer as a new customer.

(3) Assign the vehicle to the new customer. The vehicle and the new customer become an assigned vehicle and an assigned customer, respectively. Exit algorithm and await new trigger.

(4) The new customer is denoted as an unassigned customer and starts waiting in a queue. Exit algorithm and await new trigger.

If Event B occurs: A vehicle becomes idle

(5) (a) If an assigned or unassigned customer exists, determine the closest unassigned customer or assigned non-priority customer whose vehicle is strictly farther away (not equidistant) than the currently idle vehicle:

(i) If the customer is an assigned customer, unassign them from their previously assigned vehicle and proceed to Step 6;

(ii) If the customer is an unassigned customer, proceed to Step 7.

(b) If assigned customers exist, but none are strictly farther away from their currently assigned vehicle than the idle vehicle, proceed to Step 8.

(c) If there are only customers-in-service, proceed to Step 8.

(6) Assign the newly idle vehicle to the assigned customer. The idle vehicle becomes an assigned vehicle and the previously assigned vehicle becomes unassigned. Return to Step 5, and treat the unassigned vehicle as a new idle vehicle.
Appendix A: Complete DNN Policy Algorithm

(7) Assign the newly idle vehicle to the unassigned customer. The vehicle and the new customer become an assigned vehicle and an assigned customer, respectively. Exit algorithm and await new trigger.

(8) (a) If there is no anticipatory behaviour, keep the vehicle as idle at its current location. Exit algorithm and await new trigger.

(b) If customer demand is uniform across $\mathcal{A}$, assign the idle vehicle to the closest vehicle station that has the least amount of vehicles assigned to it, and the idle vehicle becomes a VSassigned vehicle. Exit algorithm and await new trigger.

(c) If customer demand is not-uniform across $\mathcal{A}$, and the vehicle’s drop-off was in the high-demand area, assign the idle vehicle to the closest vehicle station in that area that has the least amount of vehicles assigned to it, and the idle vehicle becomes a VSassigned vehicle. Exit algorithm and await new trigger.

(d) If customer demand is not-uniform across $\mathcal{A}$, and the vehicle’s drop-off was in the low-demand area:

(i) If a random probability generated is less than $P_r$ assign the idle vehicle to the closest vehicle station in the high-demand area that has the least amount of vehicles assigned to it, and the idle vehicle becomes a VSassigned vehicle. Exit algorithm and await new trigger.

(ii) If a random probability generated is greater than $P_r$ assign the idle vehicle to the closest vehicle station in the low-demand area that has the least amount of vehicles assigned to it, and the idle vehicle becomes a VSassigned vehicle. Exit algorithm and await new trigger.

If Event C occurs: A vehicle arrives at its assigned customer’s pickup location

(9) The assigned vehicle becomes a busy vehicle, and the assigned customer becomes a customer-in-service. Exit algorithm and await new trigger.

If Event D occurs: A customer reaches the wait-time limit

(10) Assign the priority customer to the vehicle which is closest to it. The new customer becomes an assigned priority-customer, the vehicle assigned with priority and the previously assigned customer becomes unassigned. Return to Step 1, and treat the now unassigned customer as a new customer.
Appendix B: Blackboard Example

In order to demonstrate how a blackboard is utilized, the following simplified example is discussed below for a 2-pursuer, 3-target scenario, Table B.1. Due to space restrictions only the x and y components of the pursuer and target position will be demonstrated for P and T.

- Iteration 1 signifies the first time any information has been written onto the blackboard. The initial positions of the pursuers and targets are written in their respective data structures. Since the interception status is set to FALSE, the Task-Allocation Module determines the appropriate pairings and writes them onto the blackboard.

- At Iteration 2, since there are still not-intercepted targets remaining in the workspace, the term FALSE is re-written as the interception status variable, while the new positions of the pursuers and targets have over-written those from Iteration 1. The appropriate pairings are determined, which are the same as for Iteration 1, and are re-written to the Pairings Matrix data-structure on the blackboard. Although there appears to be no change in the Pairings Matrix as the numerical values do not change, the data is entirely re-written at each update.

Table B.1. Blackboard Data Over-Writing Example.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Interception Status</th>
<th>Pursuer no.</th>
<th>x (mm)</th>
<th>y (mm)</th>
<th>Target no.</th>
<th>x (mm)</th>
<th>y (mm)</th>
<th>Pursuer no.</th>
<th>Target no.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FALSE</td>
<td>1</td>
<td>200</td>
<td>210</td>
<td>1</td>
<td>300</td>
<td>200</td>
<td>1</td>
<td>1 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>1200</td>
<td>650</td>
<td>2</td>
<td>600</td>
<td>800</td>
<td>2</td>
<td>0 0 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>900</td>
<td>800</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>FALSE</td>
<td>1</td>
<td>210</td>
<td>210</td>
<td>1</td>
<td>290</td>
<td>205</td>
<td>1</td>
<td>1 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>1190</td>
<td>660</td>
<td>2</td>
<td>610</td>
<td>800</td>
<td>2</td>
<td>0 0 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>910</td>
<td>802</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>FALSE</td>
<td>1</td>
<td>260</td>
<td>265</td>
<td>2</td>
<td>670</td>
<td>800</td>
<td>1</td>
<td>1 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>1100</td>
<td>750</td>
<td>3</td>
<td>970</td>
<td>840</td>
<td>2</td>
<td>0 1</td>
</tr>
</tbody>
</table>

- This process continues with content being re-written at each update (iteration) as demonstrated without any change to the size of the structures on the blackboard until Iteration 7, where it is detected that Target 1 has been intercepted by Pursuer 1. The interception status is noted as FALSE as there are still two targets remaining, however,
both the T and X data structures are re-sized to account for the change in n. Pairings for these two targets and the two pursuers are then determined and written to the re-sized X.

This process of over-writing the content of the data-structures, and resizing of P, T, and X as appropriate, is executed by the blackboard at each update interval, so that only the most current information is available to the accessing modules. While this example updates the blackboard in terms of iterations, it should be noted that different modules write onto the blackboard at different update rates.
Appendix C: Kinematic Model of the Mobile Robot

The kinematic model for a differentially-driven wheeled mobile robot presented in [56] is used as in the motion controller for these experiments, and is presented herein for completeness.

The kinematic model of the robots selected as the basis for this work is:

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\lambda}
\end{bmatrix} =
\begin{bmatrix}
\cos \lambda & 0 \\
\sin \lambda & 0 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
v \\
\omega
\end{bmatrix}, \text{ and } u = \begin{bmatrix} v & \omega \end{bmatrix}^{T}
\]

where \( y \) is the system state; the robot is located at \((x_c, y_c)\) turning to the right; \( \lambda \) is the robot heading angle with respect to the \( X \)-axis; and, the control \( u \) consists of the linear velocity \( v \) and the angular velocity \( \omega \). Although in (C.1) the controls of the mobile robot are its linear and angular velocities, the actual commands provided to the vehicle are the right and left wheel velocities, Figure C.1.

![Figure C.1: Wheel Velocities.](image)

Let \( v_l, v_r, \) and \( v_R \) represent the velocities of the left wheel, the right wheel, and the robot, respectively. Also, let \( d \) be the distance between the two wheels and \( D \) be the distance between the right wheel and the Instantaneous Center of Curvature, \( ICC \). The commands generated by the navigation-guidance/obstacle-avoidance algorithm set the linear velocity, \( v_P \), and the angular velocity, \( \omega_P \), of the pursuer:
Appendix C: Kinematic Model of the Mobile Robot

\[ \omega_p = \frac{v_p}{d^2 + D} \]  \hspace{1cm} (C.2)

The motion commands are executed by specifying \( v_l \) and \( v_r \). With \( d \) known, it is possible to determine \( R \), the turning radius of curvature of the robot, as the distance between the centre of the robot and ICC. The wheel velocities are determined using the kinematics Equations (C.3) to (C.5) given below,

\[ v_p = \frac{v_l + v_r}{2} \]  \hspace{1cm} (C.3)

\[ \frac{v_r}{D} = \frac{v_l}{D + d}, \text{ and} \]  \hspace{1cm} (C.4)

\[ v_l = r_{wh} \omega_l, \hspace{1cm} v_r = r_{wh} \omega_r, \]  \hspace{1cm} (C.5)

where \( r_{wh} \) is the radius of the wheel, and \( \omega_l \) and \( \omega_r \) are the angular velocities of the left and right wheel, respectively.

The final velocity for the next time instant, as computed by the proposed algorithm, must be one that is achievable by the robot. Therefore, given a *Feasible Velocity Region* (FVR) representing all the velocities achievable by the robot within \( \Delta t \), taking into account the kinematic and dynamic constraints on the robot, Figure C.2, the set of *Feasible Velocities* (FV) is given by:

\[ FV(t_i + \Delta t) = \{ v \mid v = v_p(t_i) \oplus \Delta t \cdot FA(t_i) \} \]  \hspace{1cm} (C.6)

where \( FA(t_i) \) represents the set of feasible accelerations of the robot at time \( t_i \). The FVR polygon is computed by scaling \( FA(t_i) \) by \( \Delta t \) and adding it to the current velocity of the pursuer \( v_p \). Thus, by ensuring that the algorithm only generates accelerations that are within the FVR for each time instant, the non-holonomic constraints of the mobile robot are satisfied.
Figure C.2: Feasible Accelerations.