Comparison of multi-robot task allocation algorithms

(Förderkennzeichen: EC grant 731848 ROPOD)

Ángela Patricia Enríquez Gómez

Publisher: Dean Prof. Dr. Wolfgang Heiden

Hochschule Bonn-Rhein-Sieg – University of Applied Sciences, Department of Computer Science

Sankt Augustin, Germany

December 2019

Technical Report 02-2019
This work was supervised by
Prof. Dr. Erwin Prassler
M. Sc. Argentina Ortega Sáinz

ROPOD is an Innovation Action
funded by
the European Commission
under grant no. 731848
within
the Horizon 2020 framework
program.

Copyright © 2019, by the author(s). All rights reserved. Permission to make
digital or hard copies of all or part of this work for personal or classroom use is
granted without fee provided that copies are not made or distributed for profit or
commercial advantage and that copies bear this notice and the full citation on the
first page. To copy otherwise, to republish, to post on servers or to redistribute to
lists, requires prior specific permission.

Das Urheberrecht des Autors bzw. der Autoren ist unveräußerlich. Das
Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Das Werk kann
innerhalb der engen Grenzen des Urheberrechtsgesetzes (UrhG), German copyright
law, genutzt werden. Jede weitergehende Nutzung regelt obiger englischsprachiger
Copyright-Vermerk. Die Nutzung des Werkes außerhalb des UrhG und des obigen
Copyright-Vermerks ist unzulässig und strafbar.

Digital Object Identifier doi:10.18418/978-3-96043-075-9
DOI-Resolver http://dx.doi.org/
Abstract

Multi-robot systems (MRS) are capable of performing a set of tasks by dividing them among the robots in the fleet. One of the challenges of working with multi-robot systems is deciding which robot should execute each task. Multi-robot task allocation (MRTA) algorithms address this problem by explicitly assigning tasks to robots with the goal of maximizing the overall performance of the system. The indoor transportation of goods is a practical application of multi-robot systems in the area of logistics. The ROPOD project works on developing multi-robot system solutions for logistics in hospital facilities. The correct selection of an MRTA algorithm is crucial for enhancing transportation tasks. Several multi-robot task allocation algorithms exist in the literature, but just few experimental comparative analysis have been performed. This project analyzes and assesses the performance of MRTA algorithms for allocating supply cart transportation tasks to a fleet of robots. We conducted a qualitative analysis of MRTA algorithms, selected the most suitable ones based on the ROPOD requirements, implemented four of them (MURDOCH, SSI, TeSSI, and TeSSIduo), and evaluated the quality of their allocations using a common experimental setup and 10 experiments. Our experiments include off-line and semi on-line allocation of tasks as well as scalability tests and use virtual robots implemented as Docker containers. This design should facilitate deployment of the system on the physical robots. Our experiments conclude that TeSSI and TeSSIduo suit best the ROPOD requirements. Both use temporal constraints to build task schedules and run in polynomial time, which allow them to scale well with the number of tasks and robots. TeSSI distributes the tasks among more robots in the fleet, while TeSSIduo tends to use a lower percentage of the available robots. Subsequently, we have integrated TeSSI and TeSSIduo to perform multi-robot task allocation for the ROPOD project.
Acknowledgements

Thanks to my supervisors Prof. Dr. Erwin Prassler and M. Sc. Argentina Ortega for their guidance and support during the realization of this project. I am grateful for their advice and readiness to help whenever I faced difficulties. Thanks for our constant meetings and discussions, which steered this project in the right direction and lead it to achieve its goals. I very much appreciate the opportunity to be part of a project like ROPOD. This experience has allowed me to observe the development of a research project from the inside and has shown me the rewards and difficulties that such an endeavor entails. I have not only learned about multi-robot systems but also how to work in a more structured and scientific way. I am looking forward to continuing the work presented here, and to continuing learning and contributing to the ROPOD project.

A special thanks to my parents, who support me from my beautiful and distant Mexico. They have always encouraged me to be a better version of myself and to enrich my life through education and academia. Thank you for pushing me to pursue my goals, even if it means that an ocean separates us, you are always close to my heart. Thousand thanks to Tim, for making me feel at home. Thank you for your loving support and motivation, and for giving me the emotional strength that carries me on this journey.
Contents

1 Introduction .................................................. 1
  1.1 MRTA Taxonomies ............................................. 2
    1.1.1 Gerkey and Matarić’s Taxonomy ............................ 2
    1.1.2 iTax Taxonomy ............................................ 4
    1.1.3 MRTA-TOC Taxonomy ....................................... 6
  1.2 Weakly-Cooperative Versus Tightly-Cooperative Solutions ....... 7
  1.3 Task Decomposition .......................................... 9
  1.4 Single-Task Single-Robot Instantaneous Assignment (ST-SR-IA) .. 10
  1.5 Single-Task Single-Robot Time Extended Assignment (ST-SR-TA) .. 11
  1.6 Motivation .................................................. 11
  1.7 Challenges and Difficulties ................................... 12
  1.8 Problem Statement ........................................... 12

2 State of the Art ............................................... 15
  2.1 Optimization Problem ....................................... 15
  2.2 MRTA Related Problems ..................................... 16
  2.3 MRTA Topologies ............................................ 18

3 Qualitative Comparison ........................................ 25
  3.1 Comparison Criteria ......................................... 25
  3.2 ST-SR-IA Algorithms ......................................... 26
    3.2.1 MURDOCH ............................................... 26
    3.2.2 ALLIANCE ............................................... 30
    3.2.3 Consensus Based Parallel Auction and Execution (CBPAE) .. 32
  3.3 ST-SR-TA algorithms ......................................... 35
    3.3.1 Single-Round Combinatorial Auctions ....................... 36
4 Methodology

4.1 Use Case: Transportation of Supply Carts in a Hospital

4.2 Setup

4.2.1 Hardware

4.2.2 Software

4.3 Performance Metrics

4.4 Experimental Design

4.4.1 Assumptions

4.4.2 Type of Experiments

4.5 Datasets

4.5.1 Spatial Information

4.5.2 Temporal Information

4.5.3 Batched Datasets

4.5.4 Datasets Description

5 Solution

5.1 Task Allocator Implementation

5.1.1 Components Description:

5.1.2 Limitations

5.2 Docker Implementation

5.2.1 Differences Between the Docker Implementation and the Task Allocator Implementation

5.2.2 Limitations

5.3 Configuration Files

5.3.1 Config Folder

5.3.2 Config_Scalability Folder
5.4 Experimental Workflow ........................................ 85
   5.4.1 Task Allocator Implementation .......................... 85
   5.4.2 Docker Implementation .................................. 88
   5.4.3 UML Diagrams ............................................. 91
5.5 ROPOD Integration ............................................. 92

6 Results .................................................................. 105
   6.1 Experiment 1: Off-line Allocation of Tasks Uniformly Distributed in the Map ........................................ 105
      6.1.1 Purpose of the Experiment .............................. 105
      6.1.2 Experimental Design Considerations ................. 105
      6.1.3 Hypothesis ................................................ 106
      6.1.4 Results .................................................... 106
      6.1.5 Analysis of Results ...................................... 112
      6.1.6 Conclusions .............................................. 114
   6.2 Experiment 2: Off-line Allocation of Tasks Clustered in the Map ........................................ 115
      6.2.1 Purpose of the Experiment .............................. 115
      6.2.2 Experimental Design Considerations ................. 115
      6.2.3 Hypothesis ................................................ 116
      6.2.4 Results .................................................... 116
      6.2.5 Analysis of Results ...................................... 122
      6.2.6 Conclusions .............................................. 123
   6.3 Experiment 3: On-line Allocation of Tasks Uniformly Distributed in the Map ........................................ 124
      6.3.1 Purpose of the Experiment .............................. 124
      6.3.2 Experimental Design Considerations ................. 124
      6.3.3 Hypothesis ................................................ 125
      6.3.4 Results .................................................... 125
      6.3.5 Analysis of Results ...................................... 134
      6.3.6 Conclusions .............................................. 135
   6.4 Experiment 4: On-line Allocation of Tasks Clustered in the Map ........................................ 136
      6.4.1 Purpose of the Experiment .............................. 136
      6.4.2 Experimental Design Considerations ................. 136
6.4.3 Hypothesis ................................................. 137
6.4.4 Results .................................................. 137
6.4.5 Analysis of Results ................................. 146
6.4.6 Conclusions ............................................. 147

6.5 Experiment 5: Off-Line Allocation of Tasks Uniformly Distributed in Time and Space .............................................. 147
6.5.1 Purpose of the Experiment ......................... 147
6.5.2 Experimental Design Considerations .............. 147
6.5.3 Hypothesis ............................................... 148
6.5.4 Results .................................................. 148
6.5.5 Analysis of Results ...................................... 154
6.5.6 Conclusions ............................................. 155

6.6 Experiment 6: Off-line Allocation of Tasks Clustered in Time and Space ............................................. 156
6.6.1 Purpose of the Experiment ........................... 156
6.6.2 Experimental Design Considerations .............. 156
6.6.3 Hypothesis ............................................... 156
6.6.4 Results .................................................. 157
6.6.5 Analysis of Results ...................................... 162
6.6.6 Conclusions ............................................. 163

6.7 Experiment 7: Off-line Allocation of Increasing Number of Tasks Uniformly Distributed in Time and Space ............................................. 164
6.7.1 Purpose of the Experiment ......................... 164
6.7.2 Experimental Design Considerations .............. 164
6.7.3 Hypothesis ............................................... 165
6.7.4 Results .................................................. 165
6.7.5 Analysis of Results ...................................... 167
6.7.6 Conclusions ............................................. 169

6.8 Experiment 8: Off-line Allocation of Increasing Number of Tasks Clustered in Time and Space ............................................. 170
6.8.1 Purpose of the Experiment ......................... 170
6.8.2 Experimental Design Considerations .............. 170
6.8.3 Hypothesis ............................................... 171
6.8.4 Results .................................................. 171
6.8.5 Analysis of Results ........................................... 173
6.8.6 Conclusions .................................................... 175
6.9 Experiment 9: Off-line Allocation of Tasks Uniformly Distributed in Time with Increasing Number of Robots. ......................... 175
   6.9.1 Purpose of the Experiment ............................. 175
   6.9.2 Experimental Design Considerations .................. 175
   6.9.3 Hypothesis ................................................. 176
   6.9.4 Results .................................................... 176
   6.9.5 Analysis of Results ...................................... 178
   6.9.6 Conclusions .............................................. 180
6.10 Experiment 10: Off-line Allocation of Tasks Clustered in Time and Space with Increasing Number of Robots ...................... 181
   6.10.1 Purpose of the Experiment ............................. 181
   6.10.2 Experimental Design Considerations .................. 181
   6.10.3 Hypothesis ................................................. 182
   6.10.4 Results .................................................... 182
   6.10.5 Analysis of Results ...................................... 184
   6.10.6 Conclusions .............................................. 186
6.11 Additional Findings ............................................ 187

7 Conclusions .................................................... 191
   7.1 Contributions ............................................... 193
   7.2 Lessons Learned ............................................. 194
   7.3 Future Work ................................................ 194

Appendix A Installation and Setup ........................................ 197
   A.1 Installation ................................................ 197
      A.1.1 Task Allocator Implementation ...................... 197
      A.1.2 Docker Implementation ............................. 198
   A.2 Creating Configuration Diles and Datasets ............... 198
      A.2.1 Experiments 1 to 9 .................................. 199
      A.2.2 Experiments 9 and 10 ............................... 199
   A.3 Instructions for Running the Experiments ................. 200
A.3.1 Task Allocation Implementation ........................................... 200
A.3.2 Docker Implementation ................................................... 201

Appendix B Datasets ................................................................. 203
B.1 Spatial Uniformly Distributed Tasks (SDU) ................................. 203
  B.1.1 SDU-TER ..................................................................... 203
  B.1.2 SDU-TGR .................................................................... 206
B.2 Spatial Clustered Tasks (SDC) .................................................. 210
  B.2.1 SDC-TER ..................................................................... 210
  B.2.2 SDC-TGR ..................................................................... 212
B.3 Temporal Uniformly Distributed Tasks (TDU) ............................... 216
  B.3.1 TDU-TGR ..................................................................... 216
  B.3.2 TDU-ST ....................................................................... 218
  B.3.3 TDU-SR ....................................................................... 219
B.4 Temporal Clustered Tasks (TDC) ................................................ 220
  B.4.1 TDC-TGR ..................................................................... 220
  B.4.2 TDC-ST ....................................................................... 222
  B.4.3 TDC-SR ....................................................................... 223

Appendix C Performance Metrics ................................................... 225

References ................................................................................. 237
# List of Figures

1.1 MRTA Taxonomy by Gerkey and Matarić, 2004. ......................... 3
1.2 Degree of interdependence of robot-task utilities. The circles represent tasks and the solid lines the routes of the robots. The arrows indicate constraints between tasks [20]. ............................................. 5
1.3 iTax as a combination of two levels. .................................... 6
1.4 iTax complete taxonomy [20]. ............................................. 7
1.5 MRTA-TOC taxonomy. .................................................. 8
2.1 MRTA approaches based on their topology. ............................ 19
3.1 Simple Temporal Network of a robot with three allocated tasks [25]. 43
4.1 Robot initial positions. .................................................. 57
4.2 Dataset of type SDU-TER. ............................................. 70
4.3 Dataset of type SDU-TGR. ............................................. 71
4.4 Dataset of type SDC-TER. ............................................. 72
4.5 Dataset of type SDC-TGR. ............................................. 73
4.6 Dataset of type TDU-TGR. ............................................. 74
4.7 Dataset of type TDU-ST. ............................................. 74
4.8 Dataset of type TDU-SR. ............................................. 75
4.9 Dataset of type TDC-TGR. ............................................. 75
4.10 Dataset of type TDC-ST. ............................................. 76
4.11 Dataset of type TDC-SR. ............................................. 76
5.1 UML Class Diagram for the Task Allocation Implementation. ...... 97
5.2 UML Class Diagram for the Docker Implementation. ............... 98
5.3 MURDOCH UML Class Diagram. ..................................... 99
5.4 SSI UML Class Diagram. ............................................. 100
5.5 TeSSI and TeSSIduo UML Class Diagram. .......................... 101
5.6 UML Sequence Diagram for MURDOCH in the Docker implementation. 102
5.7 UML Sequence Diagram for SSI in the Docker implementation. ....... 103
5.8 UML Sequence Diagram for TeSSI and TeSSIduo in the Docker implementation. .................................................. 104

6.1 Experiment 1: Number of allocations and messages sent and received. 106
6.2 Experiment 1: Travel distances and makespan of the fleet. .......... 107
6.3 Experiment 1: Temporal distribution of tasks for dataset SDU-TER-1. 107
6.4 Robot trajectories for dataset SDU-TER-1. Each rectangle represents a robot and the dots represent pickup and delivery locations. .... 108
6.5 Robot trajectories for dataset SDU-TER-2. Each rectangle represents a robot and the dots represent pickup and delivery locations. .... 109
6.6 Robot trajectories for dataset SDU-TER-3. Each rectangle represents a robot and the dots represent pickup and delivery locations. .... 110
6.7 Robot trajectories for dataset SDU-TER-4. Each rectangle represents a robot and the dots represent pickup and delivery locations. .... 111
6.8 Robot trajectories for dataset SDU-TER-5. Each rectangle represents a robot and the dots represent pickup and delivery locations. .... 112
6.9 Experiment 2: Number of allocations and messages sent and received. 116
6.10 Experiment 2: Travel distances and makespan of the fleet. ......... 117
6.11 Experiment 2: Temporal distribution of tasks for dataset SDC-TER-CR-1. .................. 117
6.12 Robot trajectories for dataset SDC-TER-CR-1. Each rectangle represents a robot and the dots represent pickup and delivery locations. .................................................. 118
6.13 Robot trajectories for dataset SDC-TER-CR-2. Each rectangle represents a robot and the dots represent pickup and delivery locations. ... 119
6.14 Robot trajectories for dataset SDC-TER-CR-3. Each rectangle represents a robot and the dots represent pickup and delivery locations. .................................................. 120

6.16 Experiment 3: Number of allocations for task batches of sizes 1, 2 and 4.

6.17 Experiment 3: Number of messages sent and received for task batches of sizes 1, 2 and 4.

6.18 Experiment 3: Sum of the distances that the robots traveled to executed their tasks for batches of sizes 1, 2 and 4.

6.19 Experiment 3: Time the fleet needed for executing all tasks for batches of sizes 1, 2 and 4.

6.20 Experiment 3: Temporal distribution of tasks for dataset SDU-TGR-

1_batch_size1.

6.21 Experiment 3: Temporal distribution of tasks for dataset SDU-TGR-1_batch_size2.


6.23 Robot trajectories for dataset SDU-TGR-1_batch_size1. Each rectangle represents a robot and the dots represent pickup and delivery locations.

6.24 Robot trajectories for dataset SDU-TGR-1_batch_size2. Each rectangle represents a robot and the dots represent pickup and delivery locations.

6.25 Robot trajectories for dataset SDU-TGR-1_batch_size4. Each rectangle represents a robot and the dots represent pickup and delivery locations.

6.26 Experiment 4: Number of allocations for task batches of sizes 1, 2 and 4.

6.27 Experiment 4: Number of messages sent and received for task batches of sizes 1, 2 and 4.

6.28 Experiment 4: Sum of the distances that the robots traveled to executed their tasks for batches of sizes 1, 2 and 4.
6.29 Experiment 4: Time the fleet needed for executing all tasks for batches of sizes 1, 2 and 4. ........................................ 141
6.30 Experiment 4: Temporal distribution of tasks for dataset SDC-TGR-CR-1_batch_size1. ........................................ 142
6.31 Experiment 4: Temporal distribution of tasks for dataset SDC-TGR-CR-1_batch_size2. ........................................ 142
6.32 Experiment 4: Temporal distribution of tasks for dataset SDC-TGR-CR-1_batch_size4. ........................................ 142
6.33 Robot trajectories for dataset SDC-TGR-CR-1_batch_size1. Each rectangle represents a robot and the dots represent pickup and delivery locations. ........................................ 143
6.34 Robot trajectories for dataset SDC-TGR-CR-1_batch_size2. Each rectangle represents a robot and the dots represent pickup and delivery locations. ........................................ 144
6.35 Robot trajectories for dataset SDC-TGR-CR-1_batch_size4. Each rectangle represents a robot and the dots represent pickup and delivery locations. ........................................ 145
6.36 Experiment 5: Number of allocations and messages sent and received. 148
6.37 Experiment 5: Travel distances and makespan of the fleet. .......... 149
6.38 Experiment 5: Temporal distribution of tasks for dataset TDU-TGR-1. 149
6.39 Robot trajectories for dataset TDU-TGR-1. Each rectangle represents a robot and the dots represent pickup and delivery locations. ...... 150
6.40 Robot trajectories for dataset TDU-TGR-2. Each rectangle represents a robot and the dots represent pickup and delivery locations. ...... 151
6.41 Robot trajectories for dataset TDU-TGR-3. Each rectangle represents a robot and the dots represent pickup and delivery locations. ...... 152
6.42 Robot trajectories for dataset TDU-TGR-4. Each rectangle represents a robot and the dots represent pickup and delivery locations. ...... 153
6.43 Robot trajectories for dataset TDU-TGR-5. Each rectangle represents a robot and the dots represent pickup and delivery locations. ...... 154
6.44 Experiment 6: Number of allocations and messages sent and received. 157
6.45 Experiment 6: Travel distances and makespan of the fleet. .......... 158
6.46 Experiment 6: Temporal distribution of tasks for dataset TDC-TGR-ITW-1. ........................................ 158
6.47 Robot trajectories for dataset TDC-TGR-ITW-1. Each rectangle represents a robot and the dots represent pickup and delivery locations. 159
6.48 Robot trajectories for dataset TDC-TGR-ITW-2. Each rectangle represents a robot and the dots represent pickup and delivery locations. 160
6.49 Robot trajectories for dataset TDC-TGR-ITW-3. Each rectangle represents a robot and the dots represent pickup and delivery locations. 161
6.50 Robot trajectories for dataset TDC-TGR-ITW-4. Each rectangle represents a robot and the dots represent pickup and delivery locations. 162
6.51 Experiment 7: Number of successful and unsuccessful allocations. . 165
6.52 Experiment 7: Number of messages sent and received by the auctioneer. 166
6.53 Experiment 7: Distances that the robots will travel to execute their tasks. ................................................. 166
6.54 Experiment 7: Time the fleet will take to execute all tasks. .......... 166
6.55 Experiment 7: TeSSI temporal distribution of tasks per robot for dataset TDU-ST-100. .................................. 167
6.56 Experiment 7: TeSSI duo temporal distribution of tasks per robot for dataset TDU-ST-100. ................................. 167
6.57 Experiment 7: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time. ............................... 168
6.58 Experiment 8: Number of successful and unsuccessful allocations. . 171
6.59 Experiment 8: Number of messages sent and received by the auctioneer. 172
6.60 Experiment 8: Distances that the robots will travel to execute their tasks. .................................................. 172
6.61 Experiment 8: Time the fleet will take to execute all tasks. .......... 172
6.62 Experiment 8: TeSSI temporal distribution of tasks per robot for dataset TDC-ST-100. .................................. 173
6.63 Experiment 8: TeSSI duo temporal distribution of tasks per robot for dataset TDC-ST-100. ................................. 173
6.64 Experiment 8: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time. ............................... 174
6.65 Experiment 9: Number of successful and unsuccessful allocations. . 176

xvii
6.66 Experiment 9: Number of messages sent and received by the auctioneer. 177
6.67 Experiment 9: Distances that the robots will travel to execute their tasks. ................................. 177
6.68 Experiment 9: Time the fleet will take to execute all tasks. ............ 177
6.69 Experiment 9: TeSSI temporal distribution of tasks per robot for a fleet of 20 robots. .................... 178
6.70 Experiment 9: TeSSI duo temporal distribution of tasks per robot for a fleet of 20 robots. .................. 178
6.71 Experiment 9: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time. .................. 179
6.72 Experiment 10: Number of successful and unsuccessful allocations. ... 182
6.73 Experiment 10: Number of messages sent and received by the auctioneer. 183
6.74 Experiment 10: Distances that the robots will travel to execute their tasks. ...................................... 183
6.75 Experiment 10: Time the fleet will take to execute all tasks. ............. 183
6.76 Experiment 10: TeSSI temporal distribution of tasks per robot for a fleet of 20 robots. ..................... 184
6.77 Experiment 10: TeSSI duo temporal distribution of tasks per robot for a fleet of 20 robots. .................. 184
6.78 Experiment 10: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time. .................. 185
6.79 Comparison of allocation time and total time using two methods for storing the STN. ......................... 188
6.80 Comparison of allocation time and total time using the two definitions of makespan. ......................... 189
6.81 Temporal distribution of tasks per robot using the two definitions of makespan. ............................ 190
6.82 Robot usage using the two definitions of makespan. ....................... 190

B.1 Dataset SDU-TER-1. ........................................... 203
B.2 Dataset SDU-TER-2. ........................................... 204
B.3 Dataset SDU-TER-3. ........................................... 204
B.4 Dataset SDU-TER-4. ........................................... 205
B.5 Dataset SDU-TER-5. .............................................. 205
B.6 Dataset SDU-TGR-1. .............................................. 206
B.7 Dataset SDU-TGR-2. .............................................. 207
B.8 Dataset SDU-TGR-3. .............................................. 208
B.9 Dataset SDU-TGR-4. .............................................. 209
B.10 Dataset SDU-TGR-5. ............................................. 210
B.11 Dataset SDC-TER-CR-1. ...................................... 211
B.12 Dataset SDC-TER-CR-2. ...................................... 211
B.13 Dataset SDC-TER-CR-3. ...................................... 212
B.14 Dataset SDC-TER-CR-4. ...................................... 212
B.15 Dataset SDC-TGR-CR-1. ...................................... 213
B.16 Dataset SDC-TGR-CR-2. ...................................... 214
B.17 Dataset SDC-TGR-CR-3. ...................................... 215
B.18 Dataset SDC-TGR-CR-4. ...................................... 216
B.19 Dataset TDU-TGR-1. ............................................. 217
B.20 Dataset TDU-TGR-2. ............................................. 217
B.21 Dataset TDU-TGR-3. ............................................. 218
B.22 Dataset TDU-TGR-4. ............................................. 218
B.23 Dataset TDU-TGR-5. ............................................. 219
B.24 Dataset TDU-ST-100. .......................................... 219
B.25 Dataset TDU-SR-100. .......................................... 220
B.26 Dataset TDC-TGR-ITW-1. .................................... 220
B.27 Dataset TDC-TGR-ITW-2. .................................... 221
B.28 Dataset TDC-TGR-ITW-3. .................................... 221
B.29 Dataset TDC-TGR-ITW-4. .................................... 222
B.30 Dataset TDC-ST-100. .......................................... 222
B.31 Dataset TDU-SR-100. .......................................... 223
# List of Tables

2.1 MRTA methods based on topology and approach. .......................... 23  
3.1 Comparison of selected algorithms. ........................................ 49  
4.1 Overview of experiments ...................................................... 56  
4.2 Algorithms tested. ............................................................. 58  
4.3 Off-line experiments. ........................................................... 69  
4.4 On-line experiments. ........................................................... 70  
5.1 Zyre messages used in the allocation process. ............................ 84  
6.1 Experiment 2: Difference on travel distance as the cluster radius increases. ................................................................. 117  
6.2 Experiment 3: Difference on travel distance as the batch size increases. Dataset: SDU-TGR-1. ......................................................... 125  
6.3 Experiment 4: Difference on travel distance as the cluster radius and batch size increases. ......................................................... 137  
6.4 Experiment 6: Difference on travel distance and makespan as the interval between time windows increases. ............................... 157  
C.3 Experiment 3. Performance metrics. Datasets: SDU-TGR-1_batch_size1, SDU-TGR-1_batch_size2, and SDU-TGR-1_batch_size3. .............. 228  
C.5 Experiment 5. Performance metrics. Datasets: TDU-TGR-1, TDU-
TGR-2, and TDU-TGR-3. .......................................................... 230
C.6 Experiment 6. Performance metrics. Datasets: TDC-TGR-ITW-1,
TDC-TGR-ITW-2, and TDC-TGR-ITW-3. ................................. 231
C.7 Experiment 7. Performance metrics. Datasets: TDU-ST-10, TDU-ST-
50, and TDU-ST-100. .............................................................. 232
C.8 Experiment 8. Performance metrics. Datasets: TDC-ST-10, TDC-ST-
50, and TDC-ST-100. .............................................................. 233
of robots: 10, 50, and 100. ..................................................... 234
C.10 Experiment 10. Performance metrics. Datasets: TDC-SR-100. Num-
ber of robots: 10, 50, and 100. .............................................. 235
Introduction

“Multi-robot systems (MRS) are a group of robots that are designed aiming to perform some collective behavior.” [18] Through their collective behavior, multi-robot systems can accomplish a set of tasks by dividing them among the robots in the group. The modularity of the system and the possibility of introducing robots with different capabilities into the fleet makes it possible to accomplish a variety of tasks that can range from simple to complicated. If a task is too complex for a single robot to handle, it can be jointly executed by a group of robots.

Instead of implementing expensive single-robot systems, with single points of failure, one can employ a fleet of simpler low-cost robots. Such a modular system is robust because, in case of robot failure, another robot in the fleet can take over. It is flexible because if more tasks are required, the size and/or heterogeneity of the group can increase; and it is efficient because more than one task can be executed simultaneously by different robots or groups of robots [18].

The problem of deciding which robot should execute each task is known as multi-robot task allocation (MRTA), and it is one of the challenges of working with multi-robot systems [18]. MRTA requires some form of multi-robot cooperation. There exist two main approaches to multi-robot cooperation, intentional cooperation, and emergent cooperation. Whereas in an intentional cooperation approach, robots cooperate explicitly with the purpose of achieving a common goal, in an emergent cooperation approach, such as swarm robotics, the cooperative behavior is a consequence of the robot’s interactions with each other and with the environment.
Emergent cooperation has been mainly applied to homogeneous groups of robots, and their solutions are problem specific, which makes them unsuitable for situations in which robots need to tackle a variety of problems [13]. Intentional cooperation for task allocation is more suitable for applications where robots are expected to perform different real-world tasks requested by humans [13]. In such scenarios, it might be needed to introduce new types of tasks, increase the number of robots required for one task, and even introduce heterogeneity to the fleet. It is thus desirable to approach such a problem from a perspective that allows for the flexibility required. Multi-robot task allocation (MRTA) uses intentional cooperation because it explicitly assigns tasks to robots with the goal of maximizing the overall performance of the system. MRTA is only responsible for the allocation of tasks. The execution of such tasks can be performed by any method deemed suitable.

Approaches for solving task allocation in multi-agent systems (MAS) cannot be directly applied to multi-robot systems (MRS). Unlike MAS, MRS operate in physical environments where the information acquired is incomplete and prone to errors [12].

1.1 MRTA Taxonomies

Before applying an MRTA algorithm, it is essential to identify the kind of problem to be solved. Based on the type of MRTA problem, a suitable algorithm can be selected.

1.1.1 Gerkey and Matarić’s Taxonomy

The taxonomy of task allocation in multi-robot systems introduced by Gerkey and Matarić in 2004 [15] classifies allocation problems based on three axes, namely the robot type, the task type, and the allocation type, as shown in Figure 1.1

- **Single-task robots** (ST). Robots execute at most one task at a time.
- **Multi-task robots** (MT). Robots execute multiple tasks at a time.
- **Single-robot tasks** (SR). Tasks require exactly one robot.
- **Multi-robot tasks** (MR). Tasks require multiple robots.
• **Instantaneous Assignment (IA).** No more than one task is assigned to a robot at a time, i.e., robots do not have a time schedule for allocating tasks to be performed in the future [11].

• **Time-extended Assignment (TA).** More than one task can be assigned to a robot. Either because there are, at present, more tasks than robots or because incoming future tasks are known or predicted. It can also be the case that only one robot has the capabilities for performing a set of tasks and hence, these tasks are allocated to it, while other robots receive just one task or remain idle. In all cases, a time schedule of tasks for each robot is built [20].

![MRTA Taxonomy by Gerkey and Matarić, 2004.](image)

MRTA problems can be described as a combination of the previous letters. For example, a problem is ST-SR-IA (Single-task, Single-robot, Instantaneous Assignment) if a robot that can perform only one task at a time is instantaneously assigned a task that requires exactly one robot. The taxonomy describes eight types of MRTA problems:

1. **ST-SR-IA.** Single-Task Single-Robot Instantaneous Assignment
3. **ST-MR-IA.** Single-Task Multi-Robot Instantaneous Assignment
4. **ST-MR-TA.** Single-Task Multi-Robot Time Extended Assignment
5. **MT-SR-IA.** Multi-Task Single-Robot Instantaneous Assignment
1.1. MRTA Taxonomies


7. **MT-MR-IA.** Multi-Task Multi-Robot Instantaneous Assignment

8. **MT-MR-TA.** Multi-Task Multi-Robot Time Extended Assignment

Gerkey and Matarić indicate that although their taxonomy covers many of the MRTA problems, there are some that are left out. Tasks with interrelated utilities and tasks with constraints are out of the scope of the taxonomy. Nonetheless, their taxonomy is extensively used in the literature to describe MRTA problems. A problem has interrelated utilities if the utility that a robot estimates for a task, depends on the utilities for other tasks; either from its own already allocated tasks or from the tasks allocated to other robots, i.e., these problems depend on task schedules. There are also problems with task constraints in which the execution of a task depends on the execution of another task. For example, tasks might need to have a specific order or be executed simultaneously [15].

1.1.2 iTax Taxonomy

Korsah, Dias and Stentz [20] proposed a taxonomy that encompasses problems with interrelated utilities and task constraints. Their taxonomy is called iTax, and extends the taxonomy of Gerkey and Matarić by introducing a new layer or level that describes the degree of interdependence of the robot’s utilities for a task. The terms used in the new layer are illustrated in Figure 1.2. Gerkey and Matarić’s taxonomy is kept to describe the problem configuration, and thus, MRTA problems can be described as a combination of terms from the previous taxonomy and the new one, as visualized in Figure 1.3:

- **No Dependencies (ND).** The utility of a robot for a task is independent of every other task or robot.

- **In-Schedule Dependencies (ID).** The utility of a robot for a task depends on the tasks already allocated to the robot. Problems with time-extended assignments fall into this category because a single robot builds a time-extended schedule to allocate its tasks. The allocation of new incoming tasks depends on the robot’s schedule.
Chapter 1. Introduction

• **Cross-Schedule Dependencies (XD).** The utility of a robot for a task depends on the tasks already allocated to the robot and on the tasks allocated to other robots in the system. Two common cases of cross-schedule dependencies are:

  – Different robots are allocated single-robot tasks with constraints such as the order in which they need to be executed, their proximity, and whether they need to be performed simultaneously.

  – A group of robots forms a coalition to jointly perform a multi-robot task.

• **Complex Dependencies (CD).** The utility of a robot for a task depends on the schedule of the other robots in the system, which depends on the way complex tasks are decomposed. That is, task decomposition and task allocation are executed simultaneously.

![Figure 1.2: Degree of interdependence of robot-task utilities. The circles represent tasks and the solid lines the routes of the robots. The arrows indicate constraints between tasks [20].](image)

In the iTax taxonomy, shown in Figure 1.4, fixing the first level to No Dependencies (ND) is theoretically equivalent to the taxonomy of Gerkey and Matarić. However, the authors of iTax do not include all kind of configuration problems of Gerkey and Matarić’s taxonomy in the ND category, because they consider that some of them represent problems with interrelated utilities. They claim that multi-task robot problems (MT) have in-schedule dependencies (ID) because the amount of tasks that can be assigned to a single robot depends on the physical and computational capabilities of the robot. The allocation of a task to a multi-task robot is only possible if the robot still has resources to perform the new task and this depends on its schedule. Likewise, multi-robot (MR) tasks cannot fall into the no dependencies (ND)
1.1. MRTA Taxonomies

1.1.1 MRTA Taxonomy

An ITax taxonomy is a combination of two levels. The category because the utility of a robot for performing a multi-robot task depends on the other robots it will be working with for accomplishing the task, and hence such problems have cross-schedule dependencies [20]. The only two problems in the iTax taxonomy that are considered to have independent utilities are ST-SR-IA (Single-Task, Single-Robot, Instantaneous Assignment) and ST-SR-TA (Single-Task, Single-Robot, Time Extended Assignment).

1.1.3 MRTA-TOC Taxonomy

MRTA-TOC [26] is a taxonomy that expands Time Extended assignment (TA) problems into problems with temporal constraints and problems with ordering constraints. Figure 1.5 illustrates the MRTA-TOC taxonomy.

Temporal constraints are expressed using a time window (TA:TW) which denotes the time interval during which a task should be executed. A constraint that cannot be violated is a hard constraint (TW-HC). A constraint that can be violated but which incurs a penalty is a soft constraint (TW-SC). Examples of temporal soft constraints include the case where deadlines are satisfied with some probability or where tasks can start early or finish late with some penalty.

Problems with ordering constraints can have either synchronization constraints or precedence constraints, both of which are expressed as (TA:SP). Synchronization constraints specify the time relationship in which tasks need to be executed. For
Chapter 1. Introduction

instance, a task must be executed 10 minutes before another task. Precedence constraints indicate the order in which tasks need to be executed.

1.2 Weakly-Cooperative Versus Tightly-Cooperative Solutions

Task solutions provided by MRTA can be either weakly-cooperative or tightly-cooperative. While a weakly-cooperative solution involves tasks that can be performed independently by a single robot, a tightly-cooperative solution requires robots to strongly cooperate in order to fulfill a common task. Tightly-cooperative solutions are also called strongly-cooperative [39].

If a task can be decomposed into subtasks that can be independently achieved by a single robot, then the problem reduces to a single-robot task problem (SR), where each subtask is treated independently. If on the other hand, the problem cannot be decomposed into independently achievable subtasks, the problem is a multi-robot task problem (MR). Solutions of single-robot task problems are weakly cooperative, and solutions of multi-robot task problems are tightly-cooperative [39].
1.2. Weakly-Cooperative Versus Tightly-Cooperative Solutions

Most approaches for MRTA deal with single-robot task problems and thus provide weakly-cooperative solutions. Tightly-cooperative solutions use coalition formation algorithms to form a team of robots that can collectively execute a multi-robot task [39]. Given a multi-robot task, groups of robots, called coalitions, are formed. Each coalition is assigned a utility, and the group with the best utility is selected for performing the task. Coalition formation algorithms for multi-agent systems can be modified to be more suitable for multi-robot environments. For instance, in [41], they modified the approach of [36] by reducing the communication needed between the robots and by introducing more constraints in the coalition formation.

ST-MR-TA adds time schedules to the problem, i.e., apart from creating coalitions of robots, a task schedule is built. According to the iTax taxonomy ST-MR-IA can be classified into XD[ST-MR-IA] and CD[ST-MR-IA], while ST-MR-TA can be classified into XD[ST-MR-TA] and CD[ST-MR-TA] [20].

F. Tang and L. Parker propose in [39] an approach that combines both, a weakly-cooperative and a tightly-cooperative solution in the same application. They use an auction-based mechanism for allocating weakly-cooperative tasks and their ASyMTRe-D coalition-formation for creating teams of robots that execute tasks in a tightly-cooperative manner. Their solution consists of two levels. In the low level, the coalition-formation algorithm generates teams of robots that could potentially perform multi-robot tasks. In the high level, the action-based algorithm allocates tasks or collection of tasks to coalitions or to individual robots. If a single robot is
better suited for executing a task than a group of robots, the task is assigned to the single robot. The action-based mechanism provides a weakly-cooperative solution but allocates tightly-cooperative solutions to coalitions of robots.

1.3 Task Decomposition

There are different ways of decomposing a task. In [20] the following terms are used to distinguish the types of tasks a robot can perform and how they can be decomposed.

- **Elemental or atomic task.** The task cannot be decomposed into subtasks, and it is allocated to a single robot.

- **Decomposable simple task.** The task can be decomposed into subtasks in exactly one way, but the resulting subtasks are allocated to the same robot.

- **Simple task.** Includes elemental tasks as well as decomposable simple tasks.

- **Compound task.** The task can be decomposed into subtasks in exactly one way, but the resulting subtasks can be allocated to different robots, i.e., the task is multi-robot allocable.

- **Complex task.** The task can be decomposed into subtasks in different ways, but at least one of the possible decompositions includes subtasks that can be allocated to multiple robots. The subtasks of a complex task can be simple, compound or complex.

There are two main approaches to task decomposition, namely decompose-then-allocate and allocate-then-decompose. In the decompose-then-allocate approach for task decomposition, a robot decomposes a task, and the resulting subtasks are allocated to the most suitable robots. The problem with this method is that a task cannot be optimally decomposed without knowing beforehand which robot will execute each subtask. The allocate-then-decompose approach allocates complex tasks to robots, which then decompose the task into subtasks. In this case, the problem is that without knowing the decomposition, an effective allocation cannot be done [11].

The MRTA problems relevant for our project are ST-SR-IA and ST-SR-TA.
1.4 Single-Task Single-Robot Instantaneous Assignment (ST-SR-IA)

In [14], Gerkey and Matarić show that the ST-SR-IA is an instance of the Optimal Assignment Problem (OAP) and as such, it can be formulated as a scheduling problem, which can be solved in polynomial time. The rest of the problems in the taxonomy are strongly NP-hard [20], which means that their solution is too computational complex to be solved in practice, but heuristics methods can be used.


- **One-time assignment.** Assumes that just a set of tasks is provided to the system and the solution of the problem is the one-time allocation of the tasks.

- **Iterated assignment.** It is an iterated instance of the one-time assignment. Whenever a new task enters the system, the allocation of the new task is performed, and robots which were assigned a task in the previous iteration become available for the new incoming task. That is, reassignment of tasks to robots is allowed. In each iteration, all robots release their previous tasks and are assigned new tasks.

- **On-line assignment.** Reassignment of robots is not allowed. A robot can only receive a new task once it has terminated the execution of its previous task or if no tasks have been assigned to it.

In most cases, multi-agent systems should be dynamic in the sense that they should be capable of receiving a new task or set of tasks at any time. Iterated assignments and on-line assignments are able to deal with the dynamic nature of the system. In applications where reassignment of tasks is not desirable, on-line assignment algorithms are preferable. In on-line assignments, a task will only be dropped in case of robot failure. In iterated assignments, a robot releases a task to favor a more capable robot even if the robot is still able of executing the task.

In the iTax taxonomy, ST-SR-IA problems can be classified into problems with no dependencies (ND[ST-SR-IA]) and cross-scheduled dependencies (XD[ST-SR-IA]).
Chapter 1. Introduction

The ND[ST-SR-IA] is equivalent to the ST-SR-IA described by [15], in which an independent task is assigned to a single robot. The XD[ST-SR-IA] problem with cross-schedule dependencies refers to the situation where a set of single-robot tasks, subject to inter-task constraints, are to be instantaneously assigned to single-task robots [20].

The MRTA-TOC taxonomy does not expand ST-SR-IA problems because this kind of problems do not build a schedule of tasks and hence cannot represent temporal constraints nor ordering constraints.

1.5 Single-Task Single-Robot Time Extended Assignment (ST-SR-TA)

In the ST-SR-TA problem, each robot builds a schedule for allocating tasks, and more than one task can be assigned to each robot. In [14] they define the problem as an instance of the class scheduling problems, which are NP-hard.

According to the iTax taxonomy, ST-SR-TA can be classified into ND[ST-SR-TA], ID[ST-SR-TA], XD[ST-SR-TA], and CD[ST-SR-TA]. The MRTA-TOC further expands TA into problems with temporal constraints (TW) and problems with synchronization or precedence constraints (SP).

1.6 Motivation

The indoor transportation of goods is a practical application of multi-robot systems in the area of logistics. While incorporating belt conveyor systems can be expensive and disruptive, the implementation of a multi-robot system does not require significant changes in the distribution or construction of the building. Moreover, the transportation of goods is not only an issue in warehouses. Moving hospital beds and medical equipment around hospital facilities are everyday tasks, whose execution could be performed by a multi-robot system.

The objective of this project is to compare multi-robot task allocation algorithms for a multi-robot system operating in a hospital. The case scenario covered in this R&D is part of the ROPOD project sponsored by the European Commission and coordinated by the Hochschule Bonn-Rhein-Sieg. The goal of the multi-robot system is to transport heavy loads within the hospital facilities. Such tasks are usually
performed by the human workforce in the hospital, but are physically demanding and can damage health if performed frequently or during extended periods.

The ROPOD project covers two use cases, namely the transportation of sick beds and the transportation of supply carts carrying medical equipment. In this R&D the focus is on analyzing algorithms for allocating supply carts transportation tasks to the fleet of robots. Relieving medical personnel from performing these tasks will benefit their health, as they will no longer be exposed to hazardous manual tasks. Moreover, patient care will improve due to the enhancement of transportation of medical equipment in the hospital.

1.7 Challenges and Difficulties

Although some comparison evaluations are reported in the literature [15, 24, 25], they do not provide an extensive set of performance metrics, nor recommendations on which algorithm to chose based on the application of the multi-robot system. It is thus difficult to decide which algorithm best suits the use cases in the ROPOD project.

Before selecting an algorithm, the specifications and requirements of the multi-robot system must be defined. While some systems require to allocate and schedule tasks to be performed at specific time windows, others do not consider temporal constraints. Some algorithms optimize distance, while other optimize execution time. Some use task priority information and have re-allocation mechanisms in case of failure, while others do not. Certain algorithms provide optimal solutions, but their run-time is too large to perform efficiently in an on-line setup, where new tasks are introduced at run-time. Even after considering all these factors, there might be no algorithm that entirely fits the needs of the application.

Moreover, algorithms should be tested using a common experimental setup that includes a variety of task distributions and means of testing robot scalability. However, there is a lack of open source implementations for MRTA algorithms [26].

1.8 Problem Statement

This project compares existing multi-robot task allocation algorithms (MRTA) and analyses their suitability for assigning transportation tasks to a multi-robot system operating in a hospital. The tasks consist in picking up and delivering supply
carts carrying medical equipment within hospital facilities. The allocation schemes covered are:

- Allocate a task so that it can be fulfilled as soon as possible.
- Allocate a task so that it can be fulfilled at some specific time in the future, e.g., tomorrow at 9 am.

Different algorithms for dealing with the task allocation problem exist in the literature, but just few experimental comparisons of their performance have been conducted.

In [15] the authors compare the algorithmic characteristics of MRTA algorithms for the ST-SR-IA problem. The criteria used for the comparison is the solution quality of the algorithms, as well as their computation and communication requirements. Previous to their study, the selected architectures were tested by different authors under different use cases and applications. That is, the algorithms were experimentally evaluated, but under different conditions. The contribution of [15] is a qualitative evaluation, whose purpose is to assess the algorithms independent of a particular application. Even though such a comparison is very valuable, it is also desirable to investigate the performance of the algorithms on a common ground. Comparing the performance of different approaches for solving the same problem helps on identifying the characteristics of each approach which makes it more suitable or unsuitable for a particular application.

In [24] the authors experimentally compare auction-based algorithms for MRTA using different optimization criteria and a common application. They evaluate two ST-SR-TA algorithms with one variation each and one ST-SR-IA algorithm. The focus of the comparison is the optimization criteria, i.e., the parameter(s) that the system tries to minimize or maximize during the allocation. The algorithms are not tested in an on-line scenario, i.e., new tasks are not introduced at runtime. Their experiments are part of the exploration domain, particularly object searching, and all experiments are performed with a fixed number of four robots. The authors report the minimum total resource usage and minimum total time for completing the mission. In their experiments, the robots have to visit a set of locations, find an object or find a set of objects in a limited time. The uses cases with which we
evaluate the MRTA algorithms correspond to a different domain, namely the indoor transportation of objects. In our scenario, new tasks will be introduced at run-time, which means that the system must be able to allocate tasks on-line.

In [25] the authors compare the Temporal Sequential Single-Item auction (TeSSI) algorithm, which creates a time-schedule, against a greedy algorithm and a consensus-based-bundle auction algorithm. The metrics used for comparison are the total number of tasks allocated, makespan and distance. Their experiments include on-line allocation tasks, but they do not consider re-allocation mechanisms.

There is, however, to the best of our knowledge, no study that evaluates the strengths and weaknesses of MRTA algorithms for allocating transportation tasks. It is thus desirable, to identify the requirements and features needed for this particular application, design experiments and report performance metrics that reflect the suitability of existing MRTA algorithms for solving this problem. This study will provide the means for making an informed decision on the selection of the algorithm for the use cases in ROPOD.
State of the Art

The problem of allocating tasks to a multi-robot system can be modeled as an optimization problem. Methods like genetic algorithms and colony systems, require a centralized topology and perform off-line allocation of tasks. Stochastic process systems are difficult to model in dynamic environments, and the design of a local objective for game theory based approaches becomes difficult when the tasks and robots are heterogeneous [9]. In this work, we focus on distributed algorithms designed to allocate tasks on-line, but also briefly review some other methods.

2.1 Optimization Problem

MRTA can be defined as an optimization problem, where the parameter to optimize is the utility. MRTA uses the concept of utility to decide which robot is better for executing a particular task. This concept is also used in other areas such as operations research, economy, and game theory and it is sometimes called cost, valuation or fitness [15].

In multi-robot systems, the utility is the performance that a robot estimates it could achieve if it were to execute a task. Robots can apply different approaches for calculating their utility for a specific task, but the importance of this value is that it can be used for comparing how good a robot could perform a task. In essence, MRTA algorithms compare the utilities of all robots for a particular task and allocate the task to the robot with the most promising performance estimate.
Because multi-robot systems operate in the real world, the utility measure (usually based on sensor readings, distances on maps or other information of the world) includes uncertainty and noise. In the context of MRTA, when a solution is referred to as “optimal”, it is optimal in the sense that its provided solution is the best given the available information [15].

Gerkey and Matarić define utility for multi-robot systems by combining the expected quality and the expected cost of the task execution [15]. Regardless of the method used for calculating utilities, the utility of a robot for performing a task should encompass all relevant information about the robot and its environment. Information used for evaluating the performance of a system but not included in the utility calculation is referred to by economists as externalities and its effects are prejudicial to the system [16].

Utilities must be expressed as scalars so that they can be compared against the utilities of other robots for performing the same task [15].

2.2 MRTA Related Problems

Multi-robot task allocation (MRTA) research started in the 90’s [26]. In [26] the authors summarize other problems strongly related to MRTA:

- Multiple Traveling Salesman Problem (mTSP).
- Vehicle Routing Problem (VRP) and Vehicle Routing Problem with Time Windows (VRPTW).
- Job Scheduling Problem (JSP).
- Team Orienting Problem (TOP) and Team Orienting Problem with Time Windows (TOPTW).
- Dial-a-ride Problem (DARP).

MRTA algorithms operate in uncertain environments, where the information acquired is incomplete and prone to errors [12], they differ from other related problems in the potentially variable number of robots, the restricted communication, the dynamic nature of the environment, the heterogeneity in tasks and robot capabilities,
Chapter 2. State of the Art

and the introduction of new tasks, among other factors [26].

**Multiple Traveling Salesman Problem (mTSP).** The multiple traveling salesman problem (mTSP) is an extension of the single salesman traveling problem in which the aim is to obtain the shortest route for visiting once each city in a list of cities. In the mTSP, each salesman has to visit each city exactly once and return to the start city. In the case of multi-robot systems, the salesmen are replaced by robots and the cities by tasks [18]. The Multi-Depot Multiple Traveling Salesman problem is a variant of the mTSP in which all salesmen start at different cities [20].

**Vehicle Routing Problem (VRP) and Vehicle Routing Problem with Time Windows (VRPTW).** Each task consists of transporting an item. The Vehicle Routing Problem deals with allocation and routing simultaneously. The solution of the problem is the best route that each vehicle can take, from the start position to the destination, taking into account operational constraints and the transportation cost. For solving this kind of problems, an objective function, subject to a set of constraints is minimized. An example of a Vehicle Routing Problem is the Capacitated Vehicle Routing Problem (CVRP), in which all vehicles start on the same position and have a maximum capacity for transporting goods. VRPTW is an extension of the VRP in which the objective function includes temporal constraints in the form of time windows. In VRP and VRPTW problems the transportation requests are known in advance [20]. The system assumes any-time availability of an infinite number of homogeneous robots and no communication constraints [26].

**Job Scheduling Problem (JSP).** Jobs are assigned to machines, so as to minimize the cost for performing the jobs. Methods for solving these problems do not consider travel time in the calculations, i.e., the job comes to the machine, and no physical transportation of machines is needed [26].

**Team Orienting Problem (TOP) and Team Orienting Problem with Time Windows (TOPTW).** Consists of visiting control points between a start and end position so as to maximize the system profit. Each visited point awards a profit and moving from one point to another incurs a cost. Usually, not all control
points can be visited. The solution includes those control points that maximize the profit [26].

**Dial-a-ride Problem (DARP).** Consists on creating allocations and routes for visiting a set of locations so as to visit as many locations as possible while satisfying a set of constraints. Unlike VRP, DARP is designed for transporting passengers and includes quality of service in the calculations. Route and length duration and violations on desired delivery start times are measurements of the quality of service of the system. Most approaches assume homogeneous vehicles and a common start position for all vehicles [7].

2.3 MRTA Topologies

MRTA methods can be classified based on their topology as centralized, decentralized and distributed approaches [9].

- **Centralized approaches.** One central component makes all the computations and allocation decisions. The central component can be a robot or a computer. Solutions are optimal because they are calculated using the full information about the robots in the system. However, computations can become intractable for large multi-robot systems [18]. Changes in the environment, like the introduction of new tasks, require the algorithm to create a new allocation from scratch [6], which makes them unsuitable for on-line allocation of tasks. Moreover, communication requirements are very high, as robots need to share their information with the central component. These approaches do not scale well and have a single point of failure. They are suitable for small fleets of robots and for applications where all tasks are known beforehand (offline allocations) [11]. They are not recommendable for applications where communication and execution of the allocated tasks are prone to failure [25].

- **Decentralized approaches.** Computations and allocations are performed by several decision-making components which have hierarchical relationships. The decision making components can be robots in the fleet. However, if the component at the top of the hierarchy fails, the whole system fails [9].
• **Distributed approaches.** Each robot decides which task to perform based on the information available to it. Conflicts in allocations are resolved through a negotiation process among the robots [9].

Some authors use the three categories described above [9], while others only use the centralized and distributed categories [11] or the centralized and decentralized categories [18, 25]. In this work, we refer to methods that divide the allocation computations among the robots in the fleet as distributed methods. Distributed approaches are robust, scalable and have fewer communication requirements compared to centralized approaches, but might produce sub-optimal solutions because not all information is taken into account for performing the allocation of tasks [18].

Figure 2.1 classifies some MRTA approaches based on their topology. Table 2.1 shows some MRTA algorithms classified based on the type of topology and approach.

![Figure 2.1: MRTA approaches based on their topology.](image)

In [40] the authors implemented a multi-robot system for allocating transportation tasks in an office building. The scheduler generates a new allocation every time a new task is introduced, i.e., a new mixed integer program (MIP) has to be solved every time a new task enters the system. According to the authors, their method “is effective for a small number of tasks, on the order of two robots and fifteen tasks.” [6]. For handling larger multi-robot systems, the authors propose in [6] a scheduler that plans to transfer tasks. Instead of recalculating the whole schedule, they adjust it by using auctions to transfer tasks between robots. A robot already carrying an item, transfers the item to another robot with the assistance of a human.

Market-based methods are inspired by economic theory. They are sometimes referred to as auction-based approaches because of their use of auctions to allocate
tasks to robots. We use the terms market based and auction-based interchangeably. Some authors classify marked based approaches as decentralized [26, 25], others as distributed [13, 15] and some others as a hybrid of the centralized and the distributed approach [11]. To avoid confusion, we will refer to them as distributed approaches. It is also important to make the distinction between pure auction-based methods and methods that use an auction phase combined with a consensus phase.

- **Auction-based methods.** Each robot computes a bid for a task and sends it to a central auctioneer. Based on the bids received, the central auctioneer decides which robot will execute the task.

- **Auction-consensus based methods.** There is no central auctioneer. In the auction phase, each robot makes allocations based on the information available to it. Conflicts in allocations are resolved using a consensus algorithm. Because auction-consensus based methods do not have a central auctioneer, we will refer to them as fully distributed.

Auction-based methods have a central component but maintain the benefits of decentralized approaches, such as robustness, flexibility, and speed [11]. They distribute the computation load among the robots on the fleet and reduce the communication overhead because robots only need to send a numeric bid to the auctioneer. Moreover, the system is robust to local changes and flexible to the introduction or removal of robots. If a robot fails, it can be removed or replaced without disrupting the allocation process [25]. Auction-based methods naturally converge to conflict-free assignments because the auctioneer determines only one winner for each task. Differences in the situation awareness of the robots do not produce conflicting allocations.

In auction-consensus based methods, each robot elects a winner during the auction phase. Because robots might have different information about the environment, they might elect different winners for the same task. Conflicting assignments are resolved during the consensus phase, in which the robots agree on the winners for each task [4].

Auction-based methods require a completely connected network so that each robot can send its bid to the auctioneer. In scenarios where robots might go out of communication range, an alternative is to perform the allocation algorithm only
between the neighboring robots of the auctioneer. However, this solution compromises allocation quality because not all robots are taken into account [4, 17, 38].

The first auction-consensus algorithms were proposed in [4]. The authors proposed the consensus-based auction algorithm (CBAA) for instantaneous assignment and the consensus-based bundle algorithm for time extended assignment. Their algorithms combine the advantages of auctions for performing decentralized allocations and add a consensus component to make them robust to network topologies. Unlike traditional consensus algorithms, the consensus phase is used to agree on the winner of bids and not to agree on the situation awareness of the robots.

In a market-based approach, a team objective is defined. The goal of the system is to allocate tasks so as to optimize the team objective [22].

Some examples of team objectives are:

- **MINISUM.** The goal is to minimize the resources needed to accomplish the team mission. For example, minimize the energy consumed by all robots. If the mission is to visit a set of locations, the MINISUM objective is to minimize the sum of the cost of the paths of all robots [22].

- **MINIMAX.** The goal is to minimize the resources needed by the worst performing robot in the fleet. For example, minimize the cost of the path of the worst performing robot [22].

The MINISUM team objective is the most commonly used in auction-based methods [24].

In market-based methods, each robot places a bid that represents its utility or cost for executing the auctioned task. If the bids represent the utility for the task, a higher bid value represents a higher bid. On the other hand, if the bids represent the cost for performing the task, a lower bid value represents a higher bid. [11]

Some MRTA algorithms based on auctions are:

- **Single-item auction.** Only one task is auctioned. The robots submit a bid for the task and the auctioneer assigns the task to the robot with the highest bid [11]. If there is more than one unallocated task, the algorithm runs iteratively until all tasks are allocated [24].
• **Sequential single-item auction.** It is an iterative algorithm. In each iteration, a set of tasks is auctioned, but only one task is allocated. In the next iteration, the remaining unallocated tasks are auctioned again, and each robot takes into account its already allocated tasks to compute its bids. The auctioneer receives a vector of bids from each robot and allocates the task with the highest bid among all the entries of all the bid vectors. The algorithm terminates when all tasks are allocated [19].

• **Parallel single-item auction.** It is similar to the sequential single-item auction, with the distinction that robots do not take into account its allocated tasks to compute their bids. Unlike sequential single-item auctions, they do not model the synergies between tasks and thus produce lower quality allocations [19].

• **Single-round combinatorial auctions.** The auctioneer auctions bundles of tasks which model the synergies between the tasks. The robots place bids for a complete bundle. Each bid represents the cost of executing all tasks in the bundle. A robot cannot win more than one bundle of tasks, and each task can be part of only one bundle [2]. One of the problems of combinatorial auctions is deciding on which bundle to bid. As the number of tasks increases, the number of bundles grows exponentially [19].

Two common ways of submitting bids are:

• **Sealed-bid auction.** Bidders submit their bids only to the auctioneer [11].

• **Open-cry auction.** Bidders make their bids publicly, allowing all participants to overhear the bids [11].

Open-cry bids allow all participants to have information about the bids, but since the bid of a robot for a task is not affected by the bid of another robot for the same task, market based MRTA commonly use sealed-bids [11].
### Table 2.1: MRTA methods based on topology and approach.

<table>
<thead>
<tr>
<th>Topology</th>
<th>Approach</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized methods</td>
<td>Exact solutions</td>
<td>Hungarian algorithm [15]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Solving a mixed integer program (MIP) [40]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Branch and bound methods [5]</td>
</tr>
<tr>
<td></td>
<td>Approximate solutions</td>
<td>MIP based heuristics [26]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Metaheuristic parameters [26]</td>
</tr>
<tr>
<td>Market-based</td>
<td></td>
<td>MURDOCH [13]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Single-item auctions [24]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Parallel single-item auctions [19]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Single-round combinatorial auctions [2]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sequential single-item auctions (SSI) [21]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Temporal sequential single-item auctions (TeSSI) [25]</td>
</tr>
<tr>
<td>Market-consensus based</td>
<td></td>
<td>Consensus-based auction algorithm (CBBA) [4]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Consensus-based bundle algorithm (CBAA) [4]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Consensus-based parallel auction and execution (CBPAE) [9]</td>
</tr>
<tr>
<td>Behavior based</td>
<td></td>
<td>BLE [42]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALLIANCE [29]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L-ALLIANCE [28]</td>
</tr>
</tbody>
</table>
2.3. MRTA Topologies
3 Qualitative Comparison

In this chapter, we describe some MRTA algorithms and evaluate them based on the following comparison criteria.

3.1 Comparison Criteria

1. Quality of solution. Indicates if a solution is optimal or how far away it is from the optimal solution [14].

2. Communication requirements. Number of messages that need to be sent between the members in the fleet before an allocation can be made. It is represented using \( n \) number of components in the fleet (robots + auctioneer) and \( m \) number of tasks [14].

3. Computation requirements. Complexity of the algorithm for electing the winning robot [14].

4. Scalability. Indicates if the system’s performance is expected to decay when the number of robots or tasks increases.

5. On-line task allocation. Ability of the algorithm to allocate new incoming tasks at runtime.

6. Fault tolerance capabilities. Reallocation mechanisms in case of failure or poor performance.
7. **Priority task allocation.** Ability to allocate tasks with different priorities.

8. **Heterogeneity.** Ability of the algorithms to work with heterogeneous robots and tasks.

9. **Validation in real-world applications.** Implementation of the algorithms in previous works and evaluation of the experimental results.

10. **Special characteristics.** Properties of the algorithms that do not fit in the previous criteria and that differentiate them from the others analyzed in this chapter.

### 3.2 ST-SR-IA Algorithms

The Single-Task Single-Robot Instantaneous Assignment algorithms analyzed in this section are MURDOCH, Alliance, and Consensus Based Parallel Auction and Execution.

#### 3.2.1 MURDOCH

MURDOCH is a distributed auction-based MRTA algorithm proposed by [13]. It is designed to work in teams of failure-prone robots, where new tasks arrive dynamically. According to Gerkey and Matarić’s taxonomy [15], it is an ST-SR-IA on-line assignment algorithm because reassignments of tasks are only allowed in case of robot failure.

It uses single-item auctions in a sequential way. One task is auctioned per iteration until there are no more tasks to allocate. The goal is to assign resources (i.e., robots) to tasks. Each task is allocated to the most suitable robot at the moment of bid closure. The introduction of a new task into the system triggers the execution of the allocation algorithm. A central auctioneer determines the winner of each task and offers it a limited-time contract. The time limited-contract reflects the estimated duration of the task. The auctioneer monitors time execution by periodically sending renewal messages to the currently engaged robots. If the robots fail to respond or are taking too long for executing the task (the contract is expiring), the auctioneer terminates the contract by not sending renewal messages to the robot.
Chapter 3. Qualitative Comparison

It is a variant of the Contract Net Protocol and uses simple fitness-based auctions. The bid submitted by each robot represents its fitness score for executing the task. The bid calculation can encompass several metrics, such as Cartesian distance to the task location or battery level of the robot. The specific metric selection depends on the implementation. The auctioneer closes the bidding process after some specified time \(^1\) and selects the robot with the best fitness score.

The communication scheme used by MURDOCH is based on publish/subscribe messaging. Messages are identified by their contents and not by their recipients. The contents of a message are called subjects and represent physical resources (e.g., laser, gripper, microphone), high-level capabilities (e.g., mobile, docking-capabilities) or the current state of the robot (e.g., idle, busy). For example, a robot with a laser, mobile capabilities and currently unengaged will subscribe to the subjects (laser, mobile, idle). A task is tagged with the subjects that represent the resources needed for executing it. For instance, a transportation task might be tagged with the subjects (mobile, docking-capabilities, idle). All robots subscribed to these subjects will receive the task announcement message, which means that only capable robots can bid for the task.

The algorithm uses the following kind of messages:

- **Announcement message.** Send by the auctioneer. Contains the information of the task to be allocated in the current iteration:
  - Task ID
  - Task duration
  - Subjects: Resources needed to execute the task.
  - Metrics: Used by the robots to calculate their fitness score for the task.

- **Bid message.** Each robot that received the task sends a bid message which contains the fitness score of the robot for executing the task.

- **Close message.** Sent by the auctioneer. Includes the robot ID of the winner and the time limited contract.

\(^1\)1.5s in [13]
• **Renewal message.** The auctioneer monitors the execution of tasks by periodically sending renewal messages to each robot engaged in task execution.

• **Acknowledgment message.** Each robot that receives a renewal message responds with an acknowledgment message. If a robot stops replying to renewal messages and the task has not been completed, the auctioneer terminates the contract. The robot releases the task and re-enters the auction process.

**Comparison Criteria**

1. **Quality of solution.** There are no guarantees of achieving an optimal solution because the algorithm behaves as a greedy algorithm. Each task is allocated to the most suitable robot at the moment of allocation. Compared to an optimal off-line solution, the greedy algorithm is 3-competitive\(^2\). According to [15] “without a model of the tasks that are to be introduced, and without the option of reassigning robots that have already been assigned, it is impossible to construct a better task allocator than MURDOCH.”

2. **Communication requirements.** The auctioneer sends an announcement message per task, and each eligible robot sends a bid message to the auctioneer. That is, each component in the fleet sends one message, resulting in \(O(n)\) communication overhead [14].

3. **Computation requirements.** In every iteration, each bidder processes a bid \(O(1)\) and the auctioneer processes \(n\) bids, one for each robot \(O(n)\) [15].

4. **Scalability.** To the best of our knowledge, no experimental results have been reported to test the scalability of the algorithm. However, based on the communication and computation complexities, we expect the algorithm to scale well. Since one task is allocated per iteration, the number of iterations increases linearly with the number of tasks.

5. **On-line task allocation.** The algorithm is designed to allocate incoming tasks at runtime. It is a sequential algorithm, i.e., one task is auctioned per

---

\(^2\)The competitive-factor is used to evaluate the quality of non-optimal solutions. “For a maximization problem, an algorithm is called \(\alpha\)-competitive if, for any input, it finds a solution whose total utility is never less than \(1/\alpha\) of the optimal utility”. [15]
iteration until there are no more tasks to allocate. The introduction of a new task into the system triggers the execution of the allocation algorithm [13].

6. **Fault tolerance capabilities.** MURDOCH monitors the performance of the robots engaged in task execution by sending them a renewal message. If a robot fails to respond, the algorithm assumes that there was a failure, terminates the contract and reallocates the task in the next iteration [13].

7. **Priority task allocation.** The algorithm does not take task priorities into account, but modifications could be made. For example, unallocated tasks could be ordered based on their priority so that higher priority tasks are auctioned first. However, if a new task with higher priority enters the system and an auction process is already taking place, the new high priority task will be allocated in the next iteration, but not in the current one, unless a preemption mechanism is introduced.

8. **Heterogeneity.** The algorithm is designed to allocate homogeneous or heterogeneous tasks to homogeneous or heterogeneous robots [13].

9. **Validation in real-world applications.** It was the first auction-based MRTA algorithm tested in physical robots using different applications [13]. The authors evaluated the algorithm using a variety of loosely-coupled single-robot tasks as well as a tightly coupled multi-robot task.

Tightly coupled multi-robot tasks are handled by structuring the tasks as hierarchical trees. A high-level parent task is assigned to a robot, which in turn is responsible for allocating and monitoring the low-level child tasks. In the experiment presented in [13], the authors allocate a box pushing task to a group of robots. A robot receives the high-level task of “watcher” and auctions the low-level tasks of “pushers” to two robots. The three robots form a team that jointly executes the task. The watcher guides the pusher robots by giving them indications on how they should move the box to transport it to the goal position.

Four heterogeneous loosely-coupled single-robot tasks (object-tracking, sentry-duty, cleanup, and monitor-object) were introduced randomly to a multi-robot
system of eight heterogeneous robots. When the resources required for a task were available, the task was always allocated to the robot with the best fitness score.

10. **Special characteristics:** Robots monitor their battery level and remove themselves from the allocation process if their battery level is below a certain threshold. They go to a charging station and re-enter the allocation decision process after charging their battery for some time [13].

### 3.2.2 ALLIANCE

ALLIANCE is a behavior-based fully distributed MRTA architecture proposed by Lynne E. Parker [27]. It uses an iterative assignment scheme, i.e., task reassignment is allowed, and tasks are allocated based on the impatience and acquiescence levels of the robots. It is fault tolerant because of its reallocation capabilities. If a robot currently engaged in a task is making unsatisfactory progress, other robots capable of performing the task become impatient until eventually one of them takes over the task. ALLIANCE adapts to changes in robot performance, and in the environment. It is designed for “small to medium sized teams of heterogeneous mobile robots, performing (in dynamic environment) missions composed of independent tasks that can have ordering dependencies” [29].

Each robot has a set of behaviors that become active based on their motivation levels (impatience and acquiesce). When the impatience level of a robot exceeds a threshold (fixed parameter), the set of behaviors for a particular task activate and the robot begins task execution. Robots broadcast their current activity so that other robots can adjust their motivation levels. Impatience of idle robots and acquiesce of busy robots increase with time, and robots decide to give up tasks or take over tasks based on their motivation levels. A variation of ALLIANCE called L-ALLIANCE tunes the parameters for calculating the motivation levels based on the performance of the fleet [28].
Comparison Criteria

1. **Quality of solution.** Compare to an optimal off-line solution, ALLIANCE is 2-competitive in the worst case [15].

2. **Communication requirements.** $O(m)$ since each robot broadcasts a heartbeat message [14].

3. **Computation requirements.** For each task, the robot computes its utility and compares it to the robot currently executing that task. Thus the computation requirements per iteration are $O(mn)$ [14].

4. **Scalability.** The algorithm is designed for small to medium size fleets of robots [29]. We are not aware of a study that tests the scalability of the algorithm.

5. **On-line task allocation.** ALLIANCE is designed to allocate tasks at runtime, but since reassignment of tasks is allowed, its allocation scheme is “iterated-assignment” and not “on-line assignment” [15].

6. **Fault tolerance capabilities.** The algorithm adapts to changes in the environment and the performance of the robots [29].

7. **Priority task allocation.** The algorithm does not account for task prioritization.

8. **Heterogeneity.** ALLIANCE is designed to work in heterogeneous multi-robot systems [29].

9. **Validation in real-world applications.** In [27] the algorithm was validated using a box pushing application. In [29], the authors used a waste cleanup application to validate the approach in physical robots.

10. **Special characteristics.** Robots adapt to changes in the environment and in the fleet performance by internally modeling the utilities of all the robots currently performing a task [14]. The parameters modeling the utility computations can be tuned based on the performance of the fleet over time [28].
3.2.3 Consensus Based Parallel Auction and Execution (CBPAE)

The CBPAE algorithm is an auction-consensus based method presented in [9]. It combines an auction phase where each robot bids on tasks and a consensus phase in which all robots reach an agreement about the allocations. Unlike traditional auction-based methods, the CBPAE does not have a centralized auctioneer. The allocations are based on the situation awareness of the robots and the messages interchanged during the consensus phase. The algorithm allocates tasks based on a priority from 0 to 5, where 0 represents an emergency task, and 5 represents a task with the lowest priority. It handles emergency tasks differently so that they are allocated as soon as possible.

CBPAE was designed to allocate heterogeneous tasks to a group of heterogeneous robots in healthcare facilities. It is based on the consensus-based decentralized auction algorithm (CBBA), but instead of allocating tasks in an extended manner, it allocates tasks instantaneously to the best suitable robot at the moment of auction closure. All robots, including the ones currently engaged, bid on a new task. Bids made by engaged robots change dynamically as robots progress in the execution of their tasks. Each robot can only allocate one task at a time, which means that a new task can only be allocated once the robot has concluded the execution of its previous task. This method belongs to the category ST-SR-IA of Gerkey and Matarić’s taxonomy.

The authors advocate for the use of IA in dynamic environments, where new tasks and robots are introduced or withdrawn at run-time. They argue that using TA in off-line assignment scenarios, where all tasks are allocated beforehand, has the disadvantage of having to recompute the schedules if a new task is introduced during the execution phase. However, some TA algorithms are also designed for allocating tasks on-line [21, 25, 6]. It would be interesting to compare the performance of CBPAE against TA algorithms.

With CBPAE, each robot has a local version of five task vectors per task and based on its local information, performs a bidding process that consists on the following steps:
1. Get the set of free tasks (FT), i.e., tasks that are currently unallocated.

2. From the free tasks, get the set of biddable tasks (BT), i.e., tasks for which the robot has the required skills and whose priority is higher or equal than the highest priority task in the set of free tasks.

3. If the highest priority is an emergency task:
   
   (a) Robots in the first execution phase (which consists on going to the task location) of a lower priority task, drop task execution and withdraw their current bid.
   
   (b) Go back to step 1.

4. Calculate a bid for each task in the biddable tasks. The bid calculation is computed differently if the robot is currently idle or if it is busy with a task.

5. For each task, check for bids that are smaller than the current bid on that task.

6. Select the smallest bid over all the bids.

7. Place the bid by changing the task vector values as indicated in [9].

The bidding process is repeated iteratively. Bids are calculated based on the robot expertise and the work estimate for executing the task. A smaller bid value represents a better bid.

The consensus phase takes place before a new iteration of the bidding process begins. During the consensus phase, robots broadcast a message that contains information of four tasks, namely the task for which the robot is currently bidding on, the previous task the robot bid on, the current task being executed by the robot and the task that was executed by the robot in the previous iteration. Each of these tasks contains four task vectors and a task index. Each message has a total size of 96B, which is independent of the number of robots and tasks in the system. Based on the messages received by other robots, each robot updates its local version of the task vectors using the consensus actions as indicated in [9].

During the task assignment phase, a robot decides if it can assign a task to itself for which it has the highest bid (smaller bid value). It makes this decision based
on a dynamic bidding window. Each task has a dynamic bid window which starts when the first bid for that task is placed and ends either after some fixed amount of time or after the currently engaged robot with the highest bid ends the execution of its current task. After assigning a task to itself, the robot updates its task vectors accordingly.

Comparison Criteria

1. **Quality of solution.** Not reported in the paper, but because of its fully distributed nature and instantaneous scheme, the solution is not guaranteed to be optimal.

2. **Communication requirements.** Each robot broadcast a message of constant size 96B to all robots in the fleet [9]. The communication overhead is $O(n)$.

3. **Computation requirements.** Each robot has the same computation overhead, i.e., the computation requirements grow linear with the number of robots [9].

4. **Scalability.** The algorithm scales well because the size of the messages exchanged between robots remains the same as the number of tasks and robots increases. The authors tested the scalability of the algorithm by performing allocations with 5 to 50 robots and tasks equal to twice the number of robots. They found that the communication bandwidth requirements increase linearly with the number of robots [9].

5. **On-line task allocation.** CBPAE is designed to work in dynamic scenarios, where new tasks arrive at run-time, the number of robots in the fleet varies, and robots drop their tasks due to execution errors or introduction of emergency tasks [9].

6. **Fault tolerance capabilities.** When a task is dropped, it is reallocated in the next bidding process [9].

7. **Priority task allocation.** The algorithm is designed to assign tasks based on their priority. Higher priority tasks are always assigned before lower priority tasks, and emergency tasks have preference over all other tasks.

34
Task execution has two phases, in the first phase the robot travels to the task location and in the second phase, it executes the task. When an emergency task enters the system, all robots engaged in the first execution phase of lower priority tasks drop their tasks and their current bids. A new bidding process for allocating the emergency task, the other unallocated tasks and the dropped tasks begins. In this new bidding process, emergency tasks are allocated first [9].

8. **Heterogeneity.** CBPAE is designed for allocating heterogeneous tasks to a fleet of heterogeneous robots. Each task has a set of required skills, and each robot has a set of skills and an expertise level (from 0 to 1) for each skill. For instance, for the skill “navigation”, a robot equipped with sonars has an expertise level of 0.7, while a robot with sonars and range lasers has an expertise level of 0.9 [9].

9. **Validation in real-world applications.** CBPAE was tested both, in a simulated environment and on real robots. In the simulated environment, the authors evaluated CBPAE and CBBA using a homogeneous fleet of robots and homogeneous tasks. They found that the overall execution time of CBPAE is lower than that from CBBA. This is because CBBA overloads some robots while keeping others idle. A detailed analysis of all results is found in [9].

10. **Special characteristics.** Designed for allocating heterogeneous tasks to a group of heterogeneous robots where tasks have different priorities, and emergency tasks have priority over all other tasks. There is no central auctioneer and robots bid and execute tasks in parallel [9].

### 3.3 ST-SR-TA algorithms

3.3.1 Single-Round Combinatorial Auctions

Single-round combinatorial auctions allocate bundles of tasks instead of individual tasks [19]. Robots bid for bundles of tasks and the auctioneer allocates each bundle to the robot that placed the best bid for that bundle.

Tasks within a bundle have positive synergies, and a task cannot be part of more than one bundle. Robots win a bundle of tasks, i.e., they allocate a set of tasks which have positive synergies between them. Two tasks have positive synergies if “their combined cost for a robot is smaller than the sum of their individual costs” [19]. For instance, in a transportation scenario, two clustered tasks have positive synergies because the cost of a robot for visiting both of them is smaller than the sum of the cost of two robots visiting one task each.

The algorithm returns optimal solutions since it considers the synergies between tasks. [19] However it becomes impractical because of the reasons mentioned in [19]:

- The number of bundles increase exponentially with the number of tasks.
- Robots bidding for a bundle have to calculate the best path for visiting all tasks in the bundle, and this is an NP-problem.
- The auctioneer has to solve an NP-hard or a problem of exponential size to determine the winners of the bundles.

In [2] the authors propose strategies for bidding on bundles of tasks. However, the solutions are no longer optimal, and the selection and computation of the bundles are complex [19].

Comparison Criteria

1. **Quality of solution.** Optimal if no approximation methods are applied [19].

2. **Communication requirements.** The auctioneer announces all tasks, and the robots bid on bundles of tasks. Single-round combinatorial auctions allocate all tasks in one auction, while multi-round combinatorial auctions consist of
several auctions, one per round. The communication requirements of multi-round combinatorial auctions are higher than for single-round combinatorial auctions [2].

3. **Computation requirements.** The auctioneer needs to solve an NP-hard problem for electing the winners of the bundles [3]. The algorithm runs in exponential time [19].

4. **Scalability.** The algorithm does not scale well with the number of tasks because the number of bundles increases exponentially with the number of tasks [19]. Moreover, as the number of robots increases, the problem of electing the winner becomes more complex.

5. **On-line task allocation.** The introduction of new tasks triggers a new auction process [2].

6. **Fault tolerance capabilities.** The algorithm, as described in [19] and [2] does consider failures in task execution.

7. **Priority task allocation.** The algorithm does not take into account the priority of tasks.

8. **Heterogeneity.** [19] and [2] do not consider robots with heterogeneous capabilities. However, one of the benefits of auction-based methods is that they can account for heterogeneity in the bid calculation [11]. Modifications in the bidding scheme would need to be made to use single-round combinatorial auctions in heterogeneous scenarios.

9. **Validation in real-world applications.** In [2] the algorithm is implemented in the domain of terrain exploration using the Teambots simulation environment, but no validation in physical robots is performed.

10. **Special characteristics.** No special features.

### 3.3.2 Sequential Single-Item Auctions (SSI)

In [21], the sequential single-item algorithm (SSI) also called multi-round single-item algorithm, or PRIMALLOCATION is presented. A multi-robot routing
3.3. ST-SR-TA algorithms

application in which tasks consist on visiting target locations is used to test the algorithm in a simulated environment. In each round or iteration, a set of tasks is advertised, but only the task with the highest bid from all the bids is allocated. Bids are received for a predefined amount of time after which the auction closes [19]. One task is allocated per round until there are no more tasks left to allocate.

The algorithm can handle off-line allocations as well as on-line allocations. However, in [21] the authors only tested off-line allocations. For on-line allocations, the introduction of a new task or set of tasks re-triggers the allocation process. This algorithm belongs to the ST-SR-TA category in Gerkey and Matarić’s taxonomy because each robot builds a schedule of tasks.

The multi-robot routing problem using a MINISUM team objective [19] works as follows:

1. The auctioneer auctions a set of unallocated tasks.

2. Each robot calculates its bid for each task using the cheapest insertion heuristic:
   - Each robot has a minimum path for visiting all its allocated tasks.
   - The new task is introduced in each position of the robot’s path, i.e., between task 1 and task 2, between task 2 and task 3 and so on.
   - For each insertion, the cost of the new path is calculated.
   - The bid for a task is the least increase in the cost of the robot’s path.

3. Each robot bids a bid vector, where each entry corresponds to the bid for one task and the length of the vector is equal to the number of unallocated tasks. Alternatively, robots can place only one numeric bid which represents the highest bid in their bid vector. In both cases, the resulting allocation is the same, but in the second case, the auctioneer has to process fewer bids.

4. The auctioneer allocates the task with the highest bid among all the received bids.

The previous steps are repeated until there are no more unallocated tasks.

---

3 [19] does not specify for how long the auction is open
Comparison Criteria

1. **Quality of solution.** In [22], the authors show that when SSI is used with the MINISUM team objective, the solution is a constant factor away from the optimal solution. Optimality is evaluated using a multi-robot routing application. The MINISUM team objective is to minimize the sum of the cost of the paths of all robots. If all robots have a complete map of the environment and the cost of moving one distance unit is equal among all robots, the sum of the cost of the paths of all robots is at least 1.5 and at most 2 away from the optimum [19].

2. **Communication requirements.** For allocating one task, the set of all unallocated tasks in broadcasted to all robots. The robots can submit their bids in different ways. In [25] each robot sends a vector of bids which contains a bid for each unallocated task. In [19], each robot either submits one bid for each task or just one bid (the highest of all its bids). In all cases, the resulting allocation is the same, but the number of bids varies. Considering that the auctioneer sends one message and each robot responds with one message (either a vector of bids or a single bid), the communication overhead is $O(n)$ per iteration, i.e. $O(nm)$ for $m$ iterations.

3. **Computation requirements.** Polynomial if the bids are calculated using the cheapest insertion heuristic [19].

4. **Scalability.** The algorithm is expected to scale well to larger number of tasks and robots because of its polynomial run-time [21].

5. **On-line task allocation.** All tasks can be known beforehand, or tasks can be introduced at run-time [21].

6. **Fault tolerance capabilities.** If a robot cannot accomplish one of its tasks, all robots react to all their tasks. This reaction scheme was used in a multi-robot routing application [19] where robots do not have a full map of the environment. If a robot encounters an obstacle, it react to all tasks so that another robot not impaired by the obstacle can accomplish it. All robots react to all of their tasks to allow the new allocation to exploit task synergies.
7. **Priority task allocation.** The algorithm does not take task priorities into account when making allocations, but modifications could be made. As indicated by [9], some MRTA algorithms use rewards to indicate task priorities. A higher priority task has a bigger reward, and robots consider rewards when computing their bids. However, because other metrics are considered in the bid calculation, it is not guaranteed that higher priority tasks are assigned first. A lower priority task could receive a higher bid and thus be assigned first.

8. **Heterogeneity.** The original paper [21] and following ones [22, 19] only consider scenarios where the tasks and robots are homogeneous. However, one of the benefits of auction-based methods is that they can account for heterogeneity in the bid calculation [11]. Modifications in the bidding scheme would need to be made to use SSI in heterogeneous scenarios.

9. **Validation in real-world applications:** In [21] the authors test the algorithm in a multi-robot simulator called Teambots and compare it against two other auction-based algorithms. The results show that the total cost of SSI is close to the optimal value and smaller than the guarantee of twice as bad as the optimal allocation.

10. **Special characteristics:** No special characteristics.

### 3.3.3 Temporal Sequential Single-Item Auctions (TeSSI and TeSSIduo)

Proposed in [25], this algorithm deals with the problem of allocating tasks that need to be performed within a time window. Each robot builds a schedule of tasks using a simple temporal network (STN) to represent the time windows of its allocated tasks. According to Gerkey and Matarić’s taxonomy it is an ST-SR-TA algorithm, and according to the MRTA-TOC taxonomy, it is an ST-SR-TW-HC deterministic algorithm.

It is based on the sequential single-item algorithm (SSI), but instead of checking for the path with the lowest cost, it checks for the schedule with the lowest makespan. The makespan is the difference between the start time of the first task and the end
time of the last task. It uses an insertion algorithm, similar to SSI, to produce compact schedules. A robot can allocate a new task between two of its already allocated tasks, and tasks can move around in the schedule as long as the temporal constraints are not violated. The Floyd-Warshall algorithm is used to check if the STN is consistent.

TeSSI uses the makespan team objective. The goal of TeSSI is to produce allocations that do not violate the temporal constraints and that minimize the makespan. The authors propose a variation of TeSSI, called TeSSIIduo, which uses as team objective a combination of the makespan and the total distance traveled.

Tasks with time constraints are defined using the parameters:

- $ES_t$: Earliest start time, where $ES_t \leq LS_t$
- $LS_t$: Latest start time
- $EF_t$: Earliest finish time, where $EF_t = ES_t + DUR_t$
- $LF_t$: Latest finish time, where $LF_t = LS_t + DUR_t$
- $DUR_t$: Duration of the task

The time window is denoted as $[ES_t, LF_t]$.

Each robot builds a simple temporal network (STN) based on a set of constraints and two time points, namely the start time $S_t$ and the finish time $F_t$ of its allocated tasks.

Time points:

- $S_t = [ES_t, LS_t]$: The start time of a task can take place between the earliest start time and the latest start time.
- $F_t = [EF_t, LF_t]$: The finish time of a task can take place between the earliest finish time and the latest finish time.

Constraints between time points in the network:

- Duration constraint: The start time of a task should always occur before its finish time.
• Travel time constraint: A robot cannot start a new task before finishing its previous task and moving to the location of the new task.

Figure 3.1 shows a Simple Temporal Network for three tasks. TeSSI and TeSSIduo work as follows:

1. The auctioneer sends a set of unallocated tasks to all the robots in the fleet.
2. Each robot computes a bid for each task based on its current schedule:
   • The robot loops through the number of tasks in the schedule ($i = 0, ... m$):
     – Inserts the task in position $i$
     – Adds the time points and constraints of the task to its STN.
     – Propagates the STN using the Floyd Warshall algorithm to check for inconsistencies in the network.
     – If the STN is consistent, it calculates the makespan.
     – If the makespan is the smallest so far, it saves the makespan and the insertion position.
     – Returns the STN to its previous state, before adding the new task.
   • The bid for a task is the minimum resulting makespan of adding the task to the robot’s schedule.
   • The bid for each unallocated task is added to a bid vector.
3. Each robot sends its bid vector to the auctioneer.
4. The auctioneer allocates the task that has the highest bid (lowest bid value).

The algorithm repeats until there are no more tasks to allocate and it is re-triggered when a new task or set of tasks enters the system. Like in the case of SSI, instead of sending the complete bid vector, robots can send their lowest bid.

Comparison Criteria

1. Quality of solution. TeSSI and TeSSIduo always terminate, but there are no guarantees of finding an optimal solution [25].
2. **Communication requirements.** In each iteration, the auctioneer sends a message with the set of unallocated tasks, and each robot responds with a bid vector which contains a bid for each unallocated task [25]. The communication overhead is $O(n)$ per iteration, i.e. $O(nm)$ for $m$ iterations.

3. **Computation requirements.** The algorithm runs in polynomial time [25].

4. **Scalability.** TeSSI and TeSSIduo are expected to scale well due to its polynomial run-time.

5. **On-line task allocation.** It is suitable for off-line as well as for on-line allocations [25].

6. **Fault tolerance capabilities.** TeSSI does not include a monitoring mechanism, nor does it describes what to do when a robot fails. When there is no capable robot available for performing a task, the task is added to an unallocated task set [25]. Modifications could be made to add a monitoring mechanism, similar to MURDOCH and to reauction unallocated tasks in a later iteration of the algorithm.

7. **Priority task allocation.** TeSSI and TeSSIduo do not consider task priorities when allocating tasks. However, like sequential single-item auctions, modifications could be made to assign higher rewards to higher priority tasks. These rewards can be considered in the bid calculation, making robots place a higher bid for a higher priority task. However, because other metrics are considered in the bid calculation, it is not guaranteed that higher priority tasks
are assigned first. A lower priority task could receive a higher bid and thus be assigned first [9].

8. **Heterogeneity.** The original paper [25] only considers scenarios with homogeneous robots. However, one of the benefits of auction-based methods is that they can account for heterogeneity in the bid calculation [11]. Similar to SSI, modifications in the bidding scheme would need to be made to use TeSSI and TeSSIduo in heterogeneous scenarios.

9. **Validation in real-world applications.** The algorithm was tested in a simulated environment, but to the best of our knowledge, it has not been validated in physical robots. In the experiments conducted in [25], the algorithm was evaluated in off-line as well as on-line scenarios and its performance was compared against a greedy algorithm and the consensus-based bundle auction algorithm (CBBA). The experiments considered a variable number of robots, tasks, batches of incoming tasks and randomly distributed and clustered tasks. The results look promising. When tasks are dynamically introduced into the system, and the number of new tasks in a newly introduced batch is less than 5, TeSSI and the greedy algorithm have the same performance. However, when batches of 5 or more tasks are dynamically introduced, TeSSI performs better than the greedy algorithm. This means that TeSSI and TeSSIduo can exploit the synergies between tasks even when the batches of new incoming tasks are small [25].

10. **Special characteristics.** TeSSI and TeSSIduo can allocate tasks that need to be performed within a time window. Each robot represents its schedule using a simple temporal network (STN). The Floyd Warshall algorithm is used for testing inconsistencies in the schedule. Time windows are temporal hard constraints, i.e., the algorithm allocates tasks only if they do not violate the constraints [25].

### 3.4 Selection of MRTA Algorithms

The transportation of hospital supply carts does not require heterogeneous robots, and all the tasks have the same requirements. However, it is preferable to have an
algorithm that can extend to more complex scenarios. We consider algorithms that do not account for heterogeneity of tasks, but that could be modified to allocate heterogeneous tasks to heterogeneous robots. We also consider algorithms that could be modified to perform priority task allocation. Since CBPAE requires task execution information and our experimental setup does not perform task execution, we did not select CBPAE for the experimental comparison.

### 3.4.1 MURDOCH

This algorithm is selected based on the following arguments:

- **Quality of solution.** Because of its greedy nature, it does not guarantee to find an optimal solution, but the utility of its solution is no less than $1/3$ of the utility of the optimal solution [15].

- **Communication requirements.** The communication overhead is linear in the number of robots [14].

- **Computation requirements.** The number of bids processed by the auctioneer grows linearly with the number of robots in the fleet [15].

- **Scalability.** Based on the communication and computation requirements, the algorithm is expected to scale well.

- **On-line task allocation.** It is suitable for allocating tasks introduced at runtime [13].

- **Fault tolerance capabilities.** From the algorithms analyzed, it is the only one that includes a monitoring phase. The auctioneer monitors the progress of the ongoing tasks and reacts to failures and poor performance progress [13].

- **Heterogeneity.** It is capable of allocating tasks with different requirements to a fleet of robots with different capabilities [13].

- **Validation in real-world applications.** It has been validated in real-world scenarios.
• **Battery monitoring.** From the algorithms analyzed, it is the only one that considers a battery monitoring mechanism.

### 3.4.2 Sequential Single-Item Auctions (SSI)

This algorithm is selected based on the following arguments:

• **Quality of solution.** The team performance is guaranteed to be a constant factor away from the optimum [22].

• **Communication requirements.** The communication overhead is linear in the number of robots.

• **Computation requirements.** Polynomial if the bids are calculated using the cheapest insertion heuristic [19].

• **Scalability.** The algorithm is expected to scale well due to its polynomial run-time [21].

• **On-line task allocation.** It is suitable for allocating tasks introduced at runtime [21].

• **Priority task allocation.** Modifications are needed, but it is not guaranteed that higher priority tasks will be allocated first.

• **Heterogeneity.** Modifications are needed to make SSI allocate heterogeneous tasks to heterogeneous robots.

• **Validation in real-world applications.** The algorithm was designed and tested in multi-robot routing applications where the tasks consist on visiting a set of target locations [21]. Our use case is part of the same domain, and hence the algorithm is suitable for our requirements.

• **Considers synergies between tasks.** Single-item auctions are also used in other algorithms like [43] [13], and [10] but with the disadvantage that the synergies between tasks are not taken into account. In traditional single-item auction methods, a robot places a bid for a new task based on its current
position and does not consider that if it moves to one of its already allocated
targets, it might have a lower cost to move from there to the new target. SSI
takes into account its already allocated tasks when computing a new bid [21].

- **Superior algorithmic characteristics than other auction-based methods.** In [22], the authors show that when SSI is used with the MINISUM team objective, the solution is a constant factor away from the optimal solution. In a later paper [19] they compare the algorithm characteristics of sequential-single-item auctions, parallel single-item auctions and combinatorial auctions. Although combinatorial auctions provide better solution guarantees, their run-time is exponential and require an exponential number of bids. Sequential single-item auctions run in polynomial time if the cheapest insertion heuristic is used, the number of bids is \(|T| \times |R|\) in the worst case (where \(T\) is number of tasks, and \(R\) is number of robots) and their solution is just a constant factor away from the optimum. Parallel single-item auctions, have the same run-time and number of bids than sequential single-item auctions but their solution quality is unbounded [19].

### 3.4.3 Temporal Sequential Single-Item Auctions (TeSSI and TeSSIduo)

TeSSI, along with its variation TeSSIduo are selected based on the following arguments:

- **Quality of solution.** The solution is a constant factor away from the optimal solution [25].

- **Communication requirements.** The communication overhead is linear in the number of robots [25].

- **Computation requirements:** The algorithms run in polynomial time [25].

- **Scalability:** The algorithms are expected to scale well due to their polynomial run-time.
3.4. Selection of MRTA Algorithms

- **On-line task allocation.** TeSSI and TeSSIduo are suitable for allocating tasks introduced at runtime [25].

- **Priority task allocation.** Modifications are needed, but it is not guaranteed that higher priority tasks will be allocated first.

- **Heterogeneity.** Modifications are needed to make TeSSI and TeSSIduo allocate heterogeneous tasks to heterogeneous robots.

- **Validation in real-world applications.** The algorithms were tested in a simulated environment, where the tasks consisted in visiting a set of target locations [25].

- **Allocates tasks with temporal constraints.** Unlike the other analyzed algorithms, TeSSI and TeSSIduo are capable of allocating tasks that need to be performed within a time window. It produces compact and consistent schedules [25].

For the ROPOD project, the allocation schemes to be considered are:

1. Allocate a task so that it can be fulfilled as soon as possible.
2. Allocate a task so that it can be fulfilled at some specific time in the future, e.g., tomorrow at 9 am.

IA assignments instantaneously allocate tasks to the most suitable robot at the moment of auction closure. Hence, IA algorithms cover the first allocation scheme. TA with time window constraints algorithms deal with the second allocation scheme, where tasks have to be performed at some specific time in the future.

Table 3.1 summarizes the selection criteria. Fault tolerant capabilities are divided into monitoring of ongoing tasks and reallocation of dropped tasks.
### Table 3.1: Comparison of selected algorithms.

<table>
<thead>
<tr>
<th></th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI(duo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality of solution</td>
<td>Suboptimal but bounded to 3-competitive.</td>
<td>Suboptimal but bounded to a constant factor away from the optimum.</td>
<td>Suboptimal no solution bounds reported.</td>
</tr>
<tr>
<td>Communications</td>
<td>Linear in the number of robots.</td>
<td>Linear in the number of robots.</td>
<td>Linear in the number of robots.</td>
</tr>
<tr>
<td>requirements</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>requirements</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scalability</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>On-line allocation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Monitoring</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>of ongoing tasks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reallocation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>of dropped tasks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Priority</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>task allocation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heterogeneity</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Validation in</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>real world applications</td>
<td>Real robots + Simulation.</td>
<td>Simulation</td>
<td>Simulation</td>
</tr>
<tr>
<td>Battery monitoring</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Temporal constraints</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
3.4. Selection of MRTA Algorithms
One use case, a common experimental setup, and 10 experiments are used for conducting an experimental comparison of the algorithms selected in chapter 3.

4.1 Use Case: Transportation of Supply Carts in a Hospital

A multi-robot system operating in a hospital allocates transportation tasks to the robots in the fleet. The transportation tasks consist on delivering supply carts carrying medical equipment within the hospital facilities. Each task requires one robot and robots can only accomplish one task at a time. The supply carts are called mobidiks. A Fleet Management System coordinates the multi-robot system. The Fleet Management System has a Task Allocator responsible for assigning robots to transportation tasks.

The task allocator implements either MURDOCH, SSI, TeSSI or TeSSIduo. Incoming tasks have the following information:

- id: UUID string that uniquely identifies the task.
- cart_type: Type of the device to be transported. For the use case “transportation of supply carts”, the device_type is always “mobidik.”
- cart_id: Number that uniquely identifies the mobidik to be transported.
- team_robots_ids: IDs of the robots assigned to perform the task (initially None)
- earliest_start_time: Earliest time at which the robot should be at the pickup location.
4.2 Setup

The hardware and software used for running the experiments are as follows.

4.2.1 Hardware


- RAM: 16 GB
- CPU: Intel Core i7-8550U @ 1.80 Hz x 8
- Disk: 512 GB SSD
- Graphics: Intel UHD Graphics 620 (Kabylake GT2)

4.2.2 Software

- Operating System: Ubuntu 16.04 LTS
- Docker version 18.06.1-ce
- Docker-compose version 1.22.0
- Python 3.5.2
- PyZMQ version 17.1.2
- PyYAML version 3.11
- Pyre

1https://www.openstreetmap.org/about
2https://github.com/zeromq/pyre
Section 5 describes the implementation details.

4.3 Performance Metrics

For each of the experiments, the following performance metrics are reported:

- **Bids per robot.** Number of bids placed by each robot.
- **Total bids.** Number of bids placed by all robots.
- **Empty bids per robot.** Number of empty bids placed by each robot.
- **Total empty bids.** Number of empty bids placed by all robots.
- **Messages sent.** Number of messages sent by the auctioneer.
- **Messages received.** Number of messages received by the auctioneer.
- **Travel distance per robot.** Distance each robot has to travel to execute its allocated tasks.
- **Total travel distance.** Distance all robots have to travel to execute their allocated tasks.
- **Usage per robot.** Percentage task allocation per robot.
- **Robot usage.** Percentage of robots used for allocating all tasks.
- **Unsuccessful allocations.** Number of tasks that could not be allocated.
- **Allocation time per task.** Time between the task announcement and task allocation of each task.
- **Average time.** Average time needed for allocating one task.
- **Allocation time.** Time between the task announcement and task allocation of all tasks.
- **Total time.** Total time for running the experiment. Includes the preprocessing of tasks by the auctioneer before it announces them to the robots.
Additional metrics are reported for the algorithms that consider temporal constraints:

- **Makespan per robot**: Time each robot will take to execute its tasks.
- **Total makespan**: Time the robots in the fleet will take to execute all tasks.

We also recorded the previously allocated tasks per robot, i.e., the number of tasks the robot had in its schedule before allocating a new task; and the order of allocation. The complete recorded information, including the start time and end time of each allocated task, is in the CD attached to this report in the path docker_implementation/results. The results of the Task Allocator implementation (see section 5.1) are in the path task_allocation/test/results.

### 4.4 Experimental Design

The experiments described here are inspired by [21] and [25]. Tasks are either uniformly distributed or clustered in space, like in [21]. Tasks arrive all at once or in batches and experiments run with an increasing number of tasks and robots, like in [25]. Additionally, tasks are uniformly distributed or clustered in time, with overlapping or nonoverlapping time windows. We evaluate the performance of single-task single-robot instantaneous assignment (ST-SR-IA) and single-task single-robot time-extended assignment (ST-SR-TA) algorithms in static and dynamic environments. The experiments are applied to one of the use cases of ROPOD which deals with the transportation of supply carts in a hospital.

Changes in the environment can be caused by different factors, like an increase or decrease in the availability of robots, arrival of new tasks, changes in the spatial distribution of tasks, failure in the execution of allocated tasks, among others. For this evaluation, static and dynamic solely refer to the way tasks are introduced to the multi-robot system. In a static environment, all tasks are known upfront and are allocated before task execution begins. In a dynamic environment, new tasks are introduced at runtime and are allocated during task execution. In the literature, the static assignment is often referred to as “off-line allocation” and the dynamic assignment as “on-line allocation.” [25].
Chapter 4. Methodology

A time-extended assignment (TA) builds a task schedule for each robot. In the simpler case, there is no time specification at which tasks have to be executed. The schedule is just a list of tasks that a robot has to perform. One of the advantages of this kind of assignment is that it exploits the synergies between tasks because tasks with a strong relationship can be allocated to the same robot. For instance, a robot can be assigned to go to two target locations that are near each other. In an instantaneous assignment (IA), the algorithms only allocate one task at a time, which means that a new allocation is only possible when a robot has finished executing its last assigned task. When a new task is available, the algorithm allocates it to the most suitable robot at that time. Experiment 1 and 2 evaluate both approaches when all tasks are known beforehand and when tasks are either uniformly distributed or clustered in the environment.

Experiments 3 and 4 investigate the effect that the arrival of new tasks has on the quality of the allocation. Tasks can arrive either individually or in batches. TA can perform many-tasks to one-robot assignment, while IA can only do a one-task to one-robot assignment. These experiments examine how the size of incoming task batches and the size of task clusters affect the allocation. TA algorithms exploit the synergies between tasks, and thus, the quality of their allocation should be superior to the one of IA approaches. The size of the clusters varies to assess at which extend TA benefits from the distance relationship between tasks.

One case that can only be handled by TA is the situation in which tasks need to be executed within a time window. Introducing task constraints adds complexity to the problem. Time constraints can take on different forms, like synchronization, precedence and time windows [26]. IA approaches do not build schedules and hence cannot handle time constraints. A way of implementing time constraints with an IA approach would be to keep a task queue and make a task available sometime before its execution should start, but to the best of our knowledge, there are no guarantees that the robot will be at the picking location in time. According to the iTax taxonomy [20], IA approaches can handle no dependencies (ND), cross-schedule dependencies (XD) and complex dependencies (CD) but not in-schedule dependencies (ID), which require each robot to have a schedule of tasks.

In [26] they propose an extension of Gerkey and Matarić’s taxonomy [15]. The new taxonomy expands time-extended assignment to consider two kinds of con-
constraints, namely time window constraints (TW) and synchronization and precedence constraints (SP). Experiments 5 and 6 evaluate the performance of a TA algorithm and one variation of it, both presented in [26]. The algorithms are tested with overlapping and nonoverlapping time windows, clustered or uniformly distributed over time. One IA approach and one TA approach are also tested in experiment 5 and 6, even if their allocations are not in compliance with the time constraints.

Experiments 7 and 8 test the scalability of the algorithms when the number of robots is fixed and the number of tasks increases. Similarly, experiments 9 and 10 test the scalability when the number of tasks is fixed and the number of robots increases. Table 4.1 classifies the experiments in five categories: off-line spatial, on-line spatial, off-line temporal, off-line task scalability and off-line robot scalability.

<table>
<thead>
<tr>
<th>Type of experiment</th>
<th>Experiment</th>
<th>#Robots</th>
<th>#Tasks</th>
<th>#Datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off-line spatial</td>
<td>1: Tasks uniformly distributed in the map</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2: Tasks clustered in the map</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>On-line spatial</td>
<td>3: Tasks uniformly distributed in the map</td>
<td>4</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>4: Tasks clustered in the map</td>
<td>4</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Off-line temporal</td>
<td>5: Tasks uniformly distributed in time and space</td>
<td>4</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>6: Tasks clustered in time and space</td>
<td>4</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Off-line task scalability</td>
<td>7: Increasing number of tasks uniformly distributed in time and space</td>
<td>4</td>
<td>10, 20, 30, ... 100</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>8: Increasing number of tasks clustered in time and space</td>
<td>4</td>
<td>10, 20, 30, ... 100</td>
<td>10</td>
</tr>
<tr>
<td>Off-line robot scalability</td>
<td>9: Tasks uniformly distributed in time and space with increasing number of robots</td>
<td>10, 20, 30, ... 100</td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>10: Tasks clustered in time and space with increasing number of robots</td>
<td>10, 20, 30, ... 100</td>
<td>100</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.1: Overview of experiments
Chapter 4. Methodology

4.4.1 Assumptions

For all the experiments the following assumptions are made:

- Robots are homogeneous, i.e., they have the same capabilities.
- Robots are uniformly distributed in the map. Experiments 1 to 8 use the robot distribution shown in Figure 4.1a. Figure 4.1b shows the distribution of robots for experiments 9 and 10.
- Experiments start with a fixed number of robots, i.e., robots do not leave nor join the fleet after the experiment has begun.
- Robots have a complete map of the environment.
- Robots know their location and receive information about the pickup and delivery locations of incoming tasks.
- Robots move with a constant velocity of 1 m/s.

Experiments 1 to 10 evaluate the algorithms in Table 4.2. The quality of the allocations is assessed using the performance metrics defined in section 4.3.

TeSSI and TeSSIduo assign a start time to their allocated tasks. MURDOCH allocates one task per robot, and task execution is triggered as soon as a robot allocates a new task. SSI can assign more than one task per robot. Each SSI robot
4.4. Experimental Design

<table>
<thead>
<tr>
<th>MRTA allocation type</th>
<th>MRTA algorithm</th>
<th>Temporal constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>IA</td>
<td>MURDOCH</td>
<td>No</td>
</tr>
<tr>
<td>TA</td>
<td>SSI</td>
<td>No</td>
</tr>
<tr>
<td>TA</td>
<td>TeSSI</td>
<td>Yes</td>
</tr>
<tr>
<td>TA</td>
<td>TeSSIduo</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 4.2: Algorithms tested.

stores its allocations in a list. The first entry in the list is the first task that the robot will execute. MURDOCH and SSI allocations are not compliant with the time constraints. TeSSI and TeSSIduo only assign tasks that meet the time constraints.

TeSSIduo uses a dual objective bidding rule that combines makespan and distance information. The rule uses a constant $\alpha$ to weight the makespan of the robot and the increment in distance for performing the new task. In [25] the authors set $\alpha$ to 0.5. For our experiments, we set it to 0.1 to give more importance to the distance than to the makespan.

$$bid = (\alpha \times makespan) + (1 - \alpha) \times (increment \ in \ travel \ cost)$$

Experiments run once per dataset. Robot scalability tests run once per number of robots in the fleet. Table 4.3 shows the datasets used for the off-line experiments and Table 4.4 the datasets for on-line experiments.

4.4.2 Type of Experiments

Experiments are either off-line or online.

Off-line Experiments

- All tasks are sent at once to the auctioneer.
- All tasks are allocated before task execution.

On-line Experiments

- Tasks are sent in batches to the auctioneer.
- The auctioneer returns the allocations of a batch before receiving a new batch.
After the tasks in a batch are allocated, robots that allocated task(s), simulate task execution. That is, they update their current position to the delivery pose of the last task in their schedule.

4.5 Datasets
The datasets used for the experiments are classified into four categories:

- Spatial uniformly distributed tasks (SDU)
  1. SDU-TER: Spatial uniformly distributed tasks with number of tasks equal to the number of robots.
  2. SDU-TGR: Spatial uniformly distributed tasks with twice as many tasks as robots.

- Spatial clustered tasks (SDC)
  3. SDC-TER: Spatial clustered tasks with number of tasks equal to the number of robots.
  4. SDC-TGR: Spatial clustered tasks with twice as many tasks as robots.

- Temporal uniformly distributed tasks (TDU)
  5. TDU-TGR: Temporal uniformly distributed tasks with twice as many tasks as robots.
  6. TDU-ST: Temporal uniformly distributed tasks for testing task scalability. The number of tasks in each dataset increases by 10.
  7. TDU-SR: Temporal uniformly distributed tasks for testing robot scalability. The dataset contains 100 tasks.

- Temporal clustered tasks (TDC)
  8. TDC-TGR: Temporal clustered tasks with twice as many tasks as robots.
  9. TDC-ST: Temporal clustered tasks for testing task scalability. The number of tasks in each dataset increases by 10.
10. TDC-SR: Temporal clustered tasks for testing robot scalability. The dataset contains 100 tasks.

The tasks in the datasets follow the format described in section 4.1.

Task example:

```plaintext
id: 55e43a1a-758d-49b0-8f27-231185c37eb9
cart_type: mobidik
cart_id: ''
team_robot_ids: null
earliest_start_time: 1539599563000
latest_start_time: 1554995558630
estimated_duration: 1539599563000
pickup_pose:
  floor_number: 0
  id: ''
name: AMK_SDU-TER-1_X_1.06_Y.18.53
type: ''
waypoints: []
delivery_pose:
  floor_number: 0
  id: ''
name: AMK_SDU-TER-1_X_9.58_Y.10.46
type: ''
waypoints: []
priority: 5
robot_actions: {}
status:
  completed_robot_actions: {}
current_robot_actions: {}
estimated_task_duration: 1539599563000
status: unallocated
task_id: 55e43a1a-758d-49b0-8f27-231185c37eb9
```

Each task has spatial and temporal information:
4.5.1 Spatial Information

The spatial information of a task is the pickup and delivery pose, which are stored as Area objects. For the datasets defined in this project, the name of an area object follows the format:

\[
\text{organisation} + \text{dataset}_\text{id} + \text{x}_\text{coordinate} + \text{y}_\text{coordinate}
\]

where organisation is AMK = Agaplesion Markus Krankenhaus
Example: AMK_SDU-TER-1_X_1.06_Y_18.53, refers to a task located in (1.06, 18.53).
Tasks can be uniformly distributed or clustered in space.

**Spatial clusters:**

- The size of a cluster is measured by its radius.
- Pickup and delivery locations belong to the same cluster.
- Pickup and delivery locations are uniformly distributed inside the cluster.
- \(n_{tasks\_cluster}\): Number of tasks in the cluster.

4.5.2 Temporal Information

The temporal information of a task is defined by its time window \([\text{earliest}_\text{start\_time}, \text{latest}_\text{finish\_time}]\) where:

\[
\text{latest}_\text{finish\_time} = \text{latest}_\text{start\_time} + \text{estimated}_\text{duration}
\]

Temporal information is expressed as seconds since the epoch. “For Unix, the epoch is 1970” [1].

Tasks can be uniformly distributed or clustered in time. Tasks within a time cluster are separated by a fixed time interval and do not overlap, i.e., the start time of a task is \(x\) seconds away from the finish time of the previous task. Tasks that are clustered in time are also clustered in space. Since tasks need to be executed within some seconds from one another, it makes sense to distribute temporal clustered tasks within a spatial cluster.
4.5. Datasets

**Temporal clusters:**
- Time windows do not overlap.
- interval\_b\_tw: Time interval between time windows in a cluster.
- n\_tasks\_cluster: Number of tasks in a cluster.
- Pickup and delivery locations belong to the same cluster.
- Pickup and delivery locations are uniformly distributed inside the cluster.

4.5.3 Batched Datasets

Datasets SDU-TGR and SDC-TGR are split into batches of tasks. The resulting batches are called batched datasets. All tasks within a batch have the same earliest start time and latest start time but different duration. Batches within a batched dataset have the same size. For instance, batched dataset SDU-TGR-1\_batch\_size.4 is created by splitting dataset SDU-TGR-1 into batches of size 4.

4.5.4 Datasets Description

The datasets whose number of tasks depends on the number of robots were created for a fleet of 4 robots distributed in a $20m \times 20m$ map. Tasks in the datasets for scalability tests are distributed in a $100m \times 100m$ map.

1. **SDU-TER:** Spatial uniformly distributed tasks with number of tasks equal to the number of robots. Tasks are uniformly distributed in the map and have the same earliest start time and latest start time but different durations. Figure 4.2 shows an example of a dataset of type SDU-TER.

   - **Number of tasks:** 4
   - **Pickup and delivery locations:** Uniformly distributed in an obstacle free map of $20m \times 20m$.
   - **Estimated duration of a task:** Equivalent to the distance from the pickup to the delivery location, considering that robots move with a constant velocity of $1m/s$.
   - **Dataset start time:** 2018-10-28 10:40:07
Chapter 4. Methodology

- **Earliest start time of all tasks:** 2018-10-28 10:40:37, i.e., 30 seconds after the dataset start time.
- **Latest start time of all tasks:** Earliest start time + 5 seconds.
- **Number of SDU-TER datasets:** 5

2. SDU-TGR: Spatial uniformly distributed tasks with twice as many tasks as robots. Tasks are uniformly distributed in the map, and each dataset is split into batches of tasks. All tasks within a batch have the same earliest start time but different duration. Figure 4.3 shows an example of a dataset of type SDU-TGR.

- **Number of tasks:** 8
- **Pickup and delivery locations:** Uniformly distributed in an obstacle free map of $20m \times 20m$.
- **Estimated duration of a task:** Equivalent to the distance from the pickup to the delivery location, considering that robots move with a constant velocity of $1m/s$.
- **Dataset start time:** 2018-10-28 10:40:07
- **Earliest start time of the first batch:** 2018-10-28 10:40:37
- **Earliest start time of the following batches:** 30 seconds after the start time of the previous batch.
- **Latest start time of a task:** Earliest start time + 5 seconds.
- **Each dataset of this type is split into batched datasets of size 1, 2 and 4.**
- **Number of SDU-TGR datasets:** 5
- **Number of batched SDU-TGR datasets:** 15

3. SDC-TER: Spatial clustered tasks with number of tasks equal to the number of robots. Tasks are distributed in 2 clusters in the map, and each cluster contains 2 tasks. The pickup and delivery locations of a task belong to the same cluster. All tasks have the same earliest start time and latest start time but different duration. Figure 4.4 shows an example of a dataset of type SDC-TER.

- **Number of tasks:** 4
• **Pickup and delivery locations:** Clustered

• **Estimated duration of a task:** Equivalent to the distance from the pickup to the delivery location, considering that robots move with a constant velocity of 1m/s.

• **Dataset start time:** 2018-10-28 10:40:07

• **Earliest start time of all tasks:** 2018-10-28 10:40:37, i.e., 30 seconds after the dataset start time.

• **Latest start time of all tasks:** Earliest start time + 5 seconds.

• **Number of SDC-TER datasets:** 4

• The parameter that varies from dataset to dataset is the radius of the clusters:
  - SDC-TER-CR-1: radius of 1 m
  - SDC-TER-CR-2: radius of 2 m
  - SDC-TER-CR-3: radius of 3 m
  - SDC-TER-CR-4: radius of 4 m

4. **SDC-TGR:** Spatial clustered tasks with twice as many tasks as robots. Tasks are distributed in 2 clusters in the map, and each cluster contains 4 tasks. The pickup and delivery locations of a task belong to the same cluster. Each dataset is split into batches of tasks, and all tasks within a batch have the same earliest start time but different duration. Figure 4.5 shows an example of a dataset of type SDC-TGR.

• **Number of tasks:** 8

• **Pickup and delivery locations:** Clustered

• **Estimated duration of a task:** Equivalent to the distance from the pickup to the delivery location, considering that robots move with a constant velocity of 1m/s.

• **Dataset start time:** 2018-10-28 10:40:07

• **Earliest start time of the first batch:** 2018-10-28 10:40:37

• **Earliest start time of the following batches:** 30 seconds after the previous batch.

• Latest start time of a task: Earliest start time + 5 seconds.
Chapter 4. Methodology

- Each dataset of this type is split into batched datasets of size 1, 2 and 4.
- **Number of SDC-TGR datasets**: 4
- **Number of batched SDC-TGR datasets**: 12
- The parameter that varies from dataset to dataset is the radius of the clusters:
  - SDC-TGR-CR-1: radius of 1 m
  - SDC-TGR-CR-2: radius of 2 m
  - SDC-TGR-CR-3: radius of 3 m
  - SDC-TGR-CR-4: radius of 4 m

5. **TDU-TGR**: Temporal uniformly distributed tasks with twice as many tasks as robots. Tasks are uniformly distributed in space and time, and all tasks have the same duration. Figure 4.6 shows an example of a dataset of type TDU-TGR.

- **Number of tasks**: 8
- **Pickup and delivery locations**: Uniformly distributed in an obstacle free map of $20m \times 20m$.
- **Estimated duration of a task**: 4 seconds. The pickup location is uniformly distributed in the map, and the delivery location is chosen so that the duration of the task is 4 seconds, considering that robots move with a constant velocity of $1m/s$.
- **Dataset start time**: 2018-10-28 10:40:07
- **Earliest start time of a task**: Drawn from the interval $[\text{dataset start time} + 30\text{ s}, \text{dataset start time} + 120\text{ s}]$
- **Latest start time of a task**: Earliest start time + 5 seconds.
- **Number of TDU-TGR datasets**: 5

6. **TDU-ST**: Temporal uniformly distributed tasks for testing task scalability. Tasks are uniformly distributed in space and time. Each dataset has 10 tasks more than the previous dataset. Datasets are subsets of a dataset of 100 tasks, i.e., the 10 tasks of dataset TDU-ST-10 are the first 10 tasks of dataset TDU-ST-100. Figure 4.7 shows an example of a dataset of type TDU-ST.
4.5. Datasets

- **Number of tasks**: 10, 20, 30, ..., 100
- **Pickup and delivery locations**: Uniformly distributed in an obstacle free map of $20m \times 20m$.
- **Estimated duration of a task**: 4 seconds. The pickup location is uniformly distributed in the map, and the delivery location is chosen so that the duration of the task is 4 seconds, considering that robots move with a constant velocity of $1m/s$.
- **Dataset start time**: 2018-11-04 11:49:59
- **Earliest start time of a task**: Drawn from the interval [dataset start time + 30s, dataset start time + 120s]
- **Latest start time of a task**: Earliest start time + 5 seconds.
- **Number of TDU-ST datasets**: 10

7. **TDU-SR**: Temporal uniformly distributed tasks for testing robot scalability. Tasks are uniformly distributed in space and time in a $100m \times 100m$ map. Figure 4.8 shows an example of a dataset of type TDU-SR.

- **Number of tasks**: 100
- **Pickup and delivery locations**: Uniformly distributed in an obstacle free map of $20m \times 20m$.
- **Estimated duration of a task**: 4 seconds. The pickup location is uniformly distributed in the map, and the delivery location is chosen so that the duration of the task is 4 seconds, considering that robots move with a constant velocity of $1m/s$.
- **Dataset start time**: 2018-11-04 14:51:29
- **Earliest start time of a task**: Drawn from the interval [dataset start time + 30s, dataset start time + 120s]
- **Latest start time of a task**: Earliest start time + 5 seconds.
- **Number of TDU-SR datasets**: 1

8. **TDC-TGR**: Temporal clustered tasks with twice as many tasks as robots. Tasks are distributed in 2 temporal clusters. Tasks that belong to a temporal cluster are also part of a spatial cluster, i.e., there are 2 spatial clusters. Figure 4.9 shows an example of a dataset of type TDC-TGR.
Chapter 4. Methodology

- **Number of tasks:** 8
- **Pickup and delivery locations:** Tasks are distributed in 2 clusters of radius 4m.
- **Estimated duration of a task:** 4 seconds. The pickup location is uniformly distributed in the map, and the delivery location is chosen so that the duration of the task is 4 seconds, considering that robots move with a constant velocity of 1m/s.
- **Dataset start time:** 2018-10-28 10:40:08
- **Start time of the first task in the first cluster:** Drawn from the interval [dataset start time + 30s, dataset start time + 120s]
- **Start time of the first task in the following clusters:** Latest finish task of the last task in the previous cluster + number drawn from the interval [0s, 300s].
- **Earliest start time of a task:** interval between the previous task in the same cluster.
- **Latest start time of a task:** Earliest start time + 5 seconds
- **Number of TDC-TGR datasets:** 4
  - The parameter that changes from dataset to dataset is the interval between time windows.
    - TDC-TGR-ITW-1 : time interval 1 s
    - TDC-TGR-ITW-2 : time interval 2 s
    - TDC-TGR-ITW-3 : time interval 3 s
    - TDC-TGR-ITW-4 : time interval 4 s

9. **TDC-ST:** Temporal clustered tasks for testing task scalability. Tasks are distributed in 2 temporal and 2 spatial clusters. Each dataset has 10 tasks more than the previous dataset. Datasets are subsets of a dataset of 100 tasks, i.e., the 10 tasks of dataset TDC-ST-10 are the first 10 tasks of dataset TDC-ST-100. Figure 4.10 shows an example of a dataset of type TDC-ST.

- **Number of tasks:** 10, 20, 30, ..., 100
- **Pickup and delivery locations:** Tasks are distributed in 2 clusters of radius 4m.
4.5. Datasets

- **Estimated duration of a task**: 4 seconds. The pickup location is uniformly distributed in the map, and the delivery location is chosen so that the duration of the task is 4 seconds, considering that robots move with a constant velocity of $1m/s$.
- **Interval between time windows**: 10 seconds
- **Dataset start time**: 2018-11-04 11:50:06
- **Start time of the first task in the first cluster**: Drawn from the interval $[\text{dataset start time} + 30s, \text{dataset start time} + 120s]$
- **Start time of the first task in the following clusters**: Latest finish task of the last task in the previous cluster + number drawn from the interval $[0s, 300s]$.
- **Earliest start time of a task**: 10s away from the previous task in the same cluster.
- **Latest start time of a task**: Earliest start time + 5 seconds.
- **Number of TDU-SR datasets**: 10
- The parameter that changes from dataset to dataset is the number of tasks in each dataset.

10. **TDC-SR: Temporal clustered tasks for testing robot scalability.** Tasks are distributed in 2 temporal and 2 spatial clusters in a $100m \times 100m$ map. Figure 4.11 shows an example of a dataset of type TDC-SR.

- **Number of tasks**: 100
- **Pickup and delivery locations**: Tasks are distributed in 2 clusters of radius $4m$.
- **Estimated duration of a task**: 4 seconds. The pickup location is uniformly distributed in the map, and the delivery location is chosen so that the duration of the task is 4 seconds, considering that robots move with a constant velocity of $1m/s$.
- **Interval between time windows**: 4 seconds
- **Dataset start time**: 2018-11-04 14:51:29
- **Start time of the first task in the first cluster**: Drawn from the interval $[\text{dataset start time} + 30s, \text{dataset start time} + 120s]$
• **Start time of the first task in the following clusters:** Latest finish task of the last task in the previous cluster + number drawn from the interval [0s, 300s].

• **Earliest start time of a task:** 4s away from the previous task in the same cluster.

• **Latest start time of a task:** Earliest start time + 5 seconds.

• **Number of TDC-SR datasets:** 1

Appendix B contains plots for all the datasets used in the experiments.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Datasets used</th>
<th>Experiment</th>
<th>Datasets used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SDU-TER-1</td>
<td>5</td>
<td>TDU-TGR-1</td>
</tr>
<tr>
<td></td>
<td>SDU-TER-2</td>
<td></td>
<td>TDU-TGR-2</td>
</tr>
<tr>
<td></td>
<td>SDU-TER-3</td>
<td></td>
<td>TDU-TGR-3</td>
</tr>
<tr>
<td></td>
<td>SDU-TER-4</td>
<td></td>
<td>TDU-TGR-4</td>
</tr>
<tr>
<td></td>
<td>SDU-TER-5</td>
<td></td>
<td>TDU-TGR-5</td>
</tr>
<tr>
<td>2</td>
<td>SDC-TER-CR-1</td>
<td>6</td>
<td>TDC-TGR-ITW-1</td>
</tr>
<tr>
<td></td>
<td>SDC-TER-CR-2</td>
<td></td>
<td>TDC-TGR-ITW-2</td>
</tr>
<tr>
<td></td>
<td>SDC-TER-CR-3</td>
<td></td>
<td>TDC-TGR-ITW-3</td>
</tr>
<tr>
<td></td>
<td>SDC-TER-CR-4</td>
<td></td>
<td>TDC-TGR-ITW-4</td>
</tr>
<tr>
<td>7</td>
<td>TDU-ST-10</td>
<td>8</td>
<td>TDC-ST-10</td>
</tr>
<tr>
<td></td>
<td>TDU-ST-20</td>
<td></td>
<td>TDC-ST-20</td>
</tr>
<tr>
<td></td>
<td>TDU-ST-30</td>
<td></td>
<td>TDC-ST-30</td>
</tr>
<tr>
<td></td>
<td>TDU-ST-40</td>
<td></td>
<td>TDC-ST-40</td>
</tr>
<tr>
<td></td>
<td>TDU-ST-50</td>
<td></td>
<td>TDC-ST-50</td>
</tr>
<tr>
<td></td>
<td>TDU-ST-60</td>
<td></td>
<td>TDC-ST-60</td>
</tr>
<tr>
<td></td>
<td>TDU-ST-70</td>
<td></td>
<td>TDC-ST-70</td>
</tr>
<tr>
<td></td>
<td>TDU-ST-80</td>
<td></td>
<td>TDC-ST-80</td>
</tr>
<tr>
<td></td>
<td>TDU-ST-90</td>
<td></td>
<td>TDC-ST-90</td>
</tr>
<tr>
<td></td>
<td>TDU-ST-100</td>
<td></td>
<td>TDC-ST-100</td>
</tr>
<tr>
<td>9</td>
<td>TDU-SR-100</td>
<td>10</td>
<td>TDC-SR-100</td>
</tr>
</tbody>
</table>

Table 4.3: Off-line experiments.
4.5. Datasets

### On-line experiments

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Datasets used</th>
<th>Experiment</th>
<th>Datasets used</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>SDU-TGR-1_batch_size1</td>
<td>4</td>
<td>SDC-TGR-CR-1_batch_size1</td>
</tr>
<tr>
<td></td>
<td>SDU-TGR-1_batch_size2</td>
<td></td>
<td>SDC-TGR-CR-1_batch_size2</td>
</tr>
<tr>
<td></td>
<td>SDU-TGR-1_batch_size4</td>
<td></td>
<td>SDC-TGR-CR-1_batch_size4</td>
</tr>
<tr>
<td></td>
<td>SDU-TGR-2_batch_size1</td>
<td></td>
<td>SDC-TGR-CR-2_batch_size1</td>
</tr>
<tr>
<td></td>
<td>SDU-TGR-2_batch_size2</td>
<td></td>
<td>SDC-TGR-CR-2_batch_size2</td>
</tr>
<tr>
<td></td>
<td>SDU-TGR-2_batch_size4</td>
<td></td>
<td>SDC-TGR-CR-2_batch_size4</td>
</tr>
<tr>
<td></td>
<td>SDU-TGR-3_batch_size1</td>
<td></td>
<td>SDC-TGR-CR-3_batch_size1</td>
</tr>
<tr>
<td></td>
<td>SDU-TGR-3_batch_size2</td>
<td></td>
<td>SDC-TGR-CR-3_batch_size2</td>
</tr>
<tr>
<td></td>
<td>SDU-TGR-3_batch_size4</td>
<td></td>
<td>SDC-TGR-CR-3_batch_size4</td>
</tr>
<tr>
<td></td>
<td>SDU-TGR-4_batch_size1</td>
<td></td>
<td>SDC-TGR-CR-4_batch_size1</td>
</tr>
<tr>
<td></td>
<td>SDU-TGR-4_batch_size2</td>
<td></td>
<td>SDC-TGR-CR-4_batch_size2</td>
</tr>
<tr>
<td></td>
<td>SDU-TGR-4_batch_size4</td>
<td></td>
<td>SDC-TGR-CR-4_batch_size4</td>
</tr>
<tr>
<td></td>
<td>SDU-TGR-5_batch_size1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDU-TGR-5_batch_size2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDU-TGR-5_batch_size4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4: On-line experiments.

(a) Spatial distribution of dataset SDU-TER-1. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset SDU-TER-1. Each square represents the time window of a task.

Figure 4.2: Dataset of type SDU-TER.
Chapter 4. Methodology

(a) Spatial distribution of dataset SDU-TGR-1. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset SDU-TGR-1_batch_size1. Each square represents the time window of a task.

(c) Temporal distribution of dataset SDU-TGR-1_batch_size2. Each square represents the time window of a task.

(d) Temporal distribution of dataset SDU-TGR-1_batch_size4. Each square represents the time window of a task.

Figure 4.3: Dataset of type SDU-TGR.
4.5. Datasets

(a) Spatial distribution of dataset SDC-TER-CR-2. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Spatial distribution of dataset SDC-TER-CR-4. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(c) Temporal distribution of dataset SDC-TER-CR-2. Each square represents the time window of a task.

(d) Temporal distribution of dataset SDC-TER-CR-4. Each square represents the time window of a task.

Figure 4.4: Dataset of type SDC-TER.
Chapter 4. Methodology

(a) Spatial distribution of dataset SDC-TGR-CR-3. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset SDC-TGR-CR-3_batch_size1. Each square represents the time window of a task.

(c) Temporal distribution of dataset SDC-TGR-CR-3_batch_size2. Each square represents the time window of a task.

(d) Temporal distribution of dataset SDC-TGR-CR-3_batch_size4. Each square represents the time window of a task.

Figure 4.5: Dataset of type SDC-TGR.
4.5. Datasets

(a) Spatial distribution of dataset TDU-TGR-1. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset TDU-TGR-1. Each square represents the time window of a task.

Figure 4.6: Dataset of type TDU-TGR.

(a) Spatial distribution of dataset TDU-ST-100. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset TDU-ST-100. Each square represents the time window of a task.

Figure 4.7: Dataset of type TDU-ST.
Chapter 4. Methodology

(a) Spatial distribution of dataset TDU-SR-100. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset TDU-SR-100. Each square represents the time window of a task.

Figure 4.8: Dataset of type TDU-SR.

(a) Spatial distribution of dataset TDC-TGR-ITW-1. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset TDC-TGR-ITW-1. Each square represents the time window of a task. Separation between time windows in a cluster is 1 second.

Figure 4.9: Dataset of type TDC-TGR.
4.5. Datasets

(a) Spatial distribution of dataset TDC-ST-100. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset TDC-ST-100. Each square represents the time window of a task. Separation between time windows in a cluster is 10 seconds.

Figure 4.10: Dataset of type TDC-ST.

(a) Spatial distribution of dataset TDC-SR-100. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset TDC-SR-100. Each square represents the time window of a task. Separation between time windows in a cluster is 4 seconds.

Figure 4.11: Dataset of type TDC-SR.
For conducting the experimental comparison of the selected MRTA algorithms, we created Python modules for each algorithm and scripts for generating the required datasets. Communication between the components of the multi-robot system is implemented via Zyre, a communication framework for local area networks which uses automatic peer discovery \(^1\). Particularly we have used Pyre, an implementation of Zyre for Python \(^2\). The algorithms MURDOCH, SSI, TeSSI and TeSSIduo (a variation of TeSSI) have been fully implemented and tested using the experiments described in chapter 4.

Our solution benefits from the work done in the ROPOD project and builds on it by integrating multi-robot task allocation to the project. The components used from the ROPOD project are:

- Pyre Base Communicator: For enabling communication between Pyre nodes \(^3\).
- Data structures defined in the Fleet Management System: For creating tasks and areas in the map \(^4\).

Both of the above-mentioned repositories are work in progress and are private.

\(^1\)https://github.com/zeromq/zyre
\(^2\)https://github.com/zeromq/pyre
\(^3\)https://git.ropod.org/ropod/ropod_common
\(^4\)https://git.ropod.org/ropod/ccu/fleet-management
5.1. Task Allocator Implementation

The algorithms were implemented using two approaches. The first approach uses a Python script to launch all the Pyre nodes, while the second approach uses a Docker container for each Pyre node. We refer to the first approach as Task Allocator implementation and to the second approach as Docker implementation.

5.1 Task Allocator Implementation

The Task Allocator approach was implemented first. The UML diagram in Figure 5.1 shows the class structure of the system.

5.1.1 Components Description:

- **Pyre**: “Python port for Zyre” [30].

- **PyreBaseCommunicator**: Base class for enabling communication between Pyre nodes in ROPOD.

- **PathPlanner**: Maps area names to Cartesian coordinates and returns the Cartesian distance between two area objects in the map.

- **Auctioneer**: Base class for the auctioneer nodes. Includes common functions and variables for the different MRTA approaches.

- **Robot**: Base class for the robot nodes. Includes common functions and variables for the different MRTA approaches.

- Each MRTA is implemented as a module, which contains an Auctioneer and a Robot class that inherit from the base Auctioneer and Robot classes.

- The TeSSIIduo module includes two Robot classes, namely RobotTeSSIIduo1 and RobotTeSSIIduo2. The difference between them is the data structure used for storing the Simple Temporal Network (STN) and the way the STN is updated. RobotTeSSIIduo1 stores the STN as a list of lists and creates it from scratch every time the robot calculates the cost for adding a new task to its schedule. The STN implementation as a list of lists is based on the public repository [8],

\[\text{https://www.docker.com/}\]
which implements the Floyd-Warshall algorithm. RobotTeSSIduo2 stores the STN as a numpy array and keeps a copy of the current STN. Every time the robot calculates the cost for adding a new task to its schedule, the STN is updated to include the new task. The TaskAllocator decides which robot class to instantiate depending on the argument `stn_option` passed to the constructor.

- TeSSI and TeSSIduo are implemented in the same module. The AuctioneerTeSSIduo and RobotTeSSIduo1 or RobotTeSSIduo2 select the algorithm to use based on the argument `method` passed to the constructor.

- Module structure:
  - murdoch
    - `__init__.py`
    - `murdoch_auctioneer.py`
    - `murdoch_robot.py`
  - ssi
    - `__init__.py`
    - `ssi_auctioneer.py`
    - `ssi_robot.py`
  - tessi_duo
    - `__init__.py`
    - `tessi_duo_auctioneer.py`
    - `tessi_duo_robot1.py`
    - `tessi_duo_robot2.py`

- TaskAllocator: Instantiates the Auctioneer and the Robot classes depending on the method (MURDOCH, SSI, TeSSI or TeSSIduo) passed to the constructor. The number of robot instances depends on the configuration file passed to the TaskAllocator constructor. The Robot and Auctioneer objects receive the experiment name in their constructor.
5.2. Docker Implementation

- ExperimentInitiator: Creates an instance of the class TaskAllocator, passing in the experiment name, the stn_option (used for TeSSI and TeSSIduo), the configuration parameters (read from the config file), the method to be used (MURDOCH, SSI, TeSSI or TeSSIduo), the robot initial positions and the areas in the map. The optional argument verbose_mrta can be set to true to visualize debug information. The ExperimentInitiator triggers an experiment by passing to the TaskAllocator the dataset ID and the start time of the experiment to be performed. The ExperimentInitiator receives the experiment name and method as arguments in the command line.

5.1.2 Limitations

The TaskAllocator cannot launch more than 9 Pyre nodes at a time. The auctioneer and the robots are Pyre nodes, which means that the allocation is limited to 8 robots. This issue seems to be related to a limitation on the number of sockets that can be opened per process. Similar issues have been reported in [37, 31] and [32]. The recommendations indicate that a software architecture that avoids using multiple sockets per process is preferable. Therefore, we modified our software architecture and developed a second implementation based on Docker, which allows us to launch multiple nodes, each one in a Docker container.

5.2 Docker Implementation

Docker allows to package software and its dependencies in containers which can be easily deployed on diverse production environments. Several containers may run in the same hardware and communicate with one another. Unlike other virtualization tools, Docker containers do not require an own operating system but use protected portions of the operating system of the host. Because of this, they are light and do not eat up resources when idle [23]. Docker is used in the ROPOD project to pack the different components of the system so as to ease the deployment and testing of software.

The Docker Implementation approach uses the PathPlanner of the Task Allocator implementation. The UML diagram in Figure 5.2 illustrates the class structure of the
Chapter 5. Solution

Docker implementation. The PathPlanner is not included in the diagram, but the Robot and Auctioneer base classes import the PathPlanner from task_allocation.

The design of this implementation is very similar to the Task Allocator implementation, with some key differences.

5.2.1 Differences Between the Docker Implementation and the Task Allocator Implementation

1. The main difference is that the Docker implementation does not use a Python script to launch the auctioneer and the robots, but a docker-compose file to launch a container per robot and per auctioneer.

2. The TaskAllocator is no longer needed because the robots and the auctioneer are no longer launched by an individual Python script.

3. The scripts docker_robot.py and docker_auctioneer.py are used for instantiating one single robot and one single auctioneer.

4. The docker_robot.py instantiates a Robot object in its main function. The kind of Robot object to instantiate (MURDOCH, SSI, TeSSI or TeSSIduo) depends on the command line arguments:
   - method: Name of the MRTA algorithm.
   - ropod_id: Example “ropod_1”

5. The docker_auctioneer.py instantiates an Auctioneer object in its main function. The kind of Auctioneer object to instantiate (MURDOCH, SSI, TeSSI or TeSSIduo) depends on the command line argument:
   - method: Name of the MRTA algorithm.

6. docker_robot.py and docker_auctioneer.py also receive as command line arguments:
   - experiment: Name of the experiment to run.
5.3. Configuration Files

- - verbose: Optional argument to visualize debug information.

7. The docker-compose file uses `docker_auctioneer.py` for launching the auctioneer container and `docker_robot.py` for each robot container, i.e., if an experiment has 10 robots, the docker-compose file launches one container per robot.

8. The ExperimentInitiator is now a Pyre node, which sends the dataset ID and the start time via a JSON message\(^6\). The ExperimentInitiator is implemented as a docker container.

9. There is a compose file for each experiment and for each number of robots in the experiment. All compose files are in the folder `docker-compose_files`.

5.2.2 Limitations

The number of containers that a machine can launch depends on its hardware. As the number of containers increases, more RAM is needed. For our experiments, we launched a maximum of 102 containers (100 robots + 1 auctioneer + 1 experiment initiator) in a computer with 16G RAM. Depending on the number of computers available, some containers can be launched in one computer and others on another computer. Scalability depends on hardware availability.

5.3 Configuration Files

The experiments described in chapter 4 consider two setups, one with 4 robots and the other one with an increasing number of robots. The information needed for both setups is created via Python scripts and stored in the `config` and `config_scalability` folders.

5.3.1 Config Folder

Contains the information needed for running the experiments 1 to 8, i.e., the experiments with 4 robots.

\(^6\)https://json.org/
Chapter 5. Solution

- area_names.yaml: Mapping between area names and Cartesian coordinates for the initial position of 4 robots and for the datasets used in experiments 1 to 8.
- config.yaml: Contains the robot IDs of robots 1 to 4, the Zyre group and the Zyre message types.
- map.yaml: Dimensions of the map used for the experiments (20m × 20m).
- ropod_position.yaml: Initial robot positions of robots 1 to 4.

5.3.2 Config_Scalability Folder

Contains the information needed for running the experiments 9 and 10, i.e., the experiments with increasing number of robots (from 10 to 100 robots). The files contain information for running experiments with 100 robots. If less than 100 robots are needed, fewer lines from the files are read. For example, the experiments with 10 robots read the information of the first 10 robots in these files.

- area_names.yaml: Mapping between area names and Cartesian coordinates for the initial position of 100 robots and for the datasets used in experiments 9 and 10.
- config.yaml: Contains the robot IDs for robots 1 to 100, the Zyre group and the Zyre message types.
- map.yaml: Dimensions of the map used for the experiments (100m × 100m).
- ropod_position.yaml: Initial robot positions of robots 1 to 100.

For conducting our experiments, we have generated the config files once and used the same configuration to run experiments 1 to 8 and the same configuration for experiments 9 and 10. However, the config files are configurable, and if another experimental setup is needed, one can create it by changing the arguments passed to the scripts that generate the config files. Section A.2 describes the steps for creating the config files.

The installation steps, the instructions to create the configuration files and to run the experiments are in appendix A.
<table>
<thead>
<tr>
<th>Message type</th>
<th>Content</th>
<th>Sender</th>
<th>Recipient</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>Dataset ID</td>
<td>Experiment initiator</td>
<td>All nodes in group</td>
</tr>
<tr>
<td></td>
<td>Dataset start time</td>
<td></td>
<td>TASK-ALLOCATION</td>
</tr>
<tr>
<td></td>
<td>Batch id (for on-line experiments)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TASK-ANNOUNCEMENT</td>
<td>Allocation round number</td>
<td>Auctioneer</td>
<td>All nodes in group</td>
</tr>
<tr>
<td></td>
<td>One task (for MURDOCH)</td>
<td></td>
<td>TASK-ALLOCATION</td>
</tr>
<tr>
<td></td>
<td>or a dictionary of tasks(for SSI, TeSSI and TeSSIduo)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALLOCATION</td>
<td>ID of allocated task</td>
<td>Auctioneer</td>
<td>All nodes in group</td>
</tr>
<tr>
<td></td>
<td>ID of winning robot</td>
<td></td>
<td>TASK-ALLOCATION</td>
</tr>
<tr>
<td>BID</td>
<td>Task ID of bidden task</td>
<td>Bidding robot</td>
<td>Auctioneer</td>
</tr>
<tr>
<td></td>
<td>ID of bidding robot</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bid value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO-BID</td>
<td>ID of non-bidding robot</td>
<td>Non-Bidding robot</td>
<td>Auctioneer</td>
</tr>
<tr>
<td>SCHEDULE</td>
<td>Robot ID</td>
<td>Winning robot</td>
<td>Auctioneer</td>
</tr>
<tr>
<td></td>
<td>List of scheduled tasks</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Timetable (for TeSSI and TeSSIduo)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>Empty</td>
<td>Auctioneer</td>
<td>All nodes in group</td>
</tr>
<tr>
<td>CLEAN-SCHEDULE</td>
<td>Empty</td>
<td>Auctioneer</td>
<td>All nodes in group</td>
</tr>
<tr>
<td>ROBOT-POSITION</td>
<td>Robot ID</td>
<td>All Robots</td>
<td>Auctioneer</td>
</tr>
<tr>
<td>(used in on-line experiments)</td>
<td>Robot’s new position</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TERMINATE</td>
<td>Empty</td>
<td>Experiment initiator</td>
<td>All nodes in group</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TASK-ALLOCATION</td>
</tr>
</tbody>
</table>

Table 5.1: Zyre messages used in the allocation process.
5.4 Experimental Workflow

This section describes the workflow of the program when running an experiment. A Pyre node “shouts” a message if it sends it to all members of a group and it “whispers” a message if it sends it to one specific peer within the group. Zyre messages are sent as JSON messages and read as dictionaries by the recipient nodes. Table 5.1 describes the content of the messages used in the allocation process.

The Docker Implementation is used for running all experiments (1 to 10). The Task Allocator Implementation can run experiments 1 to 8. Note that the workflow of both implementations is very similar. The first steps, the last steps and the way the allocation of a new dataset is triggered differ from implementation to implementation. To make it easier to spot the differences, we have written them in italics.

5.4.1 Task Allocator Implementation

1. The experiment initiator receives the name of the experiment and the name of the algorithm via the command line. It reads the config files and creates a task allocator object.

2. The task allocator creates the auctioneer and the robot nodes based on the name of the algorithm passed to its constructor.

3. Based on the experiment name, the experiment initiator reads the dataset file names that will be used for the experiment and stores them in a list.

4. The experiment initiator pops the first dataset file name and reads its start time and dataset ID. In the case of on-line experiments, it reads the batches of the batched dataset and the start time of each batch.

5. The experiment initiator calls the function get_assignment from the task allocator and passes the start time and dataset ID as arguments. If the experiment is on-line, the experiment initiator passes the batched dataset ID, the batch ID and the start time of the batch.

6. The task allocator passes the start time and dataset ID (batched dataset ID, batch ID and batch start time in case of on-line experiments) to the auctioneer.
5.4. Experimental Workflow

7. The auctioneer reads the dataset that corresponds to the received dataset ID, stores the tasks in a list of unallocated tasks and sets the flag `self.allocate_next_task` to true.

8. The auctioneer announces unallocated tasks if the list of unallocated tasks is not empty and if the flag `self.allocate_next_task` is true. Task announcements are sent in a TASK-ANNOUNCEMENT message to all members of the TASK-ALLOCATION Pyre group.

9. After announcing a task or a dictionary of tasks \(^7\), the auctioneer sets the flag `self.allocate_next_task` to false.

10. When the robots receive a message of type TASK-ANNOUNCEMENT, they calculate their utility for performing the task(s) and send one bid to the auctioneer using a BID message \(^8\).

11. If the robots determine that they cannot perform the task(s) they send an empty bid to the auctioneer using the NO-BID message.

12. The auctioneer counts the number of BID and NO-BID messages received. If the sum of both kinds of messages is equal to the number of robots in the fleet, it calls the `elect_winner` function.

13. The auctioneer elects the winner by choosing the bid with the smallest value. Two types of ties could occur:

   - Two robots bid the same value for the same task.
   - Two tasks have the same bid value. Only happens if the algorithm announces more than one task per allocation process (like SSI, TeSSI, and TeSSIduo).

14. Resolving ties: If more than one task has the same bid, the auctioneer selects the task with the lowest task ID. If for that task, more than one robot has the same bid, the auctioneer selects the robot with the lowest ID.

---

\(^7\)MURDOCH announces just one task per allocation iteration. SSI, TeSSI, and TeSSIduo announce all unallocated tasks in each allocation iteration.

\(^8\)MURDOCH computes the utility for one task. SSI, TeSSI and TeSSIduo compute the utility for all received tasks and bid on the task with the smallest utility.
15. The auctioneer announces the winner to all members of the TASK-ALLOCATION group using an ALLOCATION message, which contains the robot ID of the winning robot and the ID of the allocated task.

16. When the winning robot reads its ID in the ALLOCATION message, it allocates the task. Each algorithm process its allocations differently:

- MURDOCH: The winning robot adds the new task to its allocated tasks and changes its status to unavailable.
- SSI: The winning robot updates its schedule to the schedule it bid on the previous iteration. The travel cost is updated by adding the cost of executing the new task.
- TeSSI: The winning robot updates its schedule to the schedule it bid on the previous iteration. The Simple Temporal Network is updated with the one used for calculating the winning bid in its previous iteration.
- TeSSIduo: The winning robot updates its schedule to the schedule it bid on the previous iteration. The Simple Temporal Network is updated with the one used for calculating the winning bid in its previous iteration. The travel cost is updated by adding the cost of executing the new task.

17. SSI, TeSSI and TeSSIduo send its updated schedule to the auctioneer using a SCHEDULE message.

18. The auctioneer sets the flag self.allocate_next_task to true, and the process repeats from step 8 to 17.

19. If the list of unallocated tasks is empty and the flag self.allocate_next_task is set to true, the allocation of a dataset has been completed. The auctioneer:

- Displays the allocations in the terminal.
- Stores the allocation performance metrics in a yaml file.
- If the experiment is off-line, it shouts a RESET message.
- If the experiment is on-line, it shouts a CLEAN-SCHEDULE message.
5.4. Experimental Workflow

- It sets the self.terminate_variable to true.

20. When a robot receives a RESET message, it resets all variables used for allocation. The robot position is its initial position. When a robot receives a CLEAN-SCHEDULE message, apart from resetting its allocation variables, it changes its position to the delivery location of the last task in its schedule. That is, the robot simulates that it has performed all tasks in its schedule.

21. When the auctioneer member variable self.terminate_variable is true, the experiment initiator passes in the next dataset ID and start time to the task allocator, and the process repeats from step 7 to 19.

22. If the list of dataset file names is empty, the experiment initiator shuts down the task allocator, which in turn shuts down the auctioneer and the robots.

5.4.2 Docker Implementation

1. The docker-compose file creates a container for each robot, for the auctioneer and for the experiment initiator. The robots, the auctioneer and the experiment initiator belong to the TASK-ALLOCATION Pyre group.

2. The robot containers and the auctioneer are launched in a terminal.

3. The experiment initiator container is launched in another terminal.

4. Based on the experiment name, the experiment initiator reads the dataset file names that will be used for the experiment and stores them in a list.

5. The experiment initiator pops the first dataset file name and reads its start time and dataset ID. In the case of on-line experiments, it reads the batches of the batched dataset and the start time of each batch.

6. If the experiment is off-line, the experiment initiator shouts the start time and the dataset ID using a START message. If the experiment is on-line, the experiment initiator shouts the batched dataset ID, the batch ID and the start time of the batch.
7. When the auctioneer receives a START message, it reads the dataset that corresponds to the received dataset ID, stores the tasks in a list of unallocated tasks and sets the flag self.allocate_next_task to true.

8. The auctioneer announces unallocated tasks if the list of unallocated tasks is not empty and if the flag self.allocate_next_task is true. Task announcements are sent in a TASK-ANNOUNCEMENT message to all members of the TASK-ALLOCATION Pyre group.

9. After announcing a task or a dictionary of tasks, the auctioneer sets the flag self.allocate_next_task to false.

10. When the robots receive a message of type TASK-ANNOUNCEMENT, they calculate their utility for performing the task(s) and send one bid to the auctioneer using a BID message.

11. If the robots determine that they cannot perform the task(s) they send an empty bid to the auctioneer using the NO-BID message.

12. The auctioneer counts the number of BID and NO-BID messages received. If the sum of both kinds of messages is equal to the number of robots in the fleet, it calls the elect_winner function.

13. The auctioneer elects the winner by choosing the bid with the smallest value. Two types of ties could occur:
   - Two robots bid the same value for the same task.
   - Two tasks have the same bid value. Only happens if the algorithm announces more than one task per allocation process (like SSI, TeSSI, and TeSSIduo).

14. Resolving ties: If more than one task has the same bid, the auctioneer selects the task with the lowest task ID. If for that task, more than one robot has the same bid, the auctioneer selects the robot with the lowest ID.

---

9. MURDOCH announces just one task per allocation iteration. SSI, TeSSI, and TeSSIduo announce all unallocated tasks in each allocation iteration.

10. MURDOCH computes the utility for one task. SSI, TeSSI and TeSSIduo compute the utility for all received tasks and bid on the task with the smallest utility.
5.4. Experimental Workflow

15. The auctioneer announces the winner to all member of the TASK-ALLOCATION group using an ALLOCATION message, which contains the robot ID of the winning robot and the ID of the allocated task.

16. When the winning robot reads its ID in the ALLOCATION message, it allocates the task. Each algorithm process its allocations differently:

- **MURDOCH**: The winning robot adds the new task to its allocated tasks and changes its status to unavailable.
- **SSI**: The winning robot updates its schedule to the schedule it bid on the previous iteration. The travel cost is updated by adding the cost of executing the new task.
- **TeSSI**: The winning robot updates its schedule to the schedule it bid on the previous iteration. The Simple Temporal Network is updated with the one used for calculating the winning bid in its previous iteration.
- **TeSSIduo**: The winning robot updates its schedule to the schedule it bid on the previous iteration. The Simple Temporal Network is updated with the one used for calculating the winning bid in its previous iteration. The travel cost is updated by adding the cost of executing the new task.

17. SSI, TeSSI and TeSSIduo send its updated schedule to the auctioneer using a SCHEDULE message.

18. The auctioneer sets the flag self.allocate_next_task to true, and the process repeats from step 8 to 17.

19. If the list of unallocated tasks is empty and the flag self.allocate_next_task is set to true, the allocation of a dataset has been completed. The auctioneer:

- Displays the allocations in the terminal.
- Stores the allocation performance metrics in a yaml file.
- If the experiment is off-line, it shouts a RESET message.
- If the experiment is on-line, it shouts a CLEAN-SCHEDULE message.
Chapter 5. Solution

- It shouts a DONE message.

20. When a robot receives a RESET message, it resets all variables used for allocation. The robot position is its initial position. When a robot receives a CLEAN-SCHEDULE message, apart from resetting its allocation variables, it changes its position to the delivery location of the last task in its schedule. That is, the robot simulates that it has performed all tasks in its schedule.

21. When the experiment initiator receives a DONE message, it sends the next dataset ID and start time to all members of the TASK-ALLOCATION group and the process repeats from step 7 to 19.

22. If the list of dataset file names is empty, the experiment initiator sends a TERMINATE message to all members of the TASK-ALLOCATION group.

23. The experiment terminates when all members of the TASK-ALLOCATION group receive the TERMINATE message.

5.4.3 UML Diagrams

UML Class Diagrams

Figure 5.1 shows the class structure of the Task Allocator implementation. Likewise, Figure 5.2 shows the class structure of the Docker implementation.

Figures 5.3, 5.4 and 5.5 show UML class diagrams for each algorithm in the Docker Implementation. Refer to Figure 5.2 to see where each algorithm fits in the whole software architecture. Note that some methods, like elect_winner and announce_task are shared between the different MRTA implementations.

UML Sequence Diagrams

Figure 5.6 shows the sequence that the allocation follows when MURDOCH is used. Since the Fleet Management System of ROPOD will have a component dedicated to monitoring task execution, we have not implemented the task monitoring components
of MURDOCH. Hence, our MURDOCH implementation is equivalent to a single-item auction algorithm as described in [11]. Moreover, because tasks and robots are homogeneous, robots do not filter messages based on task requirements.

Instead of closing an auction process after some fixed time, unavailable robots send a NO-BID message to the auctioneer to indicate that they are not eligible for the task. An allocation round closes as soon as the auctioneer receives a message from all the robots in the fleet. Adopting this approach reduces the robustness of the system but allows to compare the allocation times of MURDOCH, SSI, TeSSI, and TeSSIduo. If we had kept a fixed round auction time, the allocation time would have been dominated by the auction closure time.

MURDOCH robots send a NO-BID message if they already have an allocated task. Provided that there are no communication failures, SSI robots can always place a bid and thus, they never send a NO-BID message. Figure 5.7 shows the sequence that the allocation process follows when the SSI algorithm is used. As shown in Figure 5.8 TeSSI and TeSSIduo robots send a NO-BID message when the task cannot be allocated without violating the time constraints.

The four implemented algorithms mainly differ in their implementation of the compute_bid function. Chapter 3 describes each algorithm in detail.

5.5 ROPOD Integration

The ROPOD project uses a Fleet Management System to coordinate the multi-robot system.

Components of the Fleet Management System:

1. Task manager
2. Task planner
3. Path planner
4. Resource manager
5. Task allocator
6. Task monitoring
7. Task execution
The task allocator implements either TeSSI or TeSSIduo depending on the allocation method specified in the configuration file. The Fleet Management System performs the following steps for allocating a task:

1. The transportation of a task is triggered by sending a task request to the task manager.

2. Each task request received by the task manager contains:
   - user_id: ID number of the person making the request.
   - cart_type: Type of the device to be transported. For the use case “transportation of supply carts”, the device_type is always “mobidik.”
   - cart_id: Number that uniquely identifies the mobidik to be transported.
   - earliest_start_time: Earliest time at which the robot should be at the pickup location.
   - latest_start_time: Latest time at which the robot should be at the pickup location.
   - pickup_pose: Area in Open Street Map format, where the device to be transported is located.
   - delivery_pose: Area, in Open Street Map format, where the device should be delivered.
   - priority: The priority of a task can be high, normal, low or emergency.

3. The task manager requests a plan from the task planner. The task planner returns a list of high-level actions. For the transportation of supply carts, the actions to be performed are:
   - Go to the pickup location.
   - Dock supply cart.
   - Go to the delivery location.
   - Undock supply cart.

\[^1^1\text{https://www.openstreetmap.org/about}\]
5.5. ROPOD Integration

- Go to the charging station.

4. The task manager instantiates an object of type Task with the contents:

- id: UUID string that uniquely identifies the task.
- cart_type: Type of the device to be transported. For the use case “transportation of supply carts”, the device_type is always “mobidik.”
- cart_id: Number that uniquely identifies the mobidik to be transported.
- team_robots_ids: IDs of the robots assigned to perform the task (initially None)
- earliest_start_time: Earliest time at which the robot should be at the pickup location.
- latest_start_time: Latest time at which the robot should be at the pickup location.
- estimated_duration: Estimation of how long task execution will take.
- start_time: Time at which start execution will begin. This information is filled once the task is allocated.
- finish_time: Time at which task execution will terminate. This information is filled once the task is allocated.
- pickup_pose: Area in Open Street Map format, where the device to be transported is located.
- delivery_pose: Area, in Open Street Map format, where the device should be delivered.
- priority: The priority of a task can be high, normal, low or emergency.
- status: Describes the status of the task. The initial status is “unallocated.”
- robot_actions: Actions required to perform the task.

5. The task manager requests the resource manager to allocate the task.

6. The resource manager asks the task allocator to allocate the task. The task allocator has an object of type auctioneer.
7. The auctioneer advertises the task to the robots.

8. Each robot computes its utility for the task and communicates it to the auctioneer by placing bids.

9. The auctioneer allocates the task to the robot with the lowest bid.

10. The task allocator returns the allocation to the resource manager, and the resource manager returns it to the task manager.

11. The allocation is a dictionary where the key is the ID of the allocated task, and the value is the list of robots that will execute the task. TeSSI and TeSSIduo assign one robot per task.

12. The task manager changes the status of the task to “allocated”.

13. The task manager calls a function of the resource manager which returns the start time and finish time of an allocated task. The task manager uses this information to fill in the start time and finish time of the task.

14. Once the task manager knows which robot will execute the task, it fills in the robot actions using the plan generated by the task planner.

15. The task manager adds the task ID of the allocated task to the dictionary of scheduled tasks.

16. The task manager will dispatch the task at the start time indicated in the task.

The system has been tested with one virtual robot implemented as a Docker container. Since each robot Pyre node is implemented inside a Docker container, the system scales well with the number of robots. For our experiments described in chapter 4 we have launched a maximum of 102 Docker containers in a single computer. The auctioneer is also a Pyre node, but it is an attribute of the class Task Allocator.

Task allocation has been tested using one single task request, but the task allocator is able to allocate multiple tasks in a single allocation process. The allocation process consists of multiple rounds, where one task is allocated per round. We have integrated multi-robot task allocation to the system, but there is still some work to do:
- The task received by the task allocator does not contain its estimated duration. The task manager should request the path planner to fill this information before the task allocation process starts.

- Evaluate the performance of the system using a test that includes more tasks with different spatial and temporal distributions.

- Robots should update their current position. A possible solution is to use a Zyre node to update robot positions.

- Make use of the path planner functions to compute the robot’s utility for a task.

- Instead of waiting to receive a message from all robots, close an auction round after a fixed time, defined as `round_close_time`, has passed. The `round_close_time` will be defined in the configuration file.

- Use task priorities. Tasks should be split by priorities, and higher priority tasks should have a smaller `round_close_time`.

- Use execution information to reflect the suitability of robots for performing a task.

- In future stages of the project, it is possible that the task allocator will be implemented as a separate module.
Figure 5.1: UML Class Diagram for the Task Allocation Implementation.
Figure 5.2: UML Class Diagram for the Docker Implementation.
Figure 5.3: MURDOCH UML Class Diagram.
5.5. ROPOD Integration

Figure 5.4: SSI UML Class Diagram.
Figure 5.5: TeSSI and TeSSIduo UML Class Diagram.
5.5. ROPOD Integration

Figure 5.6: UML Sequence Diagram for MURDOCH in the Docker implementation.
Figure 5.7: UML Sequence Diagram for SSI in the Docker implementation.
Figure 5.8: UML Sequence Diagram for TeSSI and TeSSIduo in the Docker implementation.
6.1 Experiment 1: Off-line Allocation of Tasks Uniformly Distributed in the Map

6.1.1 Purpose of the Experiment

Investigate the quality of allocations when tasks are uniformly distributed in the map, and all tasks are known before execution.

6.1.2 Experimental Design Considerations

- The number of tasks is equal to the number of robots so that IA approaches do not have a disadvantage against TA approaches. IA approaches can only allocate one task per robot at a time. Consequently, in an off-line scenario, they cannot allocate all tasks if the number of tasks is bigger than the number of robots.

- Tasks are uniformly distributed in the map to prevent them from having strong positive synergies.

- Tasks have the same earliest start time and latest start time to make TA approaches with time constraints have a similar behavior than IA approaches. Since tasks need to be performed in parallel and the number of tasks is equal
to the number of robots, time constraints restrict allocation to one task per robot.

- Tasks have different durations, equal to the distance between their pickup and delivery locations (assuming a constant velocity of 1 m/s). That is, the makespans for executing the tasks are different.

### 6.1.3 Hypothesis

TA assignment approaches will take advantage of the positive synergies between tasks and provide a smaller total travel distance than IA approaches. All tasks have the same earliest start time and latest start time, which means that they need to be performed in parallel. Algorithms that consider time constraints, namely TeSSI and TeSSIduo, will allocate one task per robot. Because of its IA (instantaneous assignment) nature, MURDOCH will also allocate one task per robot. It will be interesting to see how the allocations between MURDOCH, TeSSI, and TeSSIduo differ. We expect TeSSIduo to optimize distances and thus provide a smaller total travel distance than MURDOCH and TeSSI. We expect SSI to allocate more than one task per robot and hence provide the allocations with the smallest total travel distances. Tasks are not mutually exclusive, therefore we expect all algorithms to allocate all tasks.

### 6.1.4 Results

(a) Experiment 1: Number of successful and unsuccessful allocations.

(b) Experiment 1: Number of messages sent and received by the auctioneer.

Figure 6.1: Experiment 1: Number of allocations and messages sent and received.
Chapter 6. Results

(a) Experiment 1: Distances that the robots will travel to execute their tasks.

(b) Experiment 1: Time the fleet will take to execute all tasks.

Figure 6.2: Experiment 1: Travel distances and makespan of the fleet.

(a) Experiment 1: TeSSI temporal distribution of tasks per robot.

(b) Experiment 1: TeSSIduo temporal distribution of tasks per robot.

Figure 6.3: Experiment 1: Temporal distribution of tasks for dataset SDU-TER-1.
6.1. Experiment 1: Off-line Allocation of Tasks Uniformly Distributed in the Map

(a) Distance: 82.87m, Allocations: 4, Robot usage: 100%, Time to allocate: 1.032s, Messages sent and received: 25

(b) Distance: 63.01m, Allocations: 4, Robot usage: 50%, Time to allocate: 1.041s, Messages sent and received: 29

(c) Distance: 78.28m, Makespan: 13.25s, Allocations: 4, Robot usage: 100%, Time to allocate: 1.046s, Messages sent and received: 29

(d) Distance: 78.28m, Makespan: 13.25s, Allocations: 4, Robot usage: 100%, Time to allocate: 1.040s, Messages sent and received: 29

Figure 6.4: Robot trajectories for dataset SDU-TER-1. Each rectangle represents a robot and the dots represent pickup and delivery locations.
Chapter 6. Results

(a) Distance: 81.16m, Allocations: 4, Robot usage: 100%, Time to allocate: 1.033s, Messages sent and received: 25

(b) Distance: 66.37m, Allocations: 4, Robot usage: 75%, Time to allocate: 1.033s, Messages sent and received: 29

(c) Distance: 91.25m, Makespan: 21.98s, Allocations: 4, Robot usage: 100%, Time to allocate: 1.042s, Messages sent and received: 29

(d) Distance: 81.16m, Makespan: 21.98s, Allocations: 4, Robot usage: 100%, Time to allocate: 1.041s, Messages sent and received: 29

Figure 6.5: Robot trajectories for dataset SDU-TER-2. Each rectangle represents a robot and the dots represent pickup and delivery locations.
6.1. Experiment 1: Off-line Allocation of Tasks Uniformly Distributed in the Map

(a) Distance: 80.85m, Allocations: 4, Robot usage: 100%, Time to allocate: 1.026s, Messages sent and received: 25

(b) Distance: 66.58m, Allocations: 4, Robot usage: 75%, Time to allocate: 1.033s, Messages sent and received: 29

(c) Distance: 80.39m, Makespan: 16.55s, Allocations: 4, Robot usage: 100%, Time to allocate: 1.044s, Messages sent and received: 29

(d) Distance: 80.85m, Makespan: 16.55s, Allocations: 4, Robot usage: 100%, Time to allocate: 1.028s, Messages sent and received: 29

Figure 6.6: Robot trajectories for dataset SDU-TER-3. Each rectangle represents a robot and the dots represent pickup and delivery locations.


Chapter 6. Results

(a) Distance: 70.12m, Allocations: 4, Robot usage: 100%, Time to allocate: 1.019s, Messages sent and received: 25

(b) Distance: 62.75m, Allocations: 4, Robot usage: 50%, Time to allocate: 1.036s, Messages sent and received: 29

(c) Distance: 78.99m, Makespan: 14.48s, Allocations: 4, Robot usage: 100%, Time to allocate: 1.039s, Messages sent and received: 29

(d) Distance: 73.76m, Makespan: 14.48s, Allocations: 4, Robot usage: 100%, Time to allocate: 1.027s, Messages sent and received: 29

Figure 6.7: Robot trajectories for dataset SDU-TER-4. Each rectangle represents a robot and the dots represent pickup and delivery locations.
6.1. Experiment 1: Off-line Allocation of Tasks Uniformly Distributed in the Map

(a) Distance: 78.88m, Allocations: 4, Robot usage: 100%, Time to allocate: 1.028s, Messages sent and received: 25

(b) Distance: 73.17m, Allocations: 4, Robot usage: 75%, Time to allocate: 1.044s, Messages sent and received: 29

(c) Distance: 89.9m, Makespan: 17.13s, Allocations: 4, Robot usage: 100%, Time to allocate: 1.042s, Messages sent and received: 29

(d) Distance: 78.88m, Makespan: 17.13s, Allocations: 4, Robot usage: 100%, Time to allocate: 1.033s, Messages sent and received: 29

Figure 6.8: Robot trajectories for dataset SDU-TER-5. Each rectangle represents a robot and the dots represent pickup and delivery locations.

6.1.5 Analysis of Results

Figure 6.1a shows that the four algorithms allocated all tasks for all datasets. Figure 6.1b compares the number of messages sent and received by each algorithm. The four algorithms sent the same amount of messages, but SSI, TeSSI, and TeSSIduo received more messages than MURDOCH for allocating the same amount of tasks.
Note that with MURDOCH robots do not send their updated schedule to the auctioneer upon allocation of a new task, while the other three algorithms do.

Figure 6.4 shows that for dataset SDU-TER-1, TeSSI and TeSSIIduo provided the same allocations, with a total distance of 78.28m and a makespan of 13.25s. Figure 6.3 corroborates this information by showing the temporal distribution of tasks per robot for dataset SDU-TER-1. TeSSI bids the makespan for executing a task, while TeSSIIduo incorporates distance information to its bid calculations. In the case of dataset SDU-TER-1, TeSSI and TeSSIIduo yielded the same allocations due to the spatial distribution of tasks. In the case of dataset SDU-TER-2, TeSSIIduo provided an allocation with a shorter travel distance, as illustrated in Figure 6.5. Figure 6.7 shows that TeSSI did not allocate to robot 3 its nearest task, but TeSSIIduo did.

Figure 6.2a compares the travel distance for the four algorithms in all datasets. As expected, SSI provided the allocations with the smallest total travel distance for all datasets because it allowed more than one allocation per robot. For datasets SDU-TER-1, 2, 3, and 5, SSI used 75% of the robots, and for dataset SDU-TER-4 it used 50%. SSI does not consider time constraints and will execute the tasks in the order they appear in the schedule list, without considering the earliest and latest start time of the tasks.

MURDOCH includes only one task in the TASK-ANNOUNCEMENT message, while SSI, TeSSI and TeSSIIduo include all unallocated tasks. That is, with MURDOCH, the robots have less information in each allocation round. TeSSI and TeSSIIduo receive all unallocated tasks, calculate bids for all of them and bid for the task with the lowest bid. MURDOCH’s quality of allocation depends on the order tasks are announced. This explains why for dataset SDU-TER-1, MURDOCH provided an allocation with a more considerable travel distance than TeSSI and TeSSIIduo, as shown in Figure 6.4a. From these results we formulate a new hypothesis: “If tasks had been announced in the order TeSSIIduo allocated its tasks, MURDOCH would have achieved the same allocation as TeSSIIduo”.

Figure 6.5 shows that MURDOCH provided the same allocation than TeSSIIduo for dataset SDU-TER-2. The raw results data, included in the CD attached to this report, confirm that MURDOCH announced the tasks in the same order as TeSSIIduo allocated them. Since MURDOCH assigns a task to its nearest robot and TeSSIIduo...
6.1. Experiment 1: Off-line Allocation of Tasks Uniformly Distributed in the Map

places smaller bids for the tasks in the proximities of a robot, both algorithms yielded the same allocation.

Figure 6.2b illustrates the makespan for TeSSI and TeSSIduo, which is the same for all datasets since both algorithms allocated all tasks to start at their earliest start time. Figures 6.6, and 6.8 show the robot trajectories for datasets SDU-TER-3 and SDU-TER-5. Note that for dataset SDU-TER-3, TeSSIduo’s allocation has a slightly larger travel distance than TeSSI’s allocation. (80.85m for TeSSIduo and 80.39m for TeSSI). One explanation for this is the tie-breaking rule. If two robots bid the same value for the same task, the task is assigned to the robot with the lowest ID. Since the allocation in an auction round affects the subsequent allocations, the tie-breaking rule has an impact on the quality of the allocations.

6.1.6 Conclusions

MURDOCH, TeSSI, and TeSSIduo did one-task to one-robot allocations. Regarding travel distance, TeSSIduo’s allocations were superior than MURDOCH’s in most cases because TeSSIduo uses information of all tasks in each allocation round. MURDOCH only has information of one task per allocation round. Thus, MURDOCH’s allocation quality depends on the order in which tasks are announced.

TeSSIduo optimizes distances while TeSSI only uses distance information to verify whether the time constraints are satisfied. This, however, does not mean that TeSSIduo always provides an allocation with smaller travel distance than TeSSI. Because of the tie-breaking rule, if two robots bid the same value for the same task, the task is assigned to the robot with the lowest ID. The allocation in one round affects the allocations on the following rounds, and thus the tie-breaking rule affects the quality of the allocations.

SSI does not consider time constraints and exploits the positive synergies between tasks, providing the allocations with the smallest travel distances. Makespans for TeSSI and TeSSIduo are equal because both algorithms allocated all tasks to start at their earliest start time.
Chapter 6. Results

6.2 Experiment 2: Off-line Allocation of Tasks Clustered in the Map

6.2.1 Purpose of the Experiment

Investigate the quality of the allocations when tasks are clustered in the map, and all tasks are known before execution.

6.2.2 Experimental Design Considerations

- The number of tasks is equal to the number of robots so that IA approaches do not have a disadvantage against TA approaches. IA approaches can only allocate one task per robot at a time. Consequently, in an off-line scenario, they cannot allocate all tasks if the number of tasks is bigger than the number of robots.

- Tasks are distributed in two clusters in the map to assess how much TA approaches optimize the travel distance when tasks have strong positive synergies.

- The radius of each cluster increases from dataset to dataset to evaluate at which extent TA approaches benefit from the distance relationship between tasks.

- Pickup and delivery locations of a task belong to the same cluster so that a robot assigned to a cluster stays within the cluster and potentially receives more than one task inside that cluster.

- Tasks have different durations, equal to the distance between their pickup and delivery locations (assuming a constant velocity of 1 m/s).

- Tasks have the same earliest start time and latest start time to make TA approaches with time constraints have a similar behavior than IA approaches. Since tasks need to be performed in parallel and the number of tasks is equal to the number of robots, time constraints restrict allocation to one task per robot.
6.2.3 Hypothesis

TA assignment approaches will take advantage of the synergies between tasks and provide a smaller total travel distance than IA approaches. As the size of the clusters increases, the difference between the travel distance provided by MURDOCH and SSI and the one provided by TeSSI and TeSSIduo will become smaller.

TeSSI and TeSSIduo will allocate one task per robot due to the time constraints. MURDOCH will also allocate one task per robot, but we expect the travel distance from MURDOCH to be larger than the one from TeSSI and TeSSIduo. SSI will provide the allocations with the smallest travel distance, but its allocations will violate the temporal constraints.

TeSSIduo makes a compromise between distance and makespan. Hence, we expect its makespan to be worse or equal to the makespan of TeSSI. Tasks are not mutually exclusive; therefore we expect all algorithms to allocate all tasks.

6.2.4 Results

(a) Experiment 2: Number of successful and unsuccessful allocations.

(b) Experiment 2: Number of messages sent and received by the auctioneer.

Figure 6.9: Experiment 2: Number of allocations and messages sent and received.
Chapter 6. Results

(a) Experiment 2: Distances that the robots will travel to execute their tasks.

(b) Experiment 2: Time the fleet will take to execute all tasks.

Figure 6.10: Experiment 2: Travel distances and makespan of the fleet.

(a) Experiment 2: TeSSI temporal distribution of tasks per robot.

(b) Experiment 2: TeSSIduo temporal distribution of tasks per robot.

Figure 6.11: Experiment 2: Temporal distribution of tasks for dataset SDC-TER-CR-1.

<table>
<thead>
<tr>
<th>Cluster radius [m]</th>
<th>Difference between MURDOCH’s and SSI’s travel distance [m]</th>
<th>Difference between TeSSI’s and TeSSIduo’s travel distance [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30.97</td>
<td>21.77</td>
</tr>
<tr>
<td>2</td>
<td>14.92</td>
<td>14.92</td>
</tr>
<tr>
<td>3</td>
<td>21.63</td>
<td>25.65</td>
</tr>
<tr>
<td>4</td>
<td>18.26</td>
<td>12.46</td>
</tr>
</tbody>
</table>

Table 6.1: Experiment 2: Difference on travel distance as the cluster radius increases.
6.2. Experiment 2: Off-line Allocation of Tasks Clustered in the Map

(a) Distance: 40.67m, Allocations: 4, Robot usage: 100%, Time to allocate: 1.035s, Messages sent and received: 25

(b) Distance: 9.7m, Allocations: 4, Robot usage: 25%, Time to allocate: 1.047s, Messages sent and received: 29

(c) Distance: 41.44m, Makespan: 1.2s, Allocations: 4, Robot usage: 100%, Time to allocate: 1.045s, Messages sent and received: 29

(d) Distance: 19.67m, Makespan: 5.62s, Allocations: 4, Robot usage: 50%, Time to allocate: 1.044s, Messages sent and received: 29

Figure 6.12: Robot trajectories for dataset SDC-TER-CR-1. Each rectangle represents a robot and the dots represent pickup and delivery locations.
Chapter 6. Results

Figure 6.13: Robot trajectories for dataset SDC-TER-CR-2. Each rectangle represents a robot and the dots represent pickup and delivery locations.
6.2. Experiment 2: Off-line Allocation of Tasks Clustered in the Map

Figure 6.14: Robot trajectories for dataset SDC-TER-CR-3. Each rectangle represents a robot and the dots represent pickup and delivery locations.

(a) Distance: 42.5m, Allocations: 4, Robot usage: 100%, Time to allocate: 1.037s, Messages sent and received: 25

(b) Distance: 20.87m, Allocations: 4, Robot usage: 50%, Time to allocate: 1.045s, Messages sent and received: 29

(c) Distance: 48.18m, Makespan: 2.97s, Allocations: 4, Robot usage: 100%, Time to allocate: 1.042s, Messages sent and received: 29

(d) Distance: 22.53m, Makespan: 6.28s, Allocations: 4, Robot usage: 50%, Time to allocate: 1.043s, Messages sent and received: 29
Chapter 6. Results

(a) Distance: 43.63m, Allocations: 4, Robot usage: 100%, Time to allocate: 1.035s, Messages sent and received: 25

(b) Distance: 25.37m, Allocations: 4, Robot usage: 50%, Time to allocate: 1.042s, Messages sent and received: 29

(c) Distance: 44.32m, Makespan: 5.58s, Allocations: 4, Robot usage: 100%, Time to allocate: 1.041s, Messages sent and received: 29

(d) Distance: 31.86m, Makespan: 7.44s, Allocations: 4, Robot usage: 75%, Time to allocate: 1.047s, Messages sent and received: 29

Figure 6.15: Robot trajectories for dataset SDC-TER-CR-4. Each rectangle represents a robot and the dots represent pickup and delivery locations.
6.2.5 Analysis of Results

Figure 6.9a shows that the four algorithms allocated all tasks for all the datasets. The number of messages sent is the same for all the algorithms, but MURDOCH’s auctioneer receives fewer messages than the auctioneers of the other approaches, as shown in Figure 6.9b. This is because SSI, TeSSI, and TeSSIduo send their updated schedule to the auctioneer after each allocation round.

Figure 6.10a shows that SSI provided the allocations with the smallest travel distances for all datasets, but SSI allocates all tasks of dataset SDC-TER-CR-1 to a single robot and uses only 50% of the robots to allocate each of the other datasets.

For all datasets, TeSSI’s makespan is smaller than TeSSIduo’s makespan, as shown in Figure 6.10b. Moreover, travel distances for TeSSI are larger than travel distances for TeSSIduo. For instance, Figure 6.10a shows that the TeSSI’s distance for the dataset SDC-TER-CD-4 is 12.46m larger than the TeSSIduo’s distance for the same dataset. The difference in distance and makespan can be explained by the time window robots have to start a task. Each task can be allocated to be executed at any time between its earliest and latest start time. For dataset SDC-TER-CD-4, TeSSI allocated one task per robot, as shown in Figure 6.15c, which means that all tasks will be executed at their earliest start time. Figure 6.15d reveals that for the same dataset, TeSSIduo allocated two tasks to robot 2. That is, robot 2 can perform both tasks without violating the temporal constraints, but the second task will not start at the earliest start time but some time between the earliest and latest start time, making the total makespan larger. The raw results data, included in the CD attached to this report, indicate the start time and finish time of each allocated task for dataset SDC-TER-CR-4.

Figure 6.11 shows the temporal distribution of tasks per robot for dataset SDC-TER-CR-1. TeSSI assigned a task per robot, and each task will start at its earliest start time, while TeSSIduo allocated three tasks to robot 4 and one task to robot 1. The task allocated to robot 1 will start at its earliest start time but the second and third tasks in robot’s 4 schedule will start at some time between their earliest and latest start time.

MURDOCH and TeSSI allocated one task per robot, while SSI and TeSSIduo
allocated more than one task to each robot. Figure 6.13 shows that for dataset SDC-TER-CR-2, MURDOCH’s and TeSSI’s allocation is the same and SSI’s, and TeSSIIduo’s allocation is the same. As mentioned in the results of experiment 1, ties are broken by assigning the task to the robot with the smallest ID. It is likely that this tie-breaking decision made TeSSI provide these allocations. SSI and TeSSIIduo optimize distances, which is why they provide the same allocation. Interestingly, SSI’s allocation does not violate time constraints, but because SSI does not define a start time for each allocated task, it is likely that tasks will not be executed between their earliest and latest start time.

Figures 6.12, 6.14 and 6.15 show the trajectories that robots will follow for executing their tasks.

6.2.6 Conclusions

The TA assignments that optimize travel distance, namely SSI and TeSSIIduo took advantage of the spatial relationships between tasks and provided allocations with smaller travel distances than MURDOCH. TeSSI did not provide allocations with shorter travel distances than MURDOCH because it does not use distance to bid on tasks.

We expected TeSSI and TeSSIIduo to allocate one task per robot. TeSSIIduo did not allocate one task per robot because the distance relationship between tasks affected the bids placed by the robots. Performing a task closer to a robot, but with a start time between the earliest start and latest start time was preferable than performing a task farther away from the robot but with a start time equal to the earliest start time. Because of this, a robot close to two tasks allocated the two tasks if it could make it to the pickup location anytime between the earliest and latest start time. This had the effect of providing allocations with smaller travel distances but larger makespans.

When the datasets for these experiments were generated, tasks were placed anywhere inside the cluster. A bigger cluster can allocate tasks in a larger space, but this does not guarantee that tasks will not be allocated just in a small area within the cluster. Because of this, it is difficult to draw conclusions on the effect of the cluster size on the allocations. If we compare the difference in distance between
6.3. Experiment 3: On-line Allocation of Tasks Uniformly Distributed in the Map

MURDOCH and SSI and between TeSSI and TeSSI duo (see Table 6.1) we do not see a clear trend on how this values change as the size of the clusters increase.

6.3 Experiment 3: On-line Allocation of Tasks Uniformly Distributed in the Map

6.3.1 Purpose of the Experiment

Evaluate the quality of the allocations when tasks are uniformly distributed in the map, and they are introduced in batches, i.e., not all tasks are known beforehand. The experiment runs with batches of size 1, 2 and 4.

6.3.2 Experimental Design Considerations

- To simulate the introduction of new tasks at run-time, the auctioneer receives a new batch of tasks at each iteration.
- All tasks within a batch have the same earliest start time but different duration.
- Tasks in a batch have earliest start times 30 seconds after the earliest start times of the previous batch. This is to simulate that batches are introduced every 30 seconds.
- There are 8 tasks in total, which are split into batches of 1, 2 and 4. That is, the experiment with batches of size 1 requires 8 batches, while the experiment with batches of size 2 requires 4 batches.
- The experiment runs once for each batch size. That is, for the experiment with batch size 1, batches of size 1 are introduced at each iteration. The experiment terminates after 8 batches have been allocated.
- The experiment does not perform the tasks but simulates their execution by updating the robot positions. That is, robots simulate the execution of their allocated tasks by updating their position to the delivery location of the last task in their schedule. The experiment assumes no errors in the task execution.
• In order to avoid IA approaches from having a disadvantage against TA approaches, the number of tasks in each batch is always less than the number of robots.

• Tasks are uniformly distributed in the map to prevent them from having strong positive synergies.

6.3.3 Hypothesis

When tasks are introduced one by one, the algorithms do not have enough information for optimizing the path between tasks, and thus, we expect MURDOCH, SSI, and TeSSIIduo to provide the same allocations. TeSSI will provide different allocations because it does not use the path distance to compute its bids. For batches with more than one task, we expect SSI and TeSSIIduo to provide allocations with shorter travel distances than MURDOCH. The difference between the travel distances provided by MURDOCH and the ones provided by SSI will increase as the size of the batches increase. Since robots become available before a new batch arrives, the four algorithms will be able to allocate all tasks.

6.3.4 Results

<table>
<thead>
<tr>
<th>Batch size</th>
<th>Difference between MURDOCH and SSI travel distance [m]</th>
<th>Difference between TeSSI and TeSSIIduo travel distance [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>35.57</td>
</tr>
<tr>
<td>2</td>
<td>10.61</td>
<td>41.27</td>
</tr>
<tr>
<td>4</td>
<td>39.03</td>
<td>11.75</td>
</tr>
</tbody>
</table>

Table 6.2: Experiment 3: Difference on travel distance as the batch size increases. Dataset: SDU-TGR-1.
6.3. Experiment 3: On-line Allocation of Tasks Uniformly Distributed in the Map

(a) Experiment 3: Number of allocations for batches of size 1.

(b) Experiment 3: Number of allocations for batches of size 2.

(c) Experiment 3: Number of allocations for batches of size 4.

Figure 6.16: Experiment 3: Number of allocations for task batches of sizes 1, 2 and 4.
Chapter 6. Results

(a) Experiment 3: Number of messages sent and received by the auctioneer for batches of size 1.

(b) Experiment 3: Number of messages sent and received by the auctioneer for batches of size 2.

(c) Experiment 3: Number of messages sent and received by the auctioneer for batches of size 4.

Figure 6.17: Experiment 3: Number of messages sent and received for task batches of sizes 1, 2 and 4.
6.3. Experiment 3: On-line Allocation of Tasks Uniformly Distributed in the Map

(a) Experiment 3: Sum of the distances that the robots traveled to executed their tasks for batches of size 1.

(b) Experiment 3: Sum of the distances that the robots traveled to executed their tasks for batches of size 2.

(c) Experiment 3: Sum of the distances that the robots traveled to executed their tasks for batches of size 4.

Figure 6.18: Experiment 3: Sum of the distances that the robots traveled to executed their tasks for batches of sizes 1, 2 and 4.
Chapter 6. Results

(a) Experiment 3: Time the fleet needed for executing all tasks for batches of size 1.

(b) Experiment 3: Time the fleet needed for executing all tasks for batches of size 2.

(c) Experiment 3: Time the fleet needed for executing all tasks for batches of size 4.

Figure 6.19: Experiment 3: Time the fleet needed for executing all tasks for batches of sizes 1, 2 and 4.
6.3. Experiment 3: On-line Allocation of Tasks Uniformly Distributed in the Map

(a) Experiment 3: TeSSI temporal distribution of tasks per robot for batches of size 1.

Figure 6.20: Experiment 3: Temporal distribution of tasks for dataset SDU-TGR-1 batch size 1.

(b) Experiment 3: TeSSIduo temporal distribution of tasks per robot for batches of size 1.

Figure 6.21: Experiment 3: Temporal distribution of tasks for dataset SDU-TGR-1 batch size 2.

(a) Experiment 3: TeSSI temporal distribution of tasks per robot for batches of size 2.

Figure 6.22: Experiment 3: Temporal distribution of tasks for dataset SDU-TGR-1 batch size 4.

(b) Experiment 3: TeSSIduo temporal distribution of tasks per robot for batches of size 4.

Figure 6.23: Experiment 3: Temporal distribution of tasks for dataset SDU-TGR-1 batch size 4.
Chapter 6. Results

(a) Distance: 101.07m, Allocations: 8, Robot usage: 75%, Avg. allocation time per batch: 0.259s, Messages sent and received: 55

(b) Distance: 101.07m, Allocations: 8, Robot usage: 75%, Avg. allocation time per batch: 0.259s, Messages sent and received: 63

(c) Distance: 136.64m, Makespan: 67.77s, Allocations: 8, Robot usage: 25%, Avg. allocation time per batch: 0.259s, Messages sent and received: 63

(d) Distance: 101.07m, Makespan: 67.77s, Allocations: 8, Robot usage: 75%, Avg. allocation time per batch: 0.26s, Messages sent and received: 63

Figure 6.23: Robot trajectories for dataset SDU-TGR-1_batch_size1. Each rectangle represents a robot and the dots represent pickup and delivery locations.
6.3. Experiment 3: On-line Allocation of Tasks Uniformly Distributed in the Map

(a) Distance: 109.24m, Allocations: 8, Robot usage: 100%, Avg. allocation time per batch: 0.518s, Messages sent and received: 51

(b) Distance: 98.63m, Allocations: 8, Robot usage: 75%, Avg. allocation time per batch: 0.519s, Messages sent and received: 59

(c) Distance: 149.68m, Makespan: 38.73s, Allocations: 8, Robot usage: 50%, Avg. allocation time per batch: 0.520s, Messages sent and received: 59

(d) Distance: 108.41m, Makespan: 38.73s, Allocations: 8, Robot usage: 100%, Avg. allocation time per batch: 0.520s, Messages sent and received: 59

Figure 6.24: Robot trajectories for dataset SDU-TGR-1_batch_size2. Each rectangle represents a robot and the dots represent pickup and delivery locations.
Chapter 6. Results

(a) Distance: 139.15m, Allocations: 8, Robot usage: 100%, Avg. allocation time per batch: 1.037s, Messages sent and received: 49

(b) Distance: 99.85m, Allocations: 8, Robot usage: 75%, Avg. allocation time per batch: 1.046s, Messages sent and received: 57

(c) Distance: 148.47m, Makespan: 25.39s, Allocations: 8, Robot usage: 100%, Avg. allocation time per batch: 1.044s, Messages sent and received: 57

(d) Distance: 136.72m, Makespan: 25.39s, Allocations: 8, Robot usage: 100%, Avg. allocation time per batch: 1.046s, Messages sent and received: 57

Figure 6.25: Robot trajectories for dataset SDU-TGR-1_batch_size4. Each rectangle represents a robot and the dots represent pickup and delivery locations.
6.3.5 Analysis of Results

Figure 6.16 shows that the four algorithms allocated all tasks for all batch sizes. Figure 6.17 shows that the auctioneer sent the same amount of messages for the four algorithms, but MURDOCH’s auctioneer received fewer messages than SSI, TeSSI and TeSSIduo for allocating the same amount of tasks. With MURDOCH, robots do not send their updated schedule to the auctioneer because they only have one task in their schedule.

The number of received messages decreases when the batch size increases but the number of sent messages remains constant. In all cases, 8 tasks are allocated. The auctioneer sends one TASK-ANNOUNCEMENT and one ALLOCATION message per task, which means that the auctioneer sends 16 messages for all batch sizes. As the batch size increases, more messages are received by the auctioneer per batch allocation, but there are less batch allocation iterations.

Figure 6.18 compares the travel distances of the four algorithms in the 5 datasets, using batches of size 1, 2 and 4. The distance represented in the plots is the sum of the distances traveled by the robots for performing their allocations in all batch allocation iterations. Allocating 8 tasks using batches of size 1 requires 8 batch allocation iterations while allocating the same 8 tasks using batches of size 4 only requires 2 batch allocation iterations. For instance, Figure 6.18c shows that with SSI, robot 1 traveled 30.03m for executing the tasks in dataset SDU-TGR-1_batch_size_4. The result files in the CD attached to this report show that robot 1 traveled 15.39m for executing the tasks allocated in the first batch iteration, and 14.64m for the tasks in the second iteration, giving a total of 30.03m. By comparing the total distances, we observe the effect of the batch size in the total distance that the fleet has to travel.

When each batch only contains one task, as in Figure 6.18a, MURDOCH, SSI and TeSSIduo provide the same allocations. This is because, in each iteration, the three algorithms have the same information and they use the distance to the task to calculate their bids. TeSSI provides a different allocation because it uses the makespan and not the distance in its bid calculations. Figure 6.18b shows that when each batch contains two tasks, SSI provides allocations with shorter travel distances.
than MURDOCH for dataset SDU-TGR-1\_batch\_size\_2.

The SSI travel distance for batch SDU-TGR-1\_batch\_size\_2 is 101.7m, for SDU-TGR-1\_batch\_size\_2 it is 98.63m, and for SDU-TGR-1\_batch\_size\_4 the travel distance is 99.85m. The distance for the batch of size 2 is smaller than the one for batch size 1 but larger than the one for batch size 4. That is, a bigger batch does not necessarily reduces the travel distance. The robot positions change after each batch allocation iteration. The change of robot positions might place the robots farther away from the tasks of the next batch, making the robots have an overall larger travel distance. Figure 6.18 shows the travel distances of the four algorithms for all datasets.

Figure 6.19 compares TeSSI and TeSSIduo makespans for batches of size 1, 2 and 4. TeSSI and TeSSIduo have the same makespan for the same batch size because they schedule tasks to be executed at the same time but allocate them to different robots. For instance, Figure 6.20 shows that TeSSI allocated all tasks of dataset SDU-TGR-1\_batch\_size1 to robot 1, while TeSSIduo distributed them among robots 1, 2 and 4. However, both algorithms scheduled the tasks to be performed at the same time. Similarly, Figures 6.21 and 6.22 show that for batches of size 2 and 4, TeSSI and TeSSIduo scheduled tasks at the same time but to different robots.

Figure 6.23 shows that with MURDOCH, SSI and TeSSIduo, robots follow the same trajectories for the allocations of dataset SDU-TGR-1\_batch\_size1. Table 6.2 shows that for dataset SDU-TGR-1, the difference in travel distance between MURDOCH and SSI increases as the batch size increases. The difference in distance between TeSSI and TeSSIduo decreases between batch sizes 1 and 2 but not between batch sizes 2 and 4. The change in robot positions after each batch iteration affects the total travel distance, because the algorithms use a new initial robot position for allocating the tasks in the next batch.

### 6.3.6 Conclusions

MURDOCH, SSI, and TeSSIduo provide the same allocations when tasks arrive individually to the system. This is under the assumption that all tasks are executed before attempting to allocate a new batch. A variation of this experiment would be to retain some tasks in the schedule before a new batch arrives and observe how the schedule changes because of the arrival of new tasks.
6.4. Experiment 4: On-line Allocation of Tasks Clustered in the Map

The increase in batch size does not necessarily reduce the distance traveled by the robots. The change in robot positions after each batch iteration affects the allocations of the next batch. SSI and TeSSIduo optimize the distances between tasks within the same batch, but might place the robots farther away from the tasks of the next batch. Robots have no information about future tasks, and thus, can only optimize the paths of the current batch allocation. The travel distances of MURDOCH increase as the batch size increases because only one task within a batch can be allocated per robot. When tasks are introduced in smaller batches, the same robot can execute two tasks near to each other, reducing the over travel distance.

TeSSI and TeSSIduo schedule tasks at the same time but allocate them to different robots. TeSSIduo distributes the tasks among the robots so as to minimize the distance traveled by the fleet.

6.4.1 Purpose of the Experiment

Evaluate the quality of the allocations when tasks are clustered in the map, and they are introduced in batches of increasing sizes, i.e., not all tasks are known beforehand. The experiment runs with clusters of size 1, 2, 3 and 4m and batches of size 1, 2 and 4.

6.4.2 Experimental Design Considerations

This experiment shares the design considerations of experiment 3, except for the last point. Tasks are not uniformly distributed in the map but distributed in two clusters to assess how much TA approaches optimize the travel distance when tasks have positive synergies. The experiment runs for datasets with clusters of 1, 2, 3 and 4m. Pickup and delivery locations of a task belong to the same cluster so that a robot assigned to a cluster stays within the cluster and potentially receives more than one task inside that cluster.
6.4.3 Hypothesis

The four algorithms will allocate all tasks for all batch sizes because the size of a batch is at most equal to the number of robots in the fleet. MURDOCH, SSI, and TeSSIduo will provide the same allocations when tasks arrive in batches of size 1. SSI and TeSSIduo will provide allocations with smaller travel distances than MURDOCH as the size of the batches increases. TeSSI will not benefit from the clustered distribution of tasks because it does not use distance to compute its bids. TeSSIduo will allocate more than one task to a robot if the robot is the nearest one to the tasks and can make it to the pickup location before the latest start time. We expect TeSSIduo to have schedules with larger makespans than TeSSI.

6.4.4 Results

<table>
<thead>
<tr>
<th>Cluster radius [m]</th>
<th>Batch size</th>
<th>Difference between MURDOCH and SSI travel distance [m]</th>
<th>Difference between TeSSI and TeSSIduo travel distance [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3.9</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>15.89</td>
<td>18.24</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>46.25</td>
<td>37.79</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>11.89</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>12.4</td>
<td>29.81</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>66.62</td>
<td>55.27</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>10.46</td>
<td>1.22</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>34.73</td>
<td>26.24</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>8.76</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>9.78</td>
<td>29.3</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>61.62</td>
<td>10.94</td>
</tr>
</tbody>
</table>

Table 6.3: Experiment 4: Difference on travel distance as the cluster radius and batch size increases.
6.4. Experiment 4: On-line Allocation of Tasks Clustered in the Map

(a) Experiment 4: Number of allocations for batches of size 1.
(b) Experiment 4: Number of allocations for batches of size 2.
(c) Experiment 4: Number of allocations for batches of size 4.

Figure 6.26: Experiment 4: Number of allocations for task batches of sizes 1, 2 and 4.
Chapter 6. Results

(a) Experiment 4: Number of messages sent and received by the auctioneer for batches of size 1.

(b) Experiment 4: Number of messages sent and received by the auctioneer for batches of size 2.

(c) Experiment 4: Number of messages sent and received by the auctioneer for batches of size 4.

Figure 6.27: Experiment 4: Number of messages sent and received for task batches of sizes 1, 2 and 4.
6.4. Experiment 4: On-line Allocation of Tasks Clustered in the Map

(a) Experiment 4: Sum of the distances that the robots traveled to executed their tasks for batches of size 1.

(b) Experiment 4: Sum of the distances that the robots traveled to executed their tasks for batches of size 2.

(c) Experiment 4: Sum of the distances that the robots traveled to executed their tasks for batches of size 4.

Figure 6.28: Experiment 4: Sum of the distances that the robots traveled to executed their tasks for batches of sizes 1, 2 and 4.
Chapter 6. Results

(a) Experiment 4: Time the fleet needed for executing all tasks for batches of size 1.

(b) Experiment 4: Time the fleet needed for executing all tasks for batches of size 2.

(c) Experiment 4: Time the fleet needed for executing all tasks for batches of size 4.

Figure 6.29: Experiment 4: Time the fleet needed for executing all tasks for batches of sizes 1, 2 and 4.
6.4. Experiment 4: On-line Allocation of Tasks Clustered in the Map

(a) Experiment 4: TeSSI temporal distribution of tasks per robot for batches of size 1.

(b) Experiment 4: TeSSEiduo temporal distribution of tasks per robot for batches of size 1.

Figure 6.30: Experiment 4: Temporal distribution of tasks for dataset SDC-TGR-CR-1_batch_size1.

(a) Experiment 4: TeSSI temporal distribution of tasks per robot for batches of size 2.

(b) Experiment 4: TeSSEiduo temporal distribution of tasks per robot for batches of size 2.

Figure 6.31: Experiment 4: Temporal distribution of tasks for dataset SDC-TGR-CR-1_batch_size2.

(a) Experiment 4: TeSSI temporal distribution of tasks per robot for batches of size 4.

(b) Experiment 4: TeSSEiduo temporal distribution of tasks per robot for batches of size 4.

Figure 6.32: Experiment 4: Temporal distribution of tasks for dataset SDC-TGR-CR-1_batch_size4.
Chapter 6. Results

(a) Distance: 25.84m, Allocations: 8, Robot usage: 50%, Avg. allocation time per batch: 0.26s, Messages sent and received: 55

(b) Distance: 25.84m, Allocations: 8, Robot usage: 50%, Avg. allocation time per batch: 0.26s, Messages sent and received: 63

(c) Distance: 29.74m, Makespan: 7.43s, Allocations: 8, Robot usage: 25%, Avg. allocation time per batch: 0.26s, Messages sent and received: 63

(d) Distance: 25.84m, Makespan: 7.43s, Allocations: 8, Robot usage: 50%, Avg. allocation time per batch: 0.26s, Messages sent and received: 63

Figure 6.33: Robot trajectories for dataset SDC-TGR-CR-1_batch_size1. Each rectangle represents a robot and the dots represent pickup and delivery locations.
6.4. Experiment 4: On-line Allocation of Tasks Clustered in the Map

(a) Distance: 39.39m, Allocations: 8, Robot usage: 75%, Avg. allocation time per batch: 0.518s, Messages sent and received: 51

(b) Distance: 23.5m, Allocations: 8, Robot usage: 50%, Avg. allocation time per batch: 0.520s, Messages sent and received: 59

(c) Distance: 42.69m, Makespan: 4.93s, Allocations: 8, Robot usage: 50%, Avg. allocation time per batch: 0.521s, Messages sent and received: 59

(d) Distance: 24.45m, Makespan: 9.84s, Allocations: 8, Robot usage: 50%, Avg. allocation time per batch: 0.521s, Messages sent and received: 59

Figure 6.34: Robot trajectories for dataset SDC-TGR-CR-1_batch_size2. Each rectangle represents a robot and the dots represent pickup and delivery locations.
Chapter 6. Results

Figure 6.35: Robot trajectories for dataset SDC-TGR-CR-1_batch_size4. Each rectangle represents a robot and the dots represent pickup and delivery locations.

(a) Distance: 69.56m, Allocations: 8, Robot usage: 100%, Avg. allocation time per batch: 1.037s, Messages sent and received: 49

(b) Distance: 23.31m, Allocations: 8, Robot usage: 50%, Avg. allocation time per batch: 1.041s, Messages sent and received: 57

(c) Distance: 69.2, Makespan: 2.56s, Allocations: 8, Robot usage: 100%, Avg. allocation time per batch: 1.046s, Messages sent and received: 57

(d) Distance: 31.41m, Makespan: 8.87s, Allocations: 8, Robot usage: 75%, Avg. allocation time per batch: 1.045s, Messages sent and received: 57
6.4.5 Analysis of Results

Figure 6.26 shows that the four algorithms allocated all tasks for all datasets and all batch sizes. Comparing Figures 6.17 and 6.27 shows that experiment 3 and 4 use the same amount of messages because the number of allocated tasks per batch is the same.

Figure 6.28a shows that MURDOCH, SSI, and TeSSIduo provide the same allocations when there is only one task per batch. Figures 6.28b and 6.28c show that SSI allocations have a shorter travel distance than MURDOCH allocations as the batch size increases. Since tasks are clustered in the map, the difference in distance is more notorious in this experiment than in experiment 3.

Table 6.3 shows that the difference in distance between MURDOCH and SSI increases as the batch size increases for all cluster sizes. Figure 6.29 shows that the makespan of TeSSI and TeSSIduo is the same for batches of size 1, but TeSSIduo’s makespan increases as the size of the batches increase. This is because TeSSIduo allocates tasks from the same batch to a single robot, while TeSSI distributes a batch among more robots, as shown in Figures 6.30, 6.31, and 6.32. Note that task batches are separated by 30s. For instance, Figure 6.31 shows that with TeSSIduo, robot 4 received both tasks within the first batch and both tasks within the second batch. Similarly, robot 1 allocated the third and fourth batches. On the other hand, TeSSI splits each batch between robots 1 and 2. TeSSIduo can allocate the whole batch to a robot because the clustered task distribution and temporal constraints allow the same robot to be in time to execute both tasks. TeSSI does not take advantage of the spatial distribution of tasks and assigns a robot per task per batch, like Figure 6.32 shows.

Figures 6.33, 6.34 and 6.35 show the trajectories that robots follow for executing tasks distributed in clusters with radius of 1m when the batch sizes of incoming tasks increase. Trajectories for MURDOCH, SSI, and TeSSI are equal for batches of size 1. Since robots move to the delivery location of their last task before receiving a new task, MURDOCH can allocate a robot per cluster for batches of size 1. When batch sizes increase, MURDOCH cannot longer take advantage of the clustered distribution of tasks.
6.4.6 Conclusions

The results of this experiment confirm the results of experiment 3. When tasks arrive individually, and robots have empty schedules, MURDOCH, SSI, and TeSSIduo provide the same allocations. MURDOCH assigns the best-suited robot for the task at the time the task is announced and because robots move before receiving a new batch, robots eligibility increases if their last allocation was near a task that will be auctioned in the next iteration. TeSSIduo assigns tasks of the same batch to one robot as long as the robot is the nearest one to the tasks and it can arrive at the task pickup location in time.

6.5 Experiment 5: Off-Line Allocation of Tasks Uniformly Distributed in Time and Space

6.5.1 Purpose of the Experiment

Investigate the quality of allocations when tasks are uniformly distributed in time and space. This experiment focuses on the algorithms that take time constraints into account, namely TeSSI and TeSSIduo, but includes MURDOCH and SSI for completeness.

6.5.2 Experimental Design Considerations

- Since the focus of the experiment is not in IA approaches, the number of tasks is the double of the number of robots. This means MURDOCH will only be able to allocate half of the tasks.

- The experiment evaluates allocations when tasks are uniformly distributed in time. To reduce the effect that distance relationships between tasks have, we have distributed them uniformly in the map.

- Tasks have different earliest and latest start times but the same duration. i.e., all tasks have the same makespan.
6.5. Experiment 5: Off-Line Allocation of Tasks Uniformly Distributed in Time and Space

- Task duration is equivalent to the distance to go from the pickup to the delivery location of a task.

6.5.3 Hypothesis

TeSSI duo will assign more tasks per robot than TeSSI. This behavior is expected because we have configured TeSSI duo to give a weight of 0.9 to the distance and a weight of 0.1 to the makespan. If a task is close to a robot that can allocate the task at some time between the earliest and latest start time, TeSSI duo will assign the task to that robot even if there is another robot that could make it to the pickup location at the earliest start time but whose travel distance is larger.

TeSSI duo’s allocations will have shorter travel distances than TeSSI’s allocations but larger than SSI’s allocations. MURDOCH will only allocate half of the tasks. TeSSI duo’s schedules will be more compact than TeSSI’s schedules, but it is unclear whether the makespan of the fleet will be larger with TeSSI duo than with TeSSI. This will depend on the start time of the first task and the finish time of the last task in the schedule. If TeSSI and TeSSI duo schedule the last task to be performed at the same time, the total makespan of the fleet will be equal for both algorithms, provided both approaches scheduled the first task to start at the same time.

6.5.4 Results

(a) Experiment 5: Number of successful and unsuccessful allocations.

(b) Experiment 5: Number of messages sent and received by the auctioneer.

Figure 6.36: Experiment 5: Number of allocations and messages sent and received.

148
Chapter 6. Results

(a) Experiment 5: Distances that the robots will travel to execute their tasks.

(b) Experiment 5: Time the fleet will take to execute all tasks.

Figure 6.37: Experiment 5: Travel distances and makespan of the fleet.

(a) Experiment 5: TeSSI temporal distribution of tasks per robot.

(b) Experiment 5: TeSSIduo temporal distribution of tasks per robot.

Figure 6.38: Experiment 5: Temporal distribution of tasks for dataset TDU-TGR-1.
6.5. Experiment 5: Off-Line Allocation of Tasks Uniformly Distributed in Time and Space

(a) Distance: 28.74m, Allocations: 4, Robot usage: 100%, Time to allocate: 1.035s, Messages sent and received: 45

(b) Distance: 48.0m, Allocations: 8, Robot usage: 75%, Time to allocate: 2.099s, Messages sent and received: 57

(c) Distance: 99.05m, Makespan: 69.91s, Allocations: 8, Robot usage: 100%, Time to allocate: 2.109s, Messages sent and received: 57

(d) Distance: 60.37m, Makespan: 69.91s, Allocations: 8, Robot usage: 75%, Time to allocate: 2.105s, Messages sent and received: 57

Figure 6.39: Robot trajectories for dataset TDU-TGR-1. Each rectangle represents a robot and the dots represent pickup and delivery locations.
Chapter 6. Results

(a) Distance: 35.66m, Allocations: 4, Robot usage: 100%, Time to allocate: 1.038s, Messages sent and received: 45

(b) Distance: 59.58m, Allocations: 8, Robot usage: 75%, Time to allocate: 2.092s, Messages sent and received: 57

(c) Distance: 85.31m, Makespan: 74.0s, Allocations: 8, Robot usage: 100%, Time to allocate: 2.103s, Messages sent and received: 57

(d) Distance: 59.68m, Makespan: 74.0s, Allocations: 8, Robot usage: 100%, Time to allocate: 2.104s, Messages sent and received: 57

Figure 6.40: Robot trajectories for dataset TDU-TGR-2. Each rectangle represents a robot and the dots represent pickup and delivery locations.
6.5. Experiment 5: Off-Line Allocation of Tasks Uniformly Distributed in Time and Space

(a) Distance: 36.51m, Allocations: 4, Robot usage: 100%, Time to allocate: 1.038s, Messages sent and received: 45

(b) Distance: 50.47m, Allocations: 8, Robot usage: 100%, Time to allocate: 2.117s, Messages sent and received: 57

(c) Distance: 80.32m, Makespan: 43.0s, Allocations: 8, Robot usage: 100%, Time to allocate: 2.107s, Messages sent and received: 57

(d) Distance: 56.68m, Makespan: 43.0s, Allocations: 8, Robot usage: 100%, Time to allocate: 2.103s, Messages sent and received: 57

Figure 6.41: Robot trajectories for dataset TDU-TGR-3. Each rectangle represents a robot and the dots represent pickup and delivery locations.
Chapter 6. Results

Figure 6.42: Robot trajectories for dataset TDU-TGR-4. Each rectangle represents a robot and the dots represent pickup and delivery locations.
6.5. Experiment 5: Off-Line Allocation of Tasks Uniformly Distributed in Time and Space

(a) Distance: 47.91m, Allocations: 4, Robot usage: 100%, Time to allocate: 1.036s, Messages sent and received: 45

(b) Distance: 54.02m, Allocations: 8, Robot usage: 100%, Time to allocate: 2.124s, Messages sent and received: 57

(c) Distance: 86.61m, Makespan: 61.0s, Allocations: 8, Robot usage: 100%, Time to allocate: 2.105s, Messages sent and received: 57

(d) Distance: 63.25m, Makespan: 61.0s, Allocations: 8, Robot usage: 100%, Time to allocate: 2.109s, Messages sent and received: 57

Figure 6.43: Robot trajectories for dataset TDU-TGR-5. Each rectangle represents a robot and the dots represent pickup and delivery locations.

6.5.5 Analysis of Results

Figure 6.36a shows that SSI, TeSSI and TeSSI duo allocated all tasks while MURDOCH only allocated half of them. MURDOCH sent fewer messages than the other methods because there were fewer ALLOCATION messages. Figure 6.36b shows that the number of received messages by the MURDOCH auctioneer is also
less because robots do not send their updated schedules after an allocation. TeSSI allocated tasks to all robots in all datasets. On the contrary, TeSSIduo used only 3 of the 4 robots to allocate dataset TDU-TGR-1.

The travel distances for MURDOCH allocations in Figure 6.37a are not comparable to the travel distances of the other algorithms because MURDOCH only allocated half of the tasks. TeSSI provided allocations with larger travel distances than TeSSIduo but with the same makespan, as illustrated in Figure 6.37b. The total makespan is the time difference between the start of the first task and the end of the last task. In this experiment, the first and the last task in TeSSI and TeSSIduo schedules are the same, and both approaches allocated them to start at the same time. Figure 6.38 shows that for dataset TDU-TGR-1, TeSSI allocated the task with the earliest start time to robot 2, while TeSSIduo allocated it to robot 1. Similarly, TeSSI allocated the last task in the schedule to robot 1, while TeSSIduo allocated it to robot 4. The distribution of tasks among robots is different, but the makespan of the fleet is the same.

Figures 6.39, 6.40, 6.41, 6.42 and 6.43 show the trajectories that each robot will follow to execute their allocated tasks. It is easy to observe that in general robots will need to travel more with TeSSI allocations than with TeSSIduo allocations. However, both allocations comply with the temporal constraints. SSI provided allocations with smaller travel distances than TeSSIduo, but its allocations violate the temporal constraints.

6.5.6 Conclusions

TeSSIduo allocated more tasks per robot than TeSSI and in some cases (dataset TDU-TGR-1) used fewer robots to allocate all the tasks. That is, TeSSIduo built larger schedules for some robots and left schedules for other robots empty.

TeSSIduo provided allocations with shorter travel distances than TeSSI but with the same makespan. The makespan is determined by the first and last task in the schedule. If the allocations of both algorithms are the same for these particular two tasks, the makespan of the fleet will be the same for both approaches. Another perhaps more interesting performance metric would be the idle time of each robot and the total idle time of the fleet.
6.6 Experiment 6: Off-line Allocation of Tasks Clustered in Time and Space

6.6.1 Purpose of the Experiment

Investigate the quality of the allocations when tasks are clustered in space and time, and all tasks are known beforehand.

6.6.2 Experimental Design Considerations

- The number of tasks is the double of the number of robots so that robots accommodate more than one task in their schedule if the time constraints allow it.

- The experiment evaluates allocations when tasks are clustered in time, and there are no overlaps between time windows.

- Tasks within a cluster are separated by a fixed time interval which changes from dataset to dataset to investigate the effect that time separation between tasks has on the allocations.

- Tasks belonging to a temporal cluster also belong to a spatial cluster. Since tasks need to be executed within some seconds from one another, it makes sense to distribute temporal clustered tasks within a spatial cluster.

6.6.3 Hypothesis

Robots will accommodate in their schedule tasks of the same temporal cluster. TeSSIduo will allocate tasks to the closest robot which meets the time constraints, while TeSSI might allocate them to a more remote robot as long as it fulfills the temporal constraints. TeSSI will use in general a more significant percentage of the robots, while TeSSIduo will allocate tasks to fewer robots of the fleet.

Since all tasks have non-overlapping time windows and have a separation in time that allows robots to go from the delivery location of one task to the pickup
location of the another task, we expect all tasks to be allocated by TeSSI, TeSSIduo, and SSI. MURDOCH will only be able to allocate half of the tasks. SSI will most likely provide the allocations with the smallest travel distance but which violate the temporal constraints.

### 6.6.4 Results

(a) Experiment 6: Number of successful and unsuccessful allocations.

(b) Experiment 6: Number of messages sent and received by the auctioneer.

Figure 6.44: Experiment 6: Number of allocations and messages sent and received.

<table>
<thead>
<tr>
<th>Interval between time windows [s]</th>
<th>Difference between TeSSI and TeSSIduo travel distance [m]</th>
<th>Difference between TeSSI and TeSSIduo makespan [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18.42</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>14.83</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1.65</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>22.4</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.4: Experiment 6: Difference on travel distance and makespan as the interval between time windows increases.

157
6.6. Experiment 6: Off-line Allocation of Tasks Clustered in Time and Space

(a) Experiment 6: Distances that the robots will travel to execute their tasks.

(b) Experiment 6: Time the fleet will take to execute all tasks.

Figure 6.45: Experiment 6: Travel distances and makespan of the fleet.

(a) Experiment 6: TeSSI temporal distribution of tasks per robot.

(b) Experiment 6: TeSSIduo temporal distribution of tasks per robot.

Figure 6.46: Experiment 6: Temporal distribution of tasks for dataset TDC-TGR-ITW-1.
Chapter 6. Results

(a) Distance: 42.04m, Allocations: 4, Robot usage: 100%, Time to allocate: 2.018s, Messages sent and received: 45

(b) Distance: 55.81m, Allocations: 8, Robot usage: 50%, Time to allocate: 2.108s, Messages sent and received: 57

(c) Distance: 76.73m, Makespan: 117.0s, Allocations: 8, Robot usage: 100%, Time to allocate: 2.129s, Messages sent and received: 57

(d) Distance: 58.31m, Makespan: 117.0s, Allocations: 8, Robot usage: 75%, Time to allocate: 2.12s, Messages sent and received: 57

Figure 6.47: Robot trajectories for dataset TDC-TGR-ITW-1. Each rectangle represents a robot and the dots represent pickup and delivery locations.
6.6. Experiment 6: Off-line Allocation of Tasks Clustered in Time and Space

(a) Distance: 51.91m, Allocations: 4, Robot usage: 100%, Time to allocate: 2.022s, Messages sent and received: 45

(b) Distance: 54.75m, Allocations: 8, Robot usage: 50%, Time to allocate: 2.118s, Messages sent and received: 57

(c) Distance: 76.03m, Makespan: 214.0s, Allocations: 8, Robot usage: 100%, Time to allocate: 2.123s, Messages sent and received: 57

(d) Distance: 61.2m, Makespan: 214.0s, Allocations: 8, Robot usage: 75%, Time to allocate: 2.117s, Messages sent and received: 57

Figure 6.48: Robot trajectories for dataset TDC-TGR-ITW-2. Each rectangle represents a robot and the dots represent pickup and delivery locations.
Chapter 6. Results

(a) Distance: 31.86m, Allocations: 4, Robot usage: 100%, Time to allocate: 2.023s, Messages sent and received: 45

(b) Distance: 47.67m, Allocations: 8, Robot usage: 50%, Time to allocate: 2.119s, Messages sent and received: 57

(c) Distance: 59.54m, Makespan: 368.0s, Allocations: 8, Robot usage: 100%, Time to allocate: 2.105s, Messages sent and received: 57

(d) Distance: 57.89m, Makespan: 368.0s, Allocations: 8, Robot usage: 100%, Time to allocate: 2.129s, Messages sent and received: 57

Figure 6.49: Robot trajectories for dataset TDC-TGR-ITW-3. Each rectangle represents a robot and the dots represent pickup and delivery locations.
6.6. Experiment 6: Off-line Allocation of Tasks Clustered in Time and Space

Figure 6.50: Robot trajectories for dataset TDC-TGR-ITW-4. Each rectangle represents a robot and the dots represent pickup and delivery locations.

(a) Distance: 39.6m, Allocations: 4, Robot usage: 100%, Time to allocate: 2.022s, Messages sent and received: 45

(b) Distance: 42.99m, Allocations: 8, Robot usage: 50%, Time to allocate: 2.101s, Messages sent and received: 57

(c) Distance: 74.69m, Makespan: 240.0s, Allocations: 8, Robot usage: 100%, Time to allocate: 2.13s, Messages sent and received: 57

(d) Distance: 52.29m, Makespan: 240.0s, Allocations: 8, Robot usage: 75%, Time to allocate: 2.106s, Messages sent and received: 57

6.6.5 Analysis of Results

SSI, TeSSI, and TeSSIduo allocated all tasks for all datasets, while MURDOCH only allocated half of them, as shown in Figure 6.44a. Figure 6.44b shows that the number of messages sent and received by SSI, TeSSI, and TeSSIduo is the same. The travel distances of MURDOCH are shorter but not comparable to the other
Chapter 6. Results

algorithms because MURDOCH only allocated half of the tasks. As expected, SSI provided the allocations with the shortest travel distances for all datasets. TeSSI provides in general larger travel distances than TeSSIduo (Figure 6.45a) because TeSSI does not use a bidding rule that tries to optimize distances. TeSSIduo uses, in general, fewer robots than TeSSI, and SSI is the algorithm that uses the least amount of robots.

Figure 6.45b shows that TeSSI and TeSSIduo have allocations with the same makespan. This is because both algorithms schedule the first task and the last task of each dataset to be executed at the same time. Figure 6.46 illustrates the temporal distribution of tasks for dataset TDC-TGR-ITW-1 per robot. It shows that TeSSI and TeSSIduo scheduled tasks to be performed at the same time but by different robots.

Figures 6.47, 6.48, 6.49, 6.50 show the trajectories robots will follow to execute their tasks. SSI assigned in all cases a robot for each of the spatial clusters, but its allocations violate the temporal constraints. Since TeSSIduo optimizes distances, it allocated fewer robots per temporal/spatial cluster than TeSSI.

Table 6.4 shows that the difference in travel distance between TeSSI and TeSSIduo decreases as the interval between temporal clusters increases from 1 to 3 seconds. However, the difference in travel distance increases when the temporal clusters are separated by 4 seconds. These results are not conclusive since the distribution of spatial clusters and tasks within them change from dataset to dataset. A variation of this experimental design is to keep the spatial distribution of tasks constant and change the time windows of the tasks.

6.6.6 Conclusions

With TeSSI and TeSSIduo, robots accommodated tasks of the same temporal/spatial cluster in their schedules. TeSSIduo allocated fewer robots per temporal/spatial cluster, but scheduled tasks to be performed at the same time as TeSSI. Allocating more tasks to one robot instead of distributing them among more robots did not affect the start times of tasks and all tasks were allocated to start at their earliest start time. This is due to the distribution of tasks in the datasets used in these experiments. If robots have time to arrive at the pickup location of tasks at the
earliest start time, then using distance in the bidding rule, as TeSSIduo does, will only affect the travel distance but not the makespan. In the previous experiment, this was not the case. Tasks did not start at their earliest start time but some time between their earliest and latest start time because TeSSIduo made a compromise between distance and makespan.

SSI allocated one robot per temporal/spatial cluster, and its travel distances were the shortest for all datasets, but its allocations violate the temporal constraints. MURDOCH did not take advantage of the spatial relationship between tasks because robots become unavailable after allocating one task.

6.7 Experiment 7: Off-line Allocation of Increasing Number of Tasks Uniformly Distributed in Time and Space

6.7.1 Purpose of the Experiment

Evaluate the quality of the allocations when the number of robots in the fleet remains constant, but the number of tasks increases and tasks have overlapping time windows.

6.7.2 Experimental Design Considerations

- The number of tasks increases by 10 per dataset. Datasets are subsets of a dataset of 100 tasks, i.e., the 10 tasks of dataset TDU-ST-10 are the first 10 tasks of dataset TDU-ST-100.

- To prevent tasks from having positive synergies, we have distributed them uniformly in space and time.

- Tasks have overlapping time windows, i.e., some of them will not be allocated because of lack of available robots at the specific time slot.

- Task duration is equivalent to the distance to go from the pickup to the delivery location of a task, and it is the same for all tasks. This means that tasks have the same makespan.
6.7.3 Hypothesis

Since there are more tasks than robots for all datasets, MURDOCH will never allocate all tasks. If only one robot is available at a time slot when two or more tasks need to be performed, only one of those tasks will be allocated. The number of unsuccessful allocations will increase as the number of tasks increases. The task with the latest start time will dominate the makespan of the fleet. In other words, the schedules generated for the datasets that contain that task will have the same finish time, provided that the last task is scheduled to be performed at the same time in all datasets that contain that task. The allocation time will increase as the number of tasks increases since robots will need to allocate more tasks in their schedules.

6.7.4 Results

![Figure 6.51: Experiment 7: Number of successful and unsuccessful allocations.](image)

Figure 6.51: Experiment 7: Number of successful and unsuccessful allocations.
6.7. Experiment 7: Off-line Allocation of Increasing Number of Tasks Uniformly Distributed in Time and Space

Figure 6.52: Experiment 7: Number of messages sent and received by the auctioneer.

Figure 6.53: Experiment 7: Distances that the robots will travel to execute their tasks.

Figure 6.54: Experiment 7: Time the fleet will take to execute all tasks.
Figure 6.55: Experiment 7: TeSSI temporal distribution of tasks per robot for dataset TDU-ST-100.

Figure 6.56: Experiment 7: TeSSIduo temporal distribution of tasks per robot for dataset TDU-ST-100.

6.7.5 Analysis of Results

Figure 6.51 shows that MURDOCH always allocates the same amount of tasks, regardless of the number of tasks in the datasets, this is because it assigns only one task per robot. SSI allocates all tasks in all datasets because it does not validate the time constraints. For the datasets with 30, 40, 70, 80 and 90 tasks, TeSSI allocates more tasks than TeSSIduo. However, TeSSIduo allocates 47 of the 100 tasks in dataset TDU-ST-100 while TeSSI only allocates 46. Figure 6.55 shows the temporal distribution of dataset TDU-ST-100 using TeSSI. The allocation of the same dataset using TeSSIduo is shown in figure 6.56. Both allocations start almost at the same time, but the first task of robot 1 is different. Refer to the raw results in the CD attached to this report, to see the allocations for dataset TDU-ST-100. With TeSSI,
6.7. Experiment 7: Off-line Allocation of Increasing Number of Tasks Uniformly Distributed in Time and Space

(a) MURDOCH: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

(b) SSI: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

c) TeSSI: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

d) TeSSIduo: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

Figure 6.57: Experiment 7: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

robot 1 will perform task fc47c3e7-aaf1-450b-9d67-aa4ba38c896f at 11:50:33, while TeSSIduo will perform task 58a79cac-400d-4319-93c6-5570a75276c9 at 11:50:30. Task 58a79cac-400d-4319-93c6-5570a75276c9 could not be allocated by TeSSI and task fc47c3e7-aaf1-450b-9d67-aa4ba38c896f could not be allocated by TeSSIduo.

As expected, the number of sent and received messages increases as the number of tasks increases, as shown in Figure 6.52. Moreover if more tasks are allocated, the number of messages increases. For instance, for dataset TDU-ST-100, the number of messages used by TeSSIduo is greater than the number of messages sent by TeSSI because TeSSIduo allocated more tasks out of the 100 tasks.

Figure 6.53 shows the travel distances for each algorithm and dataset. The total travel distances of TeSSI allocations are in general larger than the total travel
Chapter 6. Results

distances of TeSSI duo because TeSSI duo optimizes distances. SSI has the largest travel distances because it allocates all tasks.

The makespan of the fleet is dominated by the last task in the time schedule. The makespan of TeSSI is larger than the one from TeSSI duo for dataset TDU-ST-60. TeSSI and TeSSI duo allocated the first task to start at the same time, but the last task of TeSSI finishes later than the last task of TeSSI duo. Even though both algorithms allocated the same amount of tasks (40 out of 60), the allocated tasks were different. Figure 6.54 shows that makespans for TeSSI and TeSSI duo are similar, but their values cannot be directly compared without knowing the number of allocated tasks per dataset per algorithm.

Figure 6.57 shows that for MURDOCH the total time for running the experiment grows linearly as the number of tasks increases. This is expected because the number of TASK-ANNOUNCEMENT messages that MURDOCH has to create grows with the number of tasks. The allocation time does not change from dataset to dataset because MURDOCH always allocates 4 tasks per dataset. For the other algorithms, the total time and allocation time exhibit a polynomial growth. While with MURDOCH the size of the TASK-ANNOUNCEMENT message is fixed, with the other algorithms the size of the TASK-ANNOUNCEMENT increases as the tasks to allocate increase and hence the time to build and process the messages grow. The number of tasks that each robot has to consider in each iteration and the number of tasks in each robot’s schedule also increase. TeSSI duo is the algorithm that takes the longest for allocating 100 tasks to 4 robots. From the 100 tasks, it allocated 47 in 2.19 minutes.

6.7.6 Conclusions

Using distance information in the bidding rule, as TeSSI duo does has an effect on which tasks are allocated. While TeSSI allocated some tasks of a dataset, TeSSI duo allocated different tasks from the same dataset. For most of the datasets, TeSSI allocated more tasks than TeSSI duo, but for the dataset with 100 tasks, TeSSI duo allocated 47 tasks while TeSSI allocated 46 tasks. The number of tasks that can be allocated depends on how many of them have overlapping time windows and on the availability of robots at the time tasks need to be executed. As expected, the number
of unsuccessful allocations increased as the number of tasks increased. Having more tasks with overlapping time windows and the same amount of robots makes it more unlikely to have a robot available at the time slot when a task needs to be executed.

The makespan of the fleet depends on the number of allocated tasks and the start time of the first task and the finish time of the latest task in the schedule. With SSI, TeSSI, and TeSSIduo, the allocation time and the time to run the experiment grow polynomially as the number of tasks increases. This is in accordance with the information in [21] and in [25], which states that the algorithms run in polynomial time.

SSI, TeSSI, and TeSSIduo scale well when the number of tasks increases. SSI allocated 100 tasks to 4 robots in 59.77s, with a total experiment time of 148.7s. Since some tasks were mutually exclusive, TeSSI and TeSSIduo could not allocate all tasks. The maximum number of tasks TeSSI allocated was 46, and it did it in 92.31s, while TeSSIduo allocated 47 tasks in 131.83s. TeSSI and TeSSIduo take more time to allocate tasks because they check for satisfiability of temporal constraints.

### 6.8 Experiment 8: Off-line Allocation of Increasing Number of Tasks Clustered in Time and Space

#### 6.8.1 Purpose of the Experiment

Evaluate the quality of the allocations when the number of robots in the fleet remains constant, but the number of tasks increases and tasks do not have overlapping time windows.

#### 6.8.2 Experimental Design Considerations

- The number of tasks increases by 10 per dataset. Datasets are subsets of a dataset of 100 tasks, i.e., the 10 tasks of dataset TDC-ST-10 are the first 10 tasks of dataset TDC-ST-100.

- To assure that all tasks can be allocated without violating the time constraints, tasks do not have overlapping time windows. In addition, the time interval between tasks allows robots to travel from the delivery location of a task to the pickup delivery of the next one without violating the time constraints.
Chapter 6. Results

- Tasks are distributed in two clusters in the map, and each cluster contains tasks with consecutive time windows.

- Task duration is equivalent to the distance to go from the pickup to the delivery location of a task, and it is the same for all tasks.

6.8.3 Hypothesis

Since tasks do not have overlapping time windows, TeSSI and TeSSIduo will allocate all tasks. SSI will also allocate all tasks, but its allocations will not comply with the temporal constraints. MURDOCH will only allocate 4 tasks because there are 4 robots in the fleet and the algorithm can only allocate one task per robot.

TeSSIduo will benefit from the spatial distribution of tasks and will provide allocations with a shorter travel distance than TeSSI. The makespan is determined by the first and the last task in the schedule, and thus, TeSSI and TeSSIduo will have the same makespan provided that they assign the same start time to the latest task and the same start time to the first start in the schedule. Allocation time will grow polynomially as the number of tasks to allocate increases. TeSSIduo will take longer than TeSSI to allocate tasks due to its bidding rule which requires more computations than TeSSI's bidding rule.

6.8.4 Results

![Figure 6.58: Experiment 8: Number of successful and unsuccessful allocations.](image)

Figure 6.58: Experiment 8: Number of successful and unsuccessful allocations.
6.8. Experiment 8: Off-line Allocation of Increasing Number of Tasks Clustered in Time and Space

Figure 6.59: Experiment 8: Number of messages sent and received by the auctioneer.

Figure 6.60: Experiment 8: Distances that the robots will travel to execute their tasks.

Figure 6.61: Experiment 8: Time the fleet will take to execute all tasks.
Chapter 6. Results

6.8.5 Analysis of Results

Figure 6.58 shows that SSI, TeSSI, and TeSSIduo allocated all tasks for all datasets. The number of messages sent and received by the auctioneer was the same for the three algorithms, as shown in 6.59. MURDOCH sent fewer messages than the other algorithms because it allocated fewer tasks. Figure 6.60 shows that MURDOCH provided the shortest travel distances, but it allocated only 4 tasks from each dataset. From the algorithms that allocated all tasks, SSI provides the shortest travel distances, but its allocations violate the temporal constraints. TeSSIduo provided shorter travel distances than TeSSI. For instance, with TeSSI the fleet travels 779.86m to execute 100 tasks, while with TeSSIduo the travel distance is 754.48m. Figure 6.61 shows that the makespan for TeSSI and TeSSIduo is the same.
6.8. Experiment 8: Off-line Allocation of Increasing Number of Tasks Clustered in Time and Space

(a) MURDOCH: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.
(b) SSI: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.
(c) TeSSI: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.
(d) TeSSIduo: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

Figure 6.64: Experiment 8: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

for all datasets because the first and last task in the schedule are the same. The makespan for datasets TDU-ST-40 to TDU-ST-100 is the same because the task with the latest start time is the same for these datasets.

Figure 6.62 shows that TeSSI distributed the second temporal cluster between robots 3 and 4, while TeSSIduo distributed the same cluster among robots 2 and 4, as shown in Figure 6.63. Tasks in a temporal cluster also belong to a spatial cluster. Robot 4 was closest to the second cluster, and thus, TeSSIduo assigned to it most of the tasks of that cluster. TeSSI sent a more distant robot to perform tasks of the second cluster, which explains why the total travel distance of TeSSI is larger.

Figure 6.64 shows how the allocation time and total time to run the experiments grow polynomially as the number of tasks increases. This is in accordance with the
Chapter 6. Results

information in [21] and in [25], which states that the algorithms run in polynomial time. As expected, TeSSIduo takes longer than TeSSI to allocate the same amount of tasks.

6.8.6 Conclusions

TeSSI and TeSSIduo allocate all tasks if tasks have non-overlapping time windows and the time between the finish time of one task and the start time of the next one allows robots to travel from one task to the other without violating the temporal constraints. TeSSIduo provides allocations with shorter travel distances than TeSSI. However, the makespan is the same if both algorithms assign the same start time to the last and the first task in the schedule. Allocation time grows polynomially as the number of tasks increases. TeSSIduo takes longer than TeSSI to allocate the same amount of tasks, and TeSSI takes longer than SSI. For instance, SSI allocated 100 tasks in 1.36 minutes, TeSSI did it in 14.30 minutes and TeSSIduo in 22.628 minutes.

6.9 Experiment 9: Off-line Allocation of Tasks Uniformly Distributed in Time with Increasing Number of Robots.

6.9.1 Purpose of the Experiment

Evaluate the quality of the allocations when the number of robots in the fleet increases but the amount of tasks to allocate remains constant, and tasks are uniformly distributed in time and space.

6.9.2 Experimental Design Considerations

- The experiment uses one dataset with 100 tasks uniformly distributed in time and space.

- Task duration is equivalent to the distance to go from the pickup to the delivery location of a task, and it is the same for all tasks.
6.9. Experiment 9: Off-line Allocation of Tasks Uniformly Distributed in Time with Increasing Number of Robots.

- Robot positions remain constant, i.e., the position of the first 10 robots is the same when the experiment runs with 20 robots than when it runs with 100 robots. New robots are added to the fleet, but the initial position of previous robots does not change.

- Some tasks in the dataset have overlapping time windows. This temporal distribution was chosen to evaluate the effect that the increasing amount of robots has on the number of allocations.

6.9.3 Hypothesis

MURDOCH will allocate more tasks as the number of robots in the fleet increases, i.e., with 10 robots it will allocate 10 tasks, and with 100 robots it will allocate 100 tasks. SSI will allocate all tasks regardless of the amount of robots in the fleet but the allocation time will increase with the number of robots. TeSSI and TeSSI duo will allocate more tasks as the size of the fleet increases, i.e., since more robots will be available, the number of tasks that could not be allocated due to their temporal constraints will be allocated to the new added robots.

6.9.4 Results

![Figure 6.65: Experiment 9: Number of successful and unsuccessful allocations.](image)

Figure 6.65: Experiment 9: Number of successful and unsuccessful allocations.
Chapter 6. Results

Figure 6.66: Experiment 9: Number of messages sent and received by the auctioneer.

Figure 6.67: Experiment 9: Distances that the robots will travel to execute their tasks.

Figure 6.68: Experiment 9: Time the fleet will take to execute all tasks.

177
6.9. Experiment 9: Off-line Allocation of Tasks Uniformly Distributed in Time with Increasing Number of Robots.

Figure 6.69: Experiment 9: TeSSI temporal distribution of tasks per robot for a fleet of 20 robots.

Figure 6.70: Experiment 9: TeSSIduo temporal distribution of tasks per robot for a fleet of 20 robots.

6.9.5 Analysis of Results

Figure 6.66 shows that the number of messages sent by the auctioneer remains the same but the number of messages the auctioneer receives increases as the size of the fleet increases. The auctioneer shouts one TASK-ANNOUNCEMENT message and one ALLOCATION message (containing the winner of the task) per allocation round. As the number of robots increase, the auctioneer receives bids from more robots, and thus the number of messages received increases.

Figure 6.65 shows that MURDOCH allocates the same amount of tasks as the robots in the fleet, i.e., if there are 40 robots, it allocates 40 tasks, one per robot. SSI
Chapter 6. Results

(a) MURDOCH: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

(b) SSI: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

(c) TeSSI: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

(d) TeSSIduo: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

Figure 6.71: Experiment 9: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

allocates all tasks, regardless of the number of robots. TeSSI and TeSSIduo allocate more tasks as the number of robots in the fleet increases. With 50 robots TeSSI allocated all tasks, while TeSSIduo left one unallocated task with the same amount of robots.

Figure 6.67 shows that the travel distance of TeSSIduo’s allocations for 60 robots is larger than for 70 robots. That is, when TeSSIduo had more robots available, it distributed the tasks so as to optimize the travel distance of the entire fleet. TeSSI provided allocations with larger travel distances as the number of robots in the fleet increased.

TeSSI and TeSSIduo makespans are similar, as shown in Figure 6.68. For instance, with 50 robots TeSSI has a makespan of 95.32s while TeSSIduo has a makespan of
96.75s. With a fleet of 100 robots, both algorithms have a makespan of 94s.

With 20 robots TeSSI allocated 77 tasks while TeSSIduo allocated 79 out of the 100 tasks. However, TeSSI provided a travel distance of 1.86 km and a makespan of 97.98s, while TeSSIduo yielded a travel distance of 1.25 km and a makespan of 96.75s. Figures 6.76 and 6.77 show the temporal distribution of tasks for a fleet of 20 robots using TeSSI and TeSSIduo.

Figure 6.71a shows that for MURDOCH, the allocation time increases as the size of the fleet increases, while the total time remains the same. For robots 10 to 100 the algorithm creates the same amount of TASK-ANNOUNCEMENT messages with one message per task; hence the time to run the experiment remains constant. However, when more robots are added to the fleet, the allocation time grows linearly with the number of robots. Figure 6.71b shows that SSI allocates all tasks regardless of the number of robots in the fleet, but the allocation time slightly increases as the size of the fleet grows. Figure 6.71c shows that the TeSSI allocation time steadily increases at a small rate as the number of robots increases. Allocation times for TeSSIduo also increase as the size of the fleet grows, as shown in Figure 6.71d.

6.9.6 Conclusions

The number of messages sent is independent of the number of robots in the fleet. However, the number of received messages increases as the size of the fleet grows. SSI allocates all tasks regardless of the number of robots in the fleet and provides the shortest travel distances for executing all tasks, but its allocations violate the time constraints.

If there are more tasks with overlapping time windows than robots available, only some of the tasks can be allocated. As the number of robots in the fleet increases more tasks can be allocated because the amount of available robots at those particular time slots is larger. However, the number of unsuccessful allocations is not the same for TeSSI and TeSSIduo. For the same tasks and number of robots, TeSSI sometimes allocates more tasks than TeSSIduo and TeSSIduo sometimes allocates more tasks than TeSSI. For instance, TeSSI allocated the 100 tasks to 50 robots, while TeSSIduo missed to allocate one task. However, TeSSIduo allocated 98 tasks to 40 robots and TeSSI only 96 to a fleet of 40 robots. The schedule that TeSSI and TeSSIduo build
for each robot is different and thus, adding one new task violates the constraints for one schedule but not for the other.

Allocation time grows at a small rate as the number of robots increases. The growth rate is larger for TeSSI and TeSSIduo than for SSI. MURDOCH allocates 100 tasks to 100 robots in 36.554s, SSI allocates the same tasks in 64.608s, TeSSI does it in 58.982s and TeSSIduo in 84.179s.

6.10 Experiment 10: Off-line Allocation of Tasks Clustered in Time and Space with Increasing Number of Robots

6.10.1 Purpose of the Experiment

Evaluate the quality of allocations when the number of robots in the fleet increases but the amount of tasks to allocate remains constant, and tasks do not have overlapping time windows.

6.10.2 Experimental Design Considerations

- The experiment uses one dataset with 100 tasks clustered in time and space.

- Tasks belonging to a temporal cluster also belong to an spatial cluster. Since tasks need to be executed within some seconds from one another, it makes sense to distribute temporal clustered tasks in spatial clusters.

- Time windows of tasks do not overlap so that all tasks can be allocated.

- Task duration is equivalent to the distance to go from the pickup to the delivery location of a task, and it is the same for all tasks.

- Robot positions remain constant, i.e., the position of the first 10 robots is the same when the experiment runs with 20 robots than when it runs with 100 robots. New robots are added to the fleet, but the initial position of previous robots does not change.
6.10.3 Hypothesis

SSI, TeSSI, and TeSSIduo will allocate all tasks because none of the tasks are mutually exclusive, i.e., their time windows do not overlap. MURDOCH will allocate as many tasks as there are robots in the fleet. As the number of robots increase, the allocation times for TeSSI and TeSSIduo will decrease because each robot will allocate less tasks in its schedule and thus calculating the cost for allocating a new task will be faster. In other words, there will be less insertion points in the robot’s schedule to accommodate the new task and placing a bid will require less computations.

6.10.4 Results

Figure 6.72: Experiment 10: Number of successful and unsuccessful allocations.
Figure 6.73: Experiment 10: Number of messages sent and received by the auctioneer.

Figure 6.74: Experiment 10: Distances that the robots will travel to execute their tasks.

Figure 6.75: Experiment 10: Time the fleet will take to execute all tasks.
6.10. Experiment 10: Off-line Allocation of Tasks Clustered in Time and Space with Increasing Number of Robots

6.10.5 Analysis of Results

Figure 6.72 shows that SSI, TeSSI, and TeSSIduo allocated all tasks for all number of robots, while MURDOCH allocated as many tasks as there were robots in the fleet. Figure 6.73 shows that, as with experiment 9, the number of sent messages remains constant, but the number of received messages increases as the size of the fleet grows.

Figure 6.74 shows that TeSSIduo allocations have almost the same total travel distance among different fleet sizes. With a fleet of 10 robots, TeSSIduo distributed the 100 tasks among 7 robots, leaving 3 of the robots idle and yielding a travel
Chapter 6. Results

(a) MURDOCH: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

(b) SSI: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

(c) TeSSI: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

(d) TeSSIduo: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

Figure 6.78: Experiment 10: Tasks to allocate vs Allocated tasks and Time to allocate vs Total time.

distance of 839.07m. With a fleet of 100 robots, TeSSIduo only allocated tasks to 14 robots giving a total travel distance of 807.33m. On the contrary, TeSSI assigned tasks to all the robots for all the fleet sizes. The distances that the robots will travel are larger, but all robots are used, and the makespan is the same than for TeSSIduo.

The makespan of all TeSSI and TeSSIduo allocations, as illustrated in Figure 6.75, is the same because for all number of robots in the fleet the allocated tasks were the same.

Figure 6.76 shows that TeSSI distributed the 100 tasks among the fleet of 20 robots so that each robot received at least one task. The TeSSIduo temporal distribution of tasks in Figure 6.77 reveals that TeSSIduo only used 10 robots out of 20 to allocate the same 100 tasks.
6.10. Experiment 10: Off-line Allocation of Tasks Clustered in Time and Space with Increasing Number of Robots

Figure 6.78a shows that the allocation time for MURDOCH grows linearly with the number of allocated tasks. The comparison between Figure 6.78b and Figure 6.71b shows that SSI needs more time for allocating spatial/temporal clustered tasks than for allocating uniformly distributed tasks. Allocation times for SSI increase with the number of allocated tasks. On the contrary, allocation times for TeSSI and TeSSIduo decrease as the number of robots increases.

SSI has larger allocation times than TeSSI because some robots allocate more tasks in their schedules, while TeSSI distributes the tasks among more robots in the fleet. For instance, using a fleet of 100 robots, SSI allocated half of the tasks to robot 25 and the other half to robot 75 in 187.04s. TeSSI allocated one task per robot in 66.19s, while TeSSIduo used 14 robots out of the 100 to allocate all tasks in 189.17s. TeSSIduo takes longer than SSI because it checks for satisfiability of temporal constraints.

Since all tasks have the same duration, the makespan TeSSI robots bid in the first iteration is the same for all tasks. When there is a tie, TeSSI allocates the task to the robot with the smallest ID, i.e., robot 1. After robot 1 has allocated a task, its makespan is larger than the makespan of the rest of the robots, and hence, TeSSI allocates the next task to robot 2. This explains why TeSSI uses all robots to allocate the 100 tasks. TeSSIduo modifies the bids by adding distance information; thus not all robots bid the same value and tasks are distributed among fewer robots.

6.10.6 Conclusions

TeSSIduo uses fewer robots than TeSSI for allocating the same set of tasks and produces allocations with a shorter travel distance. However, TeSSIduo needs more time than TeSSI to allocate the same amount of tasks. For instance, TeSSI allocated 100 tasks in 66.19s, while TeSSIduo took 189.17s to allocate the same 100 tasks. TeSSIduo takes longer than TeSSI because tasks are distributed between fewer robots in the fleet. TeSSI used the 100 robots to allocate the 100 tasks, while TeSSIduo only used 14 out of the 100 robots. The time needed to allocate a task depends on the size of the schedules of the robots. A robot with a larger schedule will take longer to compute its bids. Allocation times decrease when more robots join the fleet, provided that each robot allocates less tasks in its schedule than in the previous
Chapter 6. Results

fleet configuration. The number of tasks that each robot allocates depends on the temporal and spatial distribution of tasks and robots.

The distance that robots will travel is shorter with TeSSI duo than with TeSSI, but a bigger percentage of the robots remain idle.

6.11 Additional Findings

During the implementation of TeSSI and TeSSI duo, we tested two methods for storing and updating the information in the Simple Temporal Network (STN).

The STN is a matrix that can be stored using different data structures. Our first implementation stores the STN as a list of lists and it is based on the public repository [8], which implements the Floyd-Warshall algorithm. Every time the robot computes its bid for a task, it builds a STN that contains its already allocated tasks plus the new task. The STN checks for consistency in the schedule, i.e., if the new task can be added to the schedule without violating the temporal constraints, the robot can place a bid for it.

The second method for storing the STN uses a numpy array. The STN is not built every time the robot computes its bid for a new task. Instead, the numpy array is updated by adding the points and temporal constraints of the new task to the STN. In [25] the authors implemented the STN in a similar way. They store a copy of the STN and update its information every time the robot computes its bid for a task. However, the authors do not mention the data structure that they used for storing the STN.

In our experiments, we found out that storing the STN as a numpy array and updating it is more time consuming than creating a list of lists. The allocation of both methods and the performance metrics are the same, except for the running times (allocation time per task, average time, allocation time, and total time). Section 4.3 describes the performance metrics recorded for each experiment. Figure 6.79 shows the difference in time when a list of lists and a numpy array are used for experiment 9 with 10 robots. The list of lists implementation allocated all tasks in 23.8s, while the numpy array provided the same allocations in 41.4s. Moreover, the list of list implementation ran the experiment in 57.01s, while the numpy array implementation did it in 77.08s. The results reported in this chapter were obtained using the list of lists STN implementation.
6.11. Additional Findings

The STN is used for computing the makespan. In [25] the authors define makespan as “the time the last robot finishes its final task”. That is each robot bids the final time in its schedule. In [26], makespan is defined as the “time difference between the end of the last task and the start of the first task”. For our experiments, we tested both definitions and concluded that the second definition distributes tasks among more robots in the fleet and hence, its runtime is smaller. The results presented in this chapter used the second definition of makespan. With TeSSI, robots bid the makespan, while with TeSSIduo, robots bid a combination of makespan and distance to the task.

- **Makespan 1 (used in our experiments):** Robots bid the difference between the start time of the first task in their schedule and the finish time of the last task in their schedule.

- **Makespan 2 (as defined in TeSSI’s paper [25]):** Robots bid the finish time of the last task in their schedule.

Figure 6.80 shows the difference in allocation time and total time between the two definitions of makespan for experiment 8 using 100 tasks. TeSSI with makespan 1 allocated 100 tasks among four robots in 858.2s (14.3 minutes), and the experiment ran in 931.9s (15.5 minutes). TeSSI with makespan 2 allocated the same 100 tasks in 75224.8s (20.89 hours) and the total time was 75305.4s (20.92 hours).

Figure 6.81 shows the temporal distribution of tasks among the robots in the fleet. With both makespan definitions, tasks are executed at the same time, and the
makespan of the fleet is the same. However, tasks are distributed to different robots depending on the makespan definition. Figure 6.82 shows that tasks are distributed among all the robots in the fleet when the first definition is used. When TeSSI uses the second definition of makespan, all tasks are allocated to robot 1. The allocation time is much larger with makespan 2 than with makespan 1 because all tasks are allocated to a single robot. The time for allocating a task to a robot increases as the schedule of the robot grows. When robot 1 has only 2 tasks, there are only 3 insertions points where the new task could be added. As the schedule expands the number of insertions and thus the computations increase. When makespan 1 is used, all robots bid the same value (the finish time of the task), and the tie-breaking rule allocates the task to the robot with the smaller ID, i.e., robot 1. In [25], the authors indicate that the worst case complexity of the TeSSI and TeSSIduo is when all tasks are assigned to one robot, and its is $O(m^2)$.

Figure 6.80: Comparison of allocation time and total time using the two definitions of makespan.

The recorded information for all the experiments is in the CD attached to this report. The results for the Docker implementation are in docker_implementation/results and the results of the Task Allocator implementation are in task_allocation/test/results/. The results using the STN as a list of lists are in folders tessi_1 and tessiduo_1. Results with the STN as a numpy array are in folders tessi_2 and tessiduo_2.

Appendix C shows the performance metrics of the first three datasets of each experiment using the Docker implementation.
6.11. Additional Findings

(a) Temporal distribution of tasks per robot. Makespan = finish_time - start_time

(b) Temporal distribution of tasks per robot. Makespan = finish_time

Figure 6.81: Temporal distribution of tasks per robot using the two definitions of makespan.

(a) Robot usage. Makespan = finish_time - start_time

(b) Robot usage. Makespan = finish_time

Figure 6.82: Robot usage using the two definitions of makespan.
Conclusions

Our literature search and qualitative comparison revealed that the MRTA algorithms MURDOCH, SSI, TeSSI and TeSSIduo are suitable options for allocating transportation tasks in the context of ROPOD. We selected these methods based on the quality of their solutions, their communication and computation requirements, their ability to allocate tasks on-line and their capabilities to scale to large multi-robot systems.

Although the use case of transportation of supply carts does not require heterogeneous robots and all tasks have the same requirements, we consider algorithms that can be extended to more complex scenarios, involving heterogeneous tasks and robots and different task priorities. From the algorithms considered in the qualitative comparison, only CBPAE accounts for different task priorities, and it is designed to allocate heterogeneous tasks to a fleet of heterogeneous robots. However, MURDOCH, SSI, TeSSI, and TeSSIduo can be modified to include these features. Since CBPAE requires task execution information and our experimental setup does not perform task execution, we did not select CBPAE for the experimental comparison.

The allocation schemes of the ROPOD project require that tasks can be scheduled to be performed within a time window sometime in the future. MRTA algorithms with temporal constraints like TeSSI and TeSSIduo provide this functionality. The ROPOD project also includes an allocation scheme where tasks need to be assigned so that they can be executed as soon as possible. MURDOCH covers this case since it assigns tasks to be performed as soon as possible. Since TeSSI and TeSSIduo are
based on SSI, we decided to include SSI in the analysis for completeness.

We implemented MURDOCH, SSI, TeSSI, and TeSSIduo as Python modules and used Zyre, an open source framework, for the communication between the auctioneer and the robots. The auctioneer and the robots were implemented as Zyre nodes in Docker containers. This design should facilitate deployment of the system on the physical robots. During the implementation and design of the experiments, it became clear that execution information is vital for testing the algorithms in on-line scenarios. Even though we proposed on-line experiments, we only performed semi on-line allocations. In other words, we did not simulate task execution but instantaneously moved the robots to the delivery location of the last task in their schedule before introducing the next batch of unallocated tasks.

The results of the experiments provided insights into the performance of the algorithms under different scenarios and task distributions. In the off-line experiments, MURDOCH was in disadvantage because it could only allocate one task per robot. In the on-line scenarios, robots became available before allocating a new batch of tasks and MURDOCH allocated as much tasks as the other algorithms provided that the size of the batches was smaller or equal to the number of robots in the fleet. For a more extensive comparison of IA (instantaneous assignment) vs. TA (time-extended assignment), it is crucial to include execution information.

MURDOCH and TeSSIduo optimize distances while TeSSI optimizes the required time to execute a task (makespan). TeSSIduo tends to overload some of the robots while keeping other idle; this behavior is more noticeable when tasks are clustered in space and time. TeSSIduo prefers to allocate a task to the closest robot, which is why it allocates the whole cluster to a robot as long as the allocations do not violate the temporal constraints. TeSSI distributes the allocations among more robots, but the distance that the whole fleet has to travel is larger than with TeSSIduo. Experiment 10 tested robot scalability when tasks have nonoverlapping time windows and are distributed in spatial/temporal clusters. The results of this experiment reveal that TeSSIduo does not take advantage of the increasing number of available robots. It uses only 14 robots out of 100 to allocate 100 tasks. Consequently, the allocation time is larger than TeSSI’s allocation time for the same set of tasks. The allocation time increases with the number of tasks scheduled to a robot because there are more insertions points where a new task could be allocated.
TeSSIduo weights makespan and travel distance using a constant between 0 and 1. For all our experiments, the weighting factor of the distance was 0.9, while the weighting factor for the makespan was 0.1. By giving less importance to the distance, TeSSIduo can be tuned to prevent the behavior observed in experiment 10.

In some cases, the tie-breaking rule yielded suboptimal allocations. For instance, if two robots using TeSSI bid the same value for one task, the task was allocated to the robot with the lowest ID. This decision affected the allocation in the subsequent rounds. It would make more sense to use another objective function to break the ties, for instance, distance to the pickup location or the battery level.

The experimental evaluation of the algorithms reveals that TeSSI and its variation TeSSIduo, are the most suitable algorithms for the ROPOD use case “transportation of supply carts”. However, some modifications to these algorithms are desirable. In the section 7.3 we describe some of the modifications that could be made.

7.1 Contributions

We performed a qualitative analysis of MRTA algorithms and selected the most suitable ones for the ROPOD use case “transportation of supply carts”. Our work gives insights into the performance of four MRTA algorithms under different experimental setups, with several task distributions and an increasing number of tasks and robots. Scalability of tasks and robots play an essential role in the selection of the algorithms. Through our experimental results, we analyze the benefits and disadvantages of the selected algorithms.

There is a lack of available open source implementations of MRTA algorithms [26], which is why we had to implement the four selected algorithms and generate all the datasets needed for our experiments. We hope that the modules we have designed serve as a basis for implementing other MRTA algorithms under several experimental setups. Our implementation is still not public, but we would like to make it available to other people. In addition, we have written a Python script to generate task datasets. With this tool, one can select the map dimensions, the number of robots and the task distribution scheme. Tasks can be uniformly distributed or clustered in space and/or time. The size of the clusters and interval between task windows is also configurable.

In summary, this work contributes a (1) qualitative analysis of MRTA algorithms,
(2) implementation of MRTA algorithms in a common experimental setup, (3) generation of datasets with configurable parameters, (4) experimental comparison of four MRTA algorithms.

7.2 Lessons Learned

It is essential to define from the beginning of the project the requirements for implementing, testing and conducting an experimental comparison of algorithms; and search for open source implementations and tools to ease the experimental setup. For our particular case, we did not find available implementations of the algorithms we wanted to test. Fortunately, the Fleet Management System of the ROPOD project was already in an advanced development stage which allowed us to use some of its components for the implementation and integration of the multi-task allocation component.

Likewise, the kind of comparative analysis also poses some requirements on the implementation. At the beginning of the project we wanted to experimentally compare IA (instantaneous assignment) against TA (time-extended assignment) approaches. During the implementation and testing of the algorithms, we determined that we needed on-line experiments with execution status information for conducting a fair comparison between both approaches. Nonetheless, we designed the experiments in such a way that we could still get some valuable results. Because of this, our experiments include more off-line than on-line scenarios.

We also learned that the design of the system should be adaptable to more complex scenarios. We found out in the early stages of the project that our initial design had problems scaling to more than eight robots. Since we detected this limitation soon enough, we were able to modify the design and increase its scalability capabilities.

7.3 Future Work

Through the realization of this project, we identified the desirable characteristics of an MRTA algorithm operating in a dynamic environment, like a hospital. A multi-robot system handling logistics for transportation tasks should be able to allocate tasks to be performed at a particular time window in the future, have the
ability to allocate tasks on-line with different priorities and possess fault tolerant
capabilities.

Our experiment evaluation revealed that TeSSI and TeSSIduo are good options
for allocating tasks with temporal constraints. However, they do not consider task
priorities, lack fault tolerance capabilities and do not include execution status informa-
tion. Moreover, they only handle single-task-single-robot assignments. TeSSI and
TeSSIduo suit well the requirements of the “transportation of supply carts” ROPOD
use case, however, some modifications are desirable to increase their flexibility to
handle more complex scenarios.

TeSSIduo modifies TeSSI by introducing distance information to the bidding
rule. We propose to further extend TeSSI by including current battery level and
execution status information. The CBPAE algorithm, which was not included in
the experimental comparison of this work, has an interesting scheme for handling
priority based allocations. CBPAE was designed to allocate heterogeneous tasks to a
heterogeneous group of robots deployed in health care facilities. Priority allocation
and on-line allocation of tasks are some of the takeaway features of this algorithm.
However, CBPAE allocates tasks to be executed as soon as possible and hence, does
not build schedules of tasks to be performed at some time in the future. We propose
to implement and adapt CBPAE on-line scheme allocation and priority task handling
to TeSSI.

Our current implementation waits until the auctioneer has received a message
from all robots before electing a robot for performing the task. To increase the
robustness of the system, it is desirable to have a fixed auction time [13]. However,
there are no guidelines on selecting the auction time. One approach would be to
have different waiting times based on the priority of the unallocated tasks.

As a result of our experiments, we found that the tie-breaking rule has an impact
on the quality of the allocations. If two robots bid the same for the same task, the
tie-breaking rule that we implemented simply selects the robot with the smaller
ID. We would like to explore other tie-breaking rules that use performance metric
information or execution status to break ties.

Another interesting direction for the project would be to explore algorithms that
build a schedule of tasks like TeSSI does, but that assign a task to a group of robots
instead of to a single robot. This feature is desirable in scenarios where the load is
too large or heavy to be transported by a single robot. The algorithms that handle this kind of scenario are called single-task multi-robot algorithms (ST-MR) [15]. Along these lines, we would like to evaluate TeSSI’s suitability to form coalitions of robots to allocate tasks.

Furthermore, the use of a simulation tool to test more complex scenarios would be of great benefit. This way we could test the algorithms on-line; add new tasks and robots at run-time; retrieve robots from the fleet and simulate execution failure to test re-allocation mechanisms. Modifications to the dataset generator to increase the flexibility of experimental setups are also considered as future work.

In addition, tight integration of multi-robot task allocation and path planning is desirable. This way the allocator could have information about conflicting paths and use estimations of how the environment will be at the time the task needs to be executed.
A

Installation and Setup

A.1 Installation

A.1.1 Task Allocator Implementation

- To use the PyreBaseCommunicator, clone the ropod_common repository. Follow the installation instructions on the README. [35]
  
  git clone git@git.ropod.org:ropod/ropod_common.git

- To use the structures defined in the Fleet-Management, clone the fleet-management repository. Follow the instructions on the README. [33]. Use the task_allocation branch.
  
  git clone git@git.ropod.org:ropod/ccu/fleet-management.git

- Clone the experimental_task_allocation repository:
  
  git clone git@git.ropod.org:brsu/experimental_task_allocation.git

- Get the requirements:
  
  pip3 install -r requirements.txt

- Add the experimental_task_allocation to the PYTHONPATH:
  
  sudo pip3 install -e .
A.1.2 Docker Implementation

- Clone the experimental_task_allocation repository:
  ```
  git clone git@git.ropod.org:brsu/experimental_task_allocation.git
  ```
- Install docker: [https://docs.docker.com/install/linux/docker-ce/ubuntu/](https://docs.docker.com/install/linux/docker-ce/ubuntu/)
- Follow the post-installation steps for Linux: [https://docs.docker.com/install/linux/linux-postinstall/](https://docs.docker.com/install/linux/linux-postinstall/)
- Install docker-compose: [https://docs.docker.com/compose/install/](https://docs.docker.com/compose/install/)
- Get the ROPOD Docker certificate by following the instructions on the ROPOD: readme repository. [34]
- Login to the Docker register of ROPOD.
  ```
  docker login git.ropod.org:4567
  ```

A.2 Creating Configuration Diles and Datasets

The configuration files and datasets used for the experiments are already included in the experimental_task_allocation repository. If a new experimental setup is needed, new configuration files and datasets can be created, but the new files will overwrite the ones used for the experiments presented in this report.

Assuming that you are at the root of the experimental_task_allocation repository, go to the scripts folder and run the setup file. This script takes as arguments the dimensions of the map, the number of robots and the type of setup. Configuration files and datasets are generated based on this information. There are two types of setup, namely normal and scalability. The normal setup is used for creating the files for experiments 1 to 8, and the scalability setup creates the files needed for experiments 9 and 10. Both types of setups can take any map dimensions and number of robots. The difference is the type of datasets they generate and the folder where they store the files.

```python
python3 setup.py [type_of_setup][width][height][n_robots]
```
A.2.1 Experiments 1 to 9

Running: python3 setup.py normal 20. 20. 4
creates an experimental setup for a $20m \times 20m$ obstacle free map and 4 robots. The script generates the files:

- config/area_names.yaml
- config/config.yaml
- config/map.yaml
- config/ropod_positions.yaml
- Datasets:
  - 5 of type SDU-TER.
  - 5 of type SDU-TGR.
  - 4 of type SDC-TER.
  - 4 of type SDC-TGR.
  - 5 of type TDU-TGR.
  - 4 of type TDC-TGR.
  - 10 of type TDU-ST.
  - 10 of type TDC-ST.

A.2.2 Experiments 9 and 10

Running: python3 setup.py scalability 100. 100. 100
creates an experimental setup for a $100m \times 100m$ obstacle free map and 100 robots. The script generates the files:

- config/scalability/area_names.yaml
- config/scalability/config.yaml
- config/scalability/map.yaml
- config/scalability/ropod_positions.yaml
- Datasets:
  - 1 of type TDU-SR.
  - 1 of type TDC-SR.
A.3. Instructions for Running the Experiments

The datasets are stored in the datasets folder. Refer to section 4.5.4 to see a description of the datasets. Refer to section 5.3 for a description of the contents of each file. Use the same configuration files and datasets to run experiments 1 to 8 and the same configuration files and datasets for experiments 9 and 10. Optionally, plot the datasets and the robots initial positions using the jupyter notebook in datasets/Plots/plot_datasets.ipynb.

The number and type of datasets that the scrips generate are fixed. Modifications to increase the flexibility of experimental setups is considered as future work.

A.3 Instructions for Running the Experiments

A.3.1 Task Allocation Implementation

Go to the folder task_allocation/test. The experiment_initiator.py requires two arguments and has an optional argument.

```
python3 experiment_initiator.py [mrta_method][experiment_name] --verbose
```

Arguments required:

- MRTA method:
  - murdoch
  - ssi
  - tessi
  - tessiduo

- Experiment name:
  1. offline-spatial-uniform
  2. offline-spatial-clustered
  3. online-spatial-uniform
  4. online-spatial-clustered
  5. offline-temporal-uniform
  6. offline-temporal-clustered
  7. offline-scale-tasks-uniform
  8. offline-scale-tasks-clustered
Appendix A. Installation and Setup

9. offline-scale-robots-uniform
10. offline-scale-robots-clustered

Example: Running the offline-spatial-uniform experiment with MURDOCH:

```python
python3 experiment_initiator.py murdoch offline-spatial-uniform --verbose
```

A.3.2 Docker Implementation

Each experiment has a docker-compose-file in the docker-compose_files folder. Experiment numbers are described in table 4.1.

- Name scheme of the docker-compose-files:
  
  `[mrta_method]-exp-[experiment_number].yml`

- For the robot scalability experiments, the names include the number of robots:
  
  `[mrta_method]-exp-[experiment_number]-[number_robots].yml`

For example, the docker-compose file for running experiment 6 with TeSSIIduo is `tessiduo-exp6.yml`, and the docker-compose file for running experiment 10 with 100 robots and SSI is `ssi-exp10-100.yml`.

From the root folder of the experimental_task_allocation repository:

- Build the images:
  
  `docker-compose -f docker_compose_files/[docke-compose file] build task_allocation_test`

- Run the robots and the auctioneer:
  
  `docker-compose -f docker_compose_files/[docke-compose file] up task_allocation`

- In another terminal, run the experiment initiator:
  
  `docker-compose -f docker_compose_files/[docke-compose file] up task_allocation_test`
Example: Running the offline-scale-robots-uniform experiment with TeSSI and 10 robots:

```
docker-compose -f docker-compose_files/tessi-exp9-10.yml build task_allocation_test
docker-compose -f docker-compose_files/tessi-exp9-10.yml up task_allocation
docker-compose -f docker-compose_files/tessi-exp9-10.yml up task_allocation_test
```
B.1 Spatial Uniformly Distributed Tasks (SDU)

B.1.1 SDU-TER

Spatial uniformly distributed tasks with number of tasks equal to the number of robots.

(a) Spatial distribution of dataset SDU-TER-1. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset SDU-TER-1. Each square represents the time window of a task.

Figure B.1: Dataset SDU-TER-1.
B.1. Spatial Uniformly Distributed Tasks (SDU)

(a) Spatial distribution of dataset SDU-TER-2. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset SDU-TER-2. Each square represents the time window of a task.

Figure B.2: Dataset SDU-TER-2.

(a) Spatial distribution of dataset SDU-TER-3. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset SDU-TER-3. Each square represents the time window of a task.

Figure B.3: Dataset SDU-TER-3.
Appendix B. Datasets

(a) Spatial distribution of dataset SDU-TER-4. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset SDU-TER-4. Each square represents the time window of a task.

Figure B.4: Dataset SDU-TER-4.

(a) Spatial distribution of dataset SDU-TER-5. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset SDU-TER-5. Each square represents the time window of a task.

Figure B.5: Dataset SDU-TER-5.
B.1. Spatial Uniformly Distributed Tasks (SDU)

B.1.2 SDU-TGR

Spatial uniformly distributed tasks with twice as many tasks as robots.

(a) Spatial distribution of dataset SDU-TGR-1. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset SDU-TGR-1 batch size 1. Each square represents the time window of a task.

(c) Temporal distribution of dataset SDU-TGR-1 batch size 2. Each square represents the time window of a task.

(d) Temporal distribution of dataset SDU-TGR-1 batch size 4. Each square represents the time window of a task.

Figure B.6: Dataset SDU-TGR-1.
Appendix B. Datasets

(a) Spatial distribution of dataset SDU-TGR-2. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset SDU-TGR-2, batch size 1. Each square represents the time window of a task.

(c) Temporal distribution of dataset SDU-TGR-2, batch size 2. Each square represents the time window of a task.

(d) Temporal distribution of dataset SDU-TGR-2, batch size 4. Each square represents the time window of a task.

Figure B.7: Dataset SDU-TGR-2.
B.1. Spatial Uniformly Distributed Tasks (SDU)

(a) Spatial distribution of dataset SDU-TGR-3. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset SDU-TGR-3, batch size 1. Each square represents the time window of a task.

(c) Temporal distribution of dataset SDU-TGR-3, batch size 2. Each square represents the time window of a task.

(d) Temporal distribution of dataset SDU-TGR-3, batch size 4. Each square represents the time window of a task.

Figure B.8: Dataset SDU-TGR-3.
Appendix B. Datasets

(a) Spatial distribution of dataset SDU-TGR-4. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset SDU-TGR-4\_batch\_size1. Each square represents the time window of a task.

(c) Temporal distribution of dataset SDU-TGR-4\_batch\_size2. Each square represents the time window of a task.

(d) Temporal distribution of dataset SDU-TGR-4\_batch\_size4. Each square represents the time window of a task.

Figure B.9: Dataset SDU-TGR-4.
B.2. Spatial Clustered Tasks (SDC)

(a) Spatial distribution of dataset SDU-TGR-5. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset SDU-TGR-5_batch_size1. Each square represents the time window of a task.

(c) Temporal distribution of dataset SDU-TGR-5_batch_size2. Each square represents the time window of a task.

(d) Temporal distribution of dataset SDU-TGR-5_batch_size4. Each square represents the time window of a task.

Figure B.10: Dataset SDU-TGR-5.

B.2 Spatial Clustered Tasks (SDC)

B.2.1 SDC-TER

Spatial clustered tasks with number of tasks equal to the number of robots.
Appendix B. Datasets

(a) Spatial distribution of dataset SDC-TER-CR-1. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset SDC-TER-CR-1. Each square represents the time window of a task.

Figure B.11: Dataset SDC-TER-CR-1.

(a) Spatial distribution of dataset SDC-TER-CR-2. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset SDC-TER-CR-2. Each square represents the time window of a task.

Figure B.12: Dataset SDC-TER-CR-2.
B.2. Spatial Clustered Tasks (SDC)

(a) Spatial distribution of dataset SDC-TER-CR-3. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset SDC-TER-CR-3. Each square represents the time window of a task.

Figure B.13: Dataset SDC-TER-CR-3.

(a) Spatial distribution of dataset SDC-TER-CR-4. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset SDC-TER-CR-4. Each square represents the time window of a task.

Figure B.14: Dataset SDC-TER-CR-4.

B.2.2 SDC-TGR

Spatial clustered tasks with twice as many tasks as robots.
Appendix B. Datasets

(a) Spatial distribution of dataset SDC-TGR-CR-1. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset SDC-TGR-CR-1_batch_size1. Each square represents the time window of a task.

(c) Temporal distribution of dataset SDC-TGR-CR-1_batch_size2. Each square represents the time window of a task.

(d) Temporal distribution of dataset SDC-TGR-CR-1_batch_size4. Each square represents the time window of a task.

Figure B.15: Dataset SDC-TGR-CR-1.
B.2. Spatial Clustered Tasks (SDC)

(a) Spatial distribution of dataset SDC-TGR-CR-2. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset SDC-TGR-CR-2_batch_size1. Each square represents the time window of a task.

(c) Temporal distribution of dataset SDC-TGR-CR-2_batch_size2. Each square represents the time window of a task.

(d) Temporal distribution of dataset SDC-TGR-CR-2_batch_size4. Each square represents the time window of a task.

Figure B.16: Dataset SDC-TGR-CR-2.
Appendix B. Datasets

(a) Spatial distribution of dataset SDC-TGR-CR-3. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset SDC-TGR-CR-3_batch_size1. Each square represents the time window of a task.

(c) Temporal distribution of dataset SDC-TGR-CR-3_batch_size2. Each square represents the time window of a task.

(d) Temporal distribution of dataset SDC-TGR-CR-3_batch_size4. Each square represents the time window of a task.

Figure B.17: Dataset SDC-TGR-CR-3.
B.3. Temporal Uniformly Distributed Tasks (TDU)

(a) Spatial distribution of dataset SDC-TGR-CR-4. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset SDC-TGR-CR-4 \texttt{batch size1}. Each square represents the time window of a task.

(c) Temporal distribution of dataset SDC-TGR-CR-4 \texttt{batch size2}. Each square represents the time window of a task.

(d) Temporal distribution of dataset SDC-TGR-CR-4 \texttt{batch size4}. Each square represents the time window of a task.

Figure B.18: Dataset SDC-TGR-CR-4.

B.3 Temporal Uniformly Distributed Tasks (TDU)

B.3.1 TDU-TGR

Temporal uniformly distributed tasks with twice as many tasks as robots.
Appendix B. Datasets

(a) Spatial distribution of dataset TDU-TGR-1. Each line represents a task. The tail of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset TDU-TGR-1. Each square represents the time window of a task.

Figure B.19: Dataset TDU-TGR-1.

(a) Spatial distribution of dataset TDU-TGR-2. Each line represents a task. The tail of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset TDU-TGR-2. Each square represents the time window of a task.

Figure B.20: Dataset TDU-TGR-2.

217
B.3. Temporal Uniformly Distributed Tasks (TDU)

(a) Spatial distribution of dataset TDU-TGR-3. Each line represents a task. The tail of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset TDU-TGR-3. Each square represents the time window of a task.

Figure B.21: Dataset TDU-TGR-3.

(a) Spatial distribution of dataset TDU-TGR-4. Each line represents a task. The tail of a line is the pickup location and the * is the delivery location.

(b) Temporal distribution of dataset TDU-TGR-4. Each square represents the time window of a task.

Figure B.22: Dataset TDU-TGR-4.

B.3.2 TDU-ST

Temporal uniformly distributed tasks for testing task scalability. The number of tasks in each dataset increases by 10.
Appendix B. Datasets

(b) Spatial distribution of dataset TDU-TGR-5. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

Figure B.23: Dataset TDU-TGR-5.

(b) Temporal distribution of dataset TDU-TGR-5. Each square represents the time window of a task.

Figure B.24: Dataset TDU-ST-100.

(a) Spatial distribution of dataset TDU-ST-100. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

B.3.3 TDU-SR

Temporal uniformly distributed tasks for testing robot scalability. There are 100 tasks in the dataset.
B.4 Temporal Clustered Tasks (TDC)

B.4.1 TDC-TGR

Temporal clustered tasks with twice as many tasks as robots.

Figure B.25: Dataset TDU-SR-100.

(a) Spatial distribution of dataset TDU-SR-100. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location.

Figure B.26: Dataset TDU-TR-100. Each square represents the time window of a task.

(b) Temporal distribution of dataset TDU-TR-100. Each square represents the time window of a task.

Figure B.26: Dataset TDC-TGR-ITW-1.

(a) Spatial distribution of dataset TDC-TGR-ITW-1. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset TDC-TGR-ITW-1. Each square represents the time window of a task. Separation between time windows in a cluster is 1 second.

Figure B.26: Dataset TDC-TGR-ITW-1.
Appendix B. Datasets

(a) Spatial distribution of dataset TDC-TGR-ITW-2. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset TDC-TGR-ITW-2. Each square represents the time window of a task. Separation between time windows in a cluster is 1 second.

Figure B.27: Dataset TDC-TGR-ITW-2.

(a) Spatial distribution of dataset TDC-TGR-ITW-3. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset TDC-TGR-ITW-3. Each square represents the time window of a task. Separation between time windows in a cluster is 1 second.

Figure B.28: Dataset TDC-TGR-ITW-3.
B.4. Temporal Clustered Tasks (TDC)

(a) Spatial distribution of dataset TDC-TGR-ITW-1. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset TDC-TGR-ITW-4. Each square represents the time window of a task. Separation between time windows in a cluster is 1 second.

Figure B.29: Dataset TDC-TGR-ITW-4.

B.4.2 TDC-ST

Temporal clustered tasks for testing task scalability. The number of tasks in each dataset increases by 10.

(a) Spatial distribution of dataset TDC-ST-100. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset TDC-ST-100. Each square represents the time window of a task. Separation between time windows in a cluster is 10 seconds.

Figure B.30: Dataset TDC-ST-100.
Appendix B. Datasets

B.4.3 TDC-SR

Temporal clustered tasks for testing robot scalability. There are 100 tasks in the dataset.

(a) Spatial distribution of dataset TDC-SR-100. Each line represents a task. The tale of a line is the pickup location and the * is the delivery location. Each cluster is surrounded by a circle.

(b) Temporal distribution of dataset TDC-SR-100. Each square represents the time window of a task. Separation between time windows in a cluster is 4 seconds.

Figure B.31: Dataset TDU-SR-100.
B.4. Temporal Clustered Tasks (TDC)
C

Performance Metrics
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MURDOCH SSI TeSSI TeSSIIduo</td>
<td>MURDOCH SSI TeSSI TeSSIIduo</td>
<td>MURDOCH SSI TeSSI TeSSIIduo</td>
<td>MURDOCH SSI TeSSI TeSSIIduo</td>
</tr>
<tr>
<td><strong>Allocations</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Successful</td>
<td>4 4 4 4</td>
<td>4 4 4 4</td>
<td>4 4 4 4</td>
<td></td>
</tr>
<tr>
<td>Unsuccessful</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>4 4 4 4</td>
<td>4 4 4 4</td>
<td>4 4 4 4</td>
<td></td>
</tr>
<tr>
<td><strong>Bids</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 1</td>
<td>3 4 4 2</td>
<td>2 4 1 1</td>
<td>3 4 1 1</td>
<td></td>
</tr>
<tr>
<td>Robot 2</td>
<td>1 4 2 1</td>
<td>1 4 2 1</td>
<td>1 4 2 1</td>
<td></td>
</tr>
<tr>
<td>Robot 3</td>
<td>4 4 3 3</td>
<td>4 4 3 3</td>
<td>4 4 3 3</td>
<td></td>
</tr>
<tr>
<td>Robot 4</td>
<td>2 4 3 2</td>
<td>4 4 4 4</td>
<td>2 4 4 4</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>10 10 10 10</td>
<td>16 16 16 16</td>
<td>10 10 10 10</td>
<td></td>
</tr>
<tr>
<td><strong>Empty bids</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 1</td>
<td>1 0 3 3</td>
<td>2 0 3 3</td>
<td>1 0 3 3</td>
<td></td>
</tr>
<tr>
<td>Robot 2</td>
<td>3 0 2 1</td>
<td>3 0 2 3</td>
<td>3 0 2 2</td>
<td></td>
</tr>
<tr>
<td>Robot 3</td>
<td>0 1 0 0</td>
<td>1 0 1 1</td>
<td>0 0 1 0</td>
<td></td>
</tr>
<tr>
<td>Robot 4</td>
<td>2 0 0 2</td>
<td>0 0 0 0</td>
<td>2 0 0 1</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>6 0 6 6</td>
<td>6 0 6 6</td>
<td>6 0 6 6</td>
<td></td>
</tr>
<tr>
<td><strong>Messages</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sent</td>
<td>8 8 8 8</td>
<td>8 8 8 8</td>
<td>8 8 8 8</td>
<td></td>
</tr>
<tr>
<td>Received</td>
<td>17 21 21 21</td>
<td>17 21 21 21</td>
<td>17 21 21 21</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>25 29 29 29</td>
<td>25 29 29 29</td>
<td>25 29 29 29</td>
<td></td>
</tr>
<tr>
<td><strong>Travel distance</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 1</td>
<td>13.62 44.01 13.62 12.64 12.64 15.22 12.64 9.51 24.55 9.51 9.51</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 3</td>
<td>28.64 0 25.44 25.44 18.11 18.11 18.11 29.31 0 25.28 29.31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 4</td>
<td>19.0 19.0 17.61 17.61 39.13 0 39.13 22.42 22.42 22.42</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>82.87 63.01 78.28 78.28 81.16 81.16 81.16 80.85 66.58 80.39 80.85</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>% Robot usage</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100 50 100 100 100 75 100 100 75 100 100 100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Makespan [s]</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 1</td>
<td>- - 5.05 5.05 - - 5.61 9.0 - - 7.11 7.11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 2</td>
<td>- - 10.82 10.82 - - 9.0 5.61 - - 12.44 16.27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 3</td>
<td>- - 10.87 10.87 - - 13.21 13.21 - - 16.27 16.24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Time [s]</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. per task</td>
<td>0.258 0.260 0.262 0.260 0.258 0.258 0.261 0.260 0.257 0.258 0.261 0.257</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allocation</td>
<td>1.032 1.041 1.046 1.04 1.033 1.043 1.042 1.041 1.026 1.033 1.044 1.028</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Appendix C. Performance Metrics

### Experiment 2

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Allocations</strong></td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Successful</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Unsuccessful</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td><strong>Bids</strong></td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Robot 1</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Robot 2</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Robot 3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Robot 4</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td><strong>Empty bids</strong></td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Robot 1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Robot 2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Robot 3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Robot 4</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Messages</strong></td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Sent</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Received</td>
<td>17</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>Total</td>
<td>25</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td><strong>Travel distance [m]</strong></td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Robot 1</td>
<td>7.18</td>
<td>0</td>
<td>7.18</td>
</tr>
<tr>
<td>Robot 2</td>
<td>12.32</td>
<td>0</td>
<td>15.03</td>
</tr>
<tr>
<td>Robot 3</td>
<td>16.92</td>
<td>0</td>
<td>14.98</td>
</tr>
<tr>
<td>Robot 4</td>
<td>4.25</td>
<td>9.7</td>
<td>4.25</td>
</tr>
<tr>
<td>Total</td>
<td>40.67</td>
<td>9.7</td>
<td>41.44</td>
</tr>
<tr>
<td><strong>% Robot usage</strong></td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Robot 1</td>
<td>25</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>Robot 2</td>
<td>25</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>Robot 3</td>
<td>25</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>Robot 4</td>
<td>25</td>
<td>100</td>
<td>25</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td>25</td>
<td>100</td>
</tr>
<tr>
<td><strong>Makespan [s]</strong></td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Robot 1</td>
<td>-</td>
<td>-</td>
<td>0.58</td>
</tr>
<tr>
<td>Robot 2</td>
<td>-</td>
<td>-</td>
<td>0.84</td>
</tr>
<tr>
<td>Robot 3</td>
<td>-</td>
<td>-</td>
<td>1.19</td>
</tr>
<tr>
<td>Robot 4</td>
<td>-</td>
<td>-</td>
<td>1.2</td>
</tr>
<tr>
<td>Total</td>
<td>-</td>
<td>-</td>
<td>1.2</td>
</tr>
<tr>
<td><strong>Time [s]</strong></td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Avg. per task</td>
<td>0.259</td>
<td>0.262</td>
<td>0.261</td>
</tr>
<tr>
<td>Allocation</td>
<td>1.035</td>
<td>1.047</td>
<td>1.045</td>
</tr>
</tbody>
</table>

## Experiment 4

### Dataset: SDC-TGR-CR-1

<table>
<thead>
<tr>
<th>Metric</th>
<th>MRTA Algorithms</th>
<th>MRTA Algorithms</th>
<th>MRTA Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocations</td>
<td>MURDOCH SSI TeSSI TeSSIduo</td>
<td>MURDOCH SSI TeSSI TeSSIduo</td>
<td>MURDOCH SSI TeSSI TeSSIduo</td>
</tr>
<tr>
<td>Successful</td>
<td>8 8 8 8</td>
<td>8 8 8 8</td>
<td>8 8 8 8</td>
</tr>
<tr>
<td>Unsuccessful</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>Total</td>
<td>8 8 8 8</td>
<td>8 8 8 8</td>
<td>8 8 8 8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bids</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot 1</td>
<td>8 8 8 8</td>
<td>7 8 8 8</td>
<td>7 8 8 8</td>
</tr>
<tr>
<td>Robot 2</td>
<td>8 8 8 8</td>
<td>8 8 8 8</td>
<td>4 8 8 8</td>
</tr>
<tr>
<td>Robot 3</td>
<td>8 8 8 8</td>
<td>8 8 8 8</td>
<td>2 8 8 7</td>
</tr>
<tr>
<td>Robot 4</td>
<td>8 8 8 8</td>
<td>6 8 8 8</td>
<td>2 8 8</td>
</tr>
<tr>
<td>Total</td>
<td>32 32 32 32</td>
<td>28 32 32 32</td>
<td>20 32 32 31</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Empty bids</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot 1</td>
<td>0 0 0 0</td>
<td>1 0 0 0</td>
<td>1 0 0 0</td>
</tr>
<tr>
<td>Robot 2</td>
<td>0 0 0 0</td>
<td>1 0 0 0</td>
<td>4 0 0 0</td>
</tr>
<tr>
<td>Robot 3</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>1 0 0 0</td>
</tr>
<tr>
<td>Robot 4</td>
<td>2 0 0 0</td>
<td>6 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>Total</td>
<td>0 0 0 0</td>
<td>4 0 0 0</td>
<td>12 0 0 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Messages</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sent</td>
<td>16 16 16</td>
<td>16 16 16</td>
<td>16 16 16</td>
</tr>
<tr>
<td>Received</td>
<td>39 47 47</td>
<td>35 43 43</td>
<td>33 41 41</td>
</tr>
<tr>
<td>Total</td>
<td>55 63 63</td>
<td>51 59 59</td>
<td>49 57 57</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Travel distance [m]</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot 2</td>
<td>0 0 0</td>
<td>11.48 0 22.46</td>
<td>0.18 0 18.73 0</td>
</tr>
<tr>
<td>Robot 3</td>
<td>0 0 0</td>
<td>0</td>
<td>17.95 0 17.11 10.13</td>
</tr>
<tr>
<td>Total</td>
<td>25.84 25.84</td>
<td>29.74 29.74</td>
<td>39.39 39.39 42.69 42.69</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>% Robot usage</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot 1</td>
<td>50 50</td>
<td>50</td>
<td>50 50</td>
</tr>
<tr>
<td>Robot 2</td>
<td>0 0</td>
<td>25</td>
<td>0 25</td>
</tr>
<tr>
<td>Robot 3</td>
<td>0 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Robot 4</td>
<td>50 50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Total</td>
<td>50 50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Makespan [s]</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot 1</td>
<td>- -</td>
<td>7.43</td>
<td>3.29</td>
</tr>
<tr>
<td>Robot 2</td>
<td>- -</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>Robot 3</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>Robot 4</td>
<td>0 0</td>
<td>4.12</td>
<td>0 0</td>
</tr>
<tr>
<td>Total</td>
<td>- -</td>
<td>7.43 7.43</td>
<td>- -</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time [s]</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
<th>MURDOCH SSI TeSSI TeSSIduo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. per batch</td>
<td>0.26 0.259 0.260</td>
<td>0.518 0.520 0.521</td>
<td>1.037 1.041 1.046</td>
</tr>
<tr>
<td>Allocation</td>
<td>2.082 2.075 2.076</td>
<td>2.075 2.086 2.086</td>
<td>2.074 2.083 2.092</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Metric</th>
<th>Dataset: TDU-TGR-1</th>
<th>Dataset: TDU-TGR-2</th>
<th>Dataset: TDU-TGR-3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Allocations</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Successful</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Unsuccessful</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td><strong>Bids</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Robot 2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Robot 3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Robot 4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td><strong>Empty bids</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 1</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Robot 2</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Robot 3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Robot 4</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Total</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td><strong>Messages</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sent</td>
<td>12</td>
<td>16</td>
<td>12</td>
</tr>
<tr>
<td>Received</td>
<td>33</td>
<td>41</td>
<td>33</td>
</tr>
<tr>
<td>Total</td>
<td>45</td>
<td>57</td>
<td>45</td>
</tr>
<tr>
<td><strong>Travel distance [m]</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 1</td>
<td>4.92</td>
<td>11.6</td>
<td>19.87</td>
</tr>
<tr>
<td>Robot 2</td>
<td>6.59</td>
<td>25.77</td>
<td>22.2</td>
</tr>
<tr>
<td>Robot 3</td>
<td>9.75</td>
<td>0</td>
<td>33.55</td>
</tr>
<tr>
<td>Robot 4</td>
<td>7.48</td>
<td>10.63</td>
<td>23.43</td>
</tr>
<tr>
<td>Total</td>
<td>28.74</td>
<td>48</td>
<td>99.05</td>
</tr>
<tr>
<td><strong>% Robot usage</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 1</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Robot 2</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Robot 3</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Robot 4</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td><strong>Makespan [s]</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 1</td>
<td>-</td>
<td>-</td>
<td>14.91</td>
</tr>
<tr>
<td>Robot 2</td>
<td>-</td>
<td>-</td>
<td>14.54</td>
</tr>
<tr>
<td>Robot 3</td>
<td>-</td>
<td>-</td>
<td>20</td>
</tr>
<tr>
<td>Robot 4</td>
<td>-</td>
<td>-</td>
<td>16</td>
</tr>
<tr>
<td>Total</td>
<td>-</td>
<td>-</td>
<td>69.91</td>
</tr>
<tr>
<td><strong>Time [s]</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. per task</td>
<td>0.259</td>
<td>0.262</td>
<td>0.264</td>
</tr>
<tr>
<td>Allocation</td>
<td>1.035</td>
<td>2.099</td>
<td>2.109</td>
</tr>
<tr>
<td>Total</td>
<td>7.109</td>
<td>7.205</td>
<td>7.334</td>
</tr>
</tbody>
</table>

Experiment 5
### Experiment 6

<table>
<thead>
<tr>
<th>Metric</th>
<th>MRTA Algorithms</th>
<th>MRTA Algorithms</th>
<th>MRTA Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Allocations</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Successful</td>
<td>MURDOCH 8</td>
<td>SSI 8</td>
<td>TeSSI 8</td>
</tr>
<tr>
<td>Unsuccessful</td>
<td>MURDOCH 4</td>
<td>SSI 0</td>
<td>TeSSI 0</td>
</tr>
<tr>
<td>Total</td>
<td>MURDOCH 8</td>
<td>SSI 8</td>
<td>TeSSI 8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Bids</strong></th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot 1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>10</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>10</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Empty bids</strong></th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot 1</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 3</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>22</td>
<td>0</td>
<td></td>
<td></td>
<td>22</td>
<td>0</td>
<td></td>
<td></td>
<td>22</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Messages</strong></th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sent</td>
<td>12</td>
<td></td>
<td>16</td>
<td></td>
<td>16</td>
<td></td>
<td>16</td>
<td></td>
<td>16</td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Received</td>
<td>33</td>
<td>41</td>
<td>41</td>
<td>41</td>
<td>33</td>
<td>41</td>
<td>41</td>
<td>41</td>
<td>33</td>
<td>41</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>Total</td>
<td>45</td>
<td>57</td>
<td>57</td>
<td>57</td>
<td>45</td>
<td>57</td>
<td>57</td>
<td>57</td>
<td>45</td>
<td>57</td>
<td>57</td>
<td>57</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Travel distance [m]</strong></th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot 1</td>
<td>7.98</td>
<td></td>
<td>26.62</td>
<td></td>
<td>8.93</td>
<td></td>
<td>19.31</td>
<td></td>
<td>25.1</td>
<td></td>
<td>8.51</td>
<td></td>
</tr>
<tr>
<td>Robot 2</td>
<td>10.34</td>
<td></td>
<td>24.72</td>
<td></td>
<td>10.18</td>
<td></td>
<td>29.13</td>
<td></td>
<td>10.18</td>
<td></td>
<td>7.94</td>
<td></td>
</tr>
<tr>
<td>Robot 3</td>
<td>9.18</td>
<td></td>
<td>10.85</td>
<td></td>
<td>17.08</td>
<td></td>
<td>17.48</td>
<td></td>
<td>17.48</td>
<td></td>
<td>9.04</td>
<td></td>
</tr>
<tr>
<td>Robot 4</td>
<td>14.54</td>
<td></td>
<td>14.54</td>
<td></td>
<td>15.72</td>
<td></td>
<td>25.62</td>
<td></td>
<td>29.06</td>
<td></td>
<td>5.04</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>42.04</td>
<td></td>
<td>55.81</td>
<td></td>
<td>76.73</td>
<td></td>
<td>58.31</td>
<td></td>
<td>61.2</td>
<td></td>
<td>31.86</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>% Robot usage</strong></th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot 1</td>
<td>25</td>
<td></td>
<td>37.5</td>
<td></td>
<td>25</td>
<td></td>
<td>37.5</td>
<td></td>
<td>25</td>
<td></td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Robot 2</td>
<td>25</td>
<td>50</td>
<td>37.5</td>
<td>12.5</td>
<td>25</td>
<td>50</td>
<td>12.5</td>
<td>12.5</td>
<td>25</td>
<td>50</td>
<td>25</td>
<td>12.5</td>
</tr>
<tr>
<td>Robot 3</td>
<td>25</td>
<td>50</td>
<td>12.5</td>
<td>25</td>
<td>25</td>
<td>50</td>
<td>12.5</td>
<td>0</td>
<td>25</td>
<td>50</td>
<td>25</td>
<td>37.5</td>
</tr>
<tr>
<td>Robot 4</td>
<td>25</td>
<td>0</td>
<td>12.5</td>
<td>25</td>
<td>25</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>25</td>
<td>50</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td>100</td>
<td>75</td>
<td>75</td>
<td>100</td>
<td>50</td>
<td>100</td>
<td>75</td>
<td>100</td>
<td>50</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Makespan [s]</strong></th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot 1</td>
<td>-</td>
<td>-</td>
<td>24</td>
<td>24</td>
<td>-</td>
<td>-</td>
<td>15</td>
<td>26</td>
<td>-</td>
<td>-</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Robot 2</td>
<td>-</td>
<td>-</td>
<td>24</td>
<td>41</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Robot 3</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>34</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>16</td>
<td>28</td>
</tr>
<tr>
<td>Robot 4</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>37</td>
<td>37</td>
<td>-</td>
<td>-</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Total</td>
<td>-</td>
<td>-</td>
<td>117</td>
<td>117</td>
<td>-</td>
<td>-</td>
<td>214</td>
<td>214</td>
<td>-</td>
<td>-</td>
<td>368</td>
<td>368</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Time [s]</strong></th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. per task</td>
<td>0.504</td>
<td>0.264</td>
<td>0.266</td>
<td>0.265</td>
<td>0.505</td>
<td>0.265</td>
<td>0.265</td>
<td>0.265</td>
<td>0.506</td>
<td>0.265</td>
<td>0.263</td>
<td>0.266</td>
</tr>
<tr>
<td>Allocation</td>
<td>2.018</td>
<td>2.108</td>
<td>2.129</td>
<td>2.120</td>
<td>2.022</td>
<td>2.118</td>
<td>2.123</td>
<td>2.117</td>
<td>2.023</td>
<td>2.119</td>
<td>2.105</td>
<td>2.129</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>Dataset: TDU-ST-10</th>
<th>Dataset: TDU-ST-50</th>
<th>Dataset: TDU-ST-100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRTA Algorithms</td>
<td>MRTA Algorithms</td>
<td>MRTA Algorithms</td>
</tr>
<tr>
<td></td>
<td>MURDOCH SSI TeSSI TeSSIduo</td>
<td>MURDOCH SSI TeSSI TeSSIduo</td>
<td>MURDOCH SSI TeSSI TeSSIduo</td>
</tr>
<tr>
<td>Allocations</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Successful</td>
<td>4</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Unsuccessful</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Bids</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 1</td>
<td>3</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Robot 2</td>
<td>1</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Robot 3</td>
<td>4</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Robot 4</td>
<td>2</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
<td>40</td>
<td>37</td>
</tr>
<tr>
<td>Empty bids</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 1</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Robot 2</td>
<td>9</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Robot 3</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Robot 4</td>
<td>8</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>30</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Messages</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sent</td>
<td>14</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Received</td>
<td>41</td>
<td>51</td>
<td>51</td>
</tr>
<tr>
<td>Total</td>
<td>55</td>
<td>71</td>
<td>71</td>
</tr>
<tr>
<td>Travel distance [m]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 1</td>
<td>9.33</td>
<td>0</td>
<td>20.68</td>
</tr>
<tr>
<td>Robot 2</td>
<td>6.88</td>
<td>40.82</td>
<td>28.91</td>
</tr>
<tr>
<td>Robot 3</td>
<td>13.32</td>
<td>0</td>
<td>29.75</td>
</tr>
<tr>
<td>Robot 4</td>
<td>6.99</td>
<td>25.08</td>
<td>30.26</td>
</tr>
<tr>
<td>Total</td>
<td>36.52</td>
<td>65.90</td>
<td>109.64</td>
</tr>
<tr>
<td>% Robot usage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 1</td>
<td>25</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>Robot 2</td>
<td>25</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>Robot 3</td>
<td>25</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>Robot 4</td>
<td>25</td>
<td>40</td>
<td>30</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>Makespan [s]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot 1</td>
<td>-</td>
<td>-</td>
<td>22</td>
</tr>
<tr>
<td>Robot 2</td>
<td>-</td>
<td>-</td>
<td>21</td>
</tr>
<tr>
<td>Robot 3</td>
<td>-</td>
<td>-</td>
<td>30</td>
</tr>
<tr>
<td>Robot 4</td>
<td>-</td>
<td>-</td>
<td>24.22</td>
</tr>
<tr>
<td>Total</td>
<td>-</td>
<td>-</td>
<td>57</td>
</tr>
<tr>
<td>Time [s]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. per task</td>
<td>0.510</td>
<td>0.267</td>
<td>0.265</td>
</tr>
<tr>
<td>Allocation</td>
<td>2.04</td>
<td>2.667</td>
<td>2.653</td>
</tr>
<tr>
<td>Total</td>
<td>12.177</td>
<td>8.784</td>
<td>8.593</td>
</tr>
</tbody>
</table>

## Experiment 8

### Performance Metrics

#### Dataset: TDC-ST-10

<table>
<thead>
<tr>
<th>Metric</th>
<th>Allocations</th>
<th>MRTA Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Successful</strong></td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td><strong>Unsuccessful</strong></td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

#### Dataset: TDC-ST-50

<table>
<thead>
<tr>
<th>Metric</th>
<th>Allocations</th>
<th>MRTA Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Successful</strong></td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td><strong>Unsuccessful</strong></td>
<td>46</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

#### Dataset: TDC-ST-100

<table>
<thead>
<tr>
<th>Metric</th>
<th>Allocations</th>
<th>MRTA Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Successful</strong></td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td><strong>Unsuccessful</strong></td>
<td>96</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

#### Bids

<table>
<thead>
<tr>
<th>Robot</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSIduo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot 1</td>
<td>2</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Robot 2</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Robot 3</td>
<td>4</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Robot 4</td>
<td>3</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
</tbody>
</table>

#### Empty bids

<table>
<thead>
<tr>
<th>Robot</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSIduo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot 1</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Robot 2</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Robot 3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Robot 4</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Messages

<table>
<thead>
<tr>
<th>Metric</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSIduo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sent</strong></td>
<td>14</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td><strong>Received</strong></td>
<td>41</td>
<td>51</td>
<td>51</td>
<td>51</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>55</td>
<td>71</td>
<td>71</td>
<td>71</td>
</tr>
</tbody>
</table>

#### Travel distance [m]

<table>
<thead>
<tr>
<th>Robot</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSIduo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot 1</td>
<td>9.04</td>
<td>0</td>
<td>29.66</td>
<td>7.96</td>
</tr>
<tr>
<td>Robot 2</td>
<td>9.31</td>
<td>37.07</td>
<td>15.97</td>
<td>15.97</td>
</tr>
<tr>
<td>Robot 3</td>
<td>19.9</td>
<td>0</td>
<td>30.11</td>
<td>34.36</td>
</tr>
<tr>
<td>Robot 4</td>
<td>10.11</td>
<td>24.55</td>
<td>17.84</td>
<td>21.56</td>
</tr>
<tr>
<td>Total</td>
<td>48.36</td>
<td>61.62</td>
<td>93.58</td>
<td>79.85</td>
</tr>
</tbody>
</table>

#### % Robot usage

<table>
<thead>
<tr>
<th>Robot</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSIduo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot 1</td>
<td>25</td>
<td>0</td>
<td>40</td>
<td>10</td>
</tr>
<tr>
<td>Robot 2</td>
<td>25</td>
<td>60</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Robot 3</td>
<td>25</td>
<td>0</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>Robot 4</td>
<td>25</td>
<td>40</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td>50</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

#### Makespan [s]

<table>
<thead>
<tr>
<th>Robot</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSIduo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot 1</td>
<td>-</td>
<td>-</td>
<td>364</td>
<td>4</td>
</tr>
<tr>
<td>Robot 2</td>
<td>-</td>
<td>58</td>
<td>58</td>
<td>-</td>
</tr>
<tr>
<td>Robot 3</td>
<td>-</td>
<td>112</td>
<td>364</td>
<td>-</td>
</tr>
<tr>
<td>Robot 4</td>
<td>-</td>
<td>130</td>
<td>148</td>
<td>-</td>
</tr>
<tr>
<td>Total</td>
<td>-</td>
<td>-</td>
<td>1578</td>
<td>1578</td>
</tr>
</tbody>
</table>

#### Time [s]

<table>
<thead>
<tr>
<th>Metric</th>
<th>MURDOCH</th>
<th>SSI</th>
<th>TeSSI</th>
<th>TeSSIduo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Avg. per task</strong></td>
<td>0.260</td>
<td>0.260</td>
<td>0.264</td>
<td>0.264</td>
</tr>
<tr>
<td><strong>Allocation</strong></td>
<td>1.041</td>
<td>2.656</td>
<td>2.639</td>
<td>2.643</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>8.744</td>
<td>8.416</td>
<td>8.666</td>
<td>9.991</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>Robots: 10 MRTA Algorithms</th>
<th>Robots: 50 MRTA Algorithms</th>
<th>Robots: 100 MRTA Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Allocations</strong></td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Successful</td>
<td>10</td>
<td>100</td>
<td>45</td>
</tr>
<tr>
<td>Unsuccessful</td>
<td>90</td>
<td>0</td>
<td>55</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td><strong>Bids</strong></td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Total</td>
<td>55</td>
<td>1000</td>
<td>387</td>
</tr>
<tr>
<td><strong>Empty bids</strong></td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Total</td>
<td>945</td>
<td>0</td>
<td>73</td>
</tr>
<tr>
<td><strong>Sent</strong></td>
<td>110</td>
<td>200</td>
<td>91</td>
</tr>
<tr>
<td><strong>Received</strong></td>
<td>1001</td>
<td>1101</td>
<td>506</td>
</tr>
<tr>
<td>Total</td>
<td>1111</td>
<td>1301</td>
<td>597</td>
</tr>
<tr>
<td><strong>Travel distance [m]</strong></td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Total</td>
<td>241.62</td>
<td>1055.67</td>
<td>984.58</td>
</tr>
<tr>
<td><strong>% Robot usage</strong></td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td><strong>Makespan [s]</strong></td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Total</td>
<td>-</td>
<td>-</td>
<td>97.13</td>
</tr>
<tr>
<td><strong>Time [s]</strong></td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Avg. per task</td>
<td>0.267</td>
<td>0.553</td>
<td>0.529</td>
</tr>
<tr>
<td>Allocation</td>
<td>2.666</td>
<td>55.304</td>
<td>23.805</td>
</tr>
<tr>
<td>Total</td>
<td>81.665</td>
<td>137.133</td>
<td>57.01</td>
</tr>
</tbody>
</table>

## Appendix C. Performance Metrics

### Experiment 10. Dataset: TDC-SR-100

<table>
<thead>
<tr>
<th>Metric</th>
<th>Robots: 10 MRTA Algorithms</th>
<th>Robots: 50 MRTA Algorithms</th>
<th>Robots: 100 MRTA Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocations</td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Successful</td>
<td>10</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Unsuccessful</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Bids</td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Total</td>
<td>55</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Empty bids</td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Total</td>
<td>945</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Messages</td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Sent</td>
<td>110</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>Received</td>
<td>1001</td>
<td>1101</td>
<td>1101</td>
</tr>
<tr>
<td>Total</td>
<td>1111</td>
<td>1301</td>
<td>1301</td>
</tr>
<tr>
<td>Travel distance [m]</td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Total</td>
<td>398.35</td>
<td>558.67</td>
<td>1156.31</td>
</tr>
<tr>
<td>% Robot usage</td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>Makespan [s]</td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Total</td>
<td>-</td>
<td>-</td>
<td>1349</td>
</tr>
<tr>
<td>Time [s]</td>
<td>MURDOCH</td>
<td>SSI</td>
<td>TeSSI</td>
</tr>
<tr>
<td>Avg. per task</td>
<td>0.267</td>
<td>0.825</td>
<td>1.399</td>
</tr>
<tr>
<td>Total</td>
<td>81.57</td>
<td>156.827</td>
<td>213.202</td>
</tr>
</tbody>
</table>

References


References


