UNIVERSITY OF LIVERPOOL

Mechanism Selection for Multi-Robot Task Allocation

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy by

Eric Schneider

February 2018

Contents

List of Figures

List of Tables

List of Algorithms

List of Code Listings

Notations

The following notations and abbreviations are found throughout this thesis:

R	A set of robots that comprises a *team*
T	A set of tasks that comprises a *scenario*
n	The number of robots in a team R (i.e., $\lvert R \rvert$)
m	The number of tasks in a scenario T (i.e., $\lvert T \rvert$)
ρ	A subset (or *sub-team*) of robots in R
$T(\rho)$	The set of tasks assigned to sub-team ρ
τ	A subset of tasks in T
t	A single task in T
$t.req$	Number of robots required to complete task t
$t.arr$	Arrival time of task t
SR	A *Single-Robot* environment
MR	A *Multi-Robot* environment
IT	A *Independent* environment
CT	A *Constrained* environment (e.g., by precedence)
SA	A *Static Allocation* environment
DA	A *Dynamic Allocation* environment
RR	Round-Robin allocation
OSI	Ordered Single-Item allocation
SSI	Sequential Single-Item allocation
PSI	Parallel Single-Item allocation
SEL	A dynamic mechanism selection method

Preface

This thesis is primarily my own work. The sources of other materials are identifed.

Abstract

There is increasing interest in fielding multi-robot teams for applications such as search and rescue, warehouse automation, and delivery of consumer goods. Task allocation is an important problem to solve in such multi-robot settings. Given a mission that can be decomposed into discrete tasks, the Multi-Robot Task Allocation (MRTA) problem looks for an assignment of tasks to robots that ultimately results in efficient execution of the mission. There is a range of approaches to this optimisation problem, from centralised solvers to fully distributed methods that involve no explicit coordination between team members. Somewhere in the middle of this range lie *market-based* approaches, where tasks can be treated as goods, robots as "buyers" who can compute and express their own preferences for tasks in a virtual marketplace, and some clearing mechanism exists to match tasks to robots according to these preferences.

The most common type of market-based mechanism for multi-robot task allocation is an *auction*, in which tasks are announced to the team, robots compute and place bids that encode some measure of cost or utility of performing the tasks, and tasks are awarded to robots over a number of rounds, according to the particular rules of the mechanism. Many different auction mechanisms exist, and they vary in the trade-offs that they make between computation time and space on the one hand, and performance of the execution of the mission on the other. In addition, the performance that results from a mechanism's allocation can be greatly affected by properties of task environments—the spatial and temporal arrangements of tasks, as well as other properties like precedence constraints, whether tasks require the simultaneous cooperation of multiple robots, and so on—in which it is employed. A simple mechanism that is inexpensive to compute and scales well may perform well in some environments, but not in others.

The work presented in this thesis focuses on this relationship between auction-based task allocation mechanisms and properties of task environments, with the goal of developing a method of selecting, from a portfolio, a mechanism that is appropriate for a given task environment. The first part of this work is an empirical performance evaluation of a range of mechanisms employed in a series of environments of increasing complexity. The second part of this work uses results from this evaluation to develop and train a data-driven method of mechanism selection using properties of environments that can be measured at the start of a mission. The results show that, under certain conditions, this method of mechanism selection can lead to significant performance improvements compared to using a single mechanism alone.

Acknowledgements

I am extremely grateful to my supervisors Elizabeth Sklar, Simon Parsons, and Karl Tuyls for their guidance and help. They have given me many opportunities over the years I have known them. They have treated me like family and have become my colleagues and friends.

I am also grateful to my advisors Rahul Savani and Katie Atkinson, and to Professor Susan Epstein at the City University of New York (CUNY). Susan gave me my first research opportunities while I was still an undergraduate, and continues to do so for cohort after cohort of computer science students.

To my friends at the University of Liverpool, without whom I could not have made it through these four years: Richard Williams, David Geleta, Matoula Kotsialou, Christof Spanring, Daan Bloembergen, Bastian Broecker, Daniel Claes, and Joscha Fossel and many others.

I am most grateful to my family, who are the most important people in the world to me. To my sisters Jenny and Aimee, to my mother, Trip, and especially to my father, Ronald, thank you for being there over all these years. I love you.

Chapter 1

Introduction

Mobile robots are being asked to perform difficult missions in increasingly complex and dynamic environments. These kinds of missions can be found in exploration, search and rescue, defence, and industrial settings, among others. The multi-task nature and scale of these missions necessitates the use of teams rather than single robots. While robot teams have the attractive potential advantage of distributing and parallelising a mission workload amongst team members, they also pose significant coordination challenges that do not appear in single-robot settings. Some of these challenges include representing a mission in a way that can be decomposed into tasks that can be allocated to team members; ensuring that team members cooperate while performing tasks instead of interfering with and hindering each other; recovering from failure of one or more robots during the course of a mission; or coping with poor or unreliable communication links between team members or a human controller. Unreliable communication links between a robot team and human controllers can make some form of autonomous decision making necessary.

One of the primary challenges of multi-robot coordination is *multi-robot task allocation* (**MRTA**), the problem of deciding which tasks of a mission should be assigned to which robots so that the overall execution of the mission is, by some measure, efficient if not optimal. While there are several kinds of approaches to solving task allocation problems (explored Chapter 2), this thesis focuses on *market-based* methods of task allocation, and *auctions* in particular.

Market-based Task Allocation

Market-based approaches to task allocation frame the assignment problem as a multi-agent systems (MAS) problem. Rather than having a monolithic, centralised planner that is responsible for computing the costs or utilities of potential allocations, a market-based approach to MRTA relies on the fact that robot team members are each capable of planning subsets or sub-problems of the mission (i.e., planning to execute individual tasks or groups of tasks) and can express the costs or utilities of these plans in a way that is simple and efficient to communicate. Task allocation is governed by a *mechanism*, a

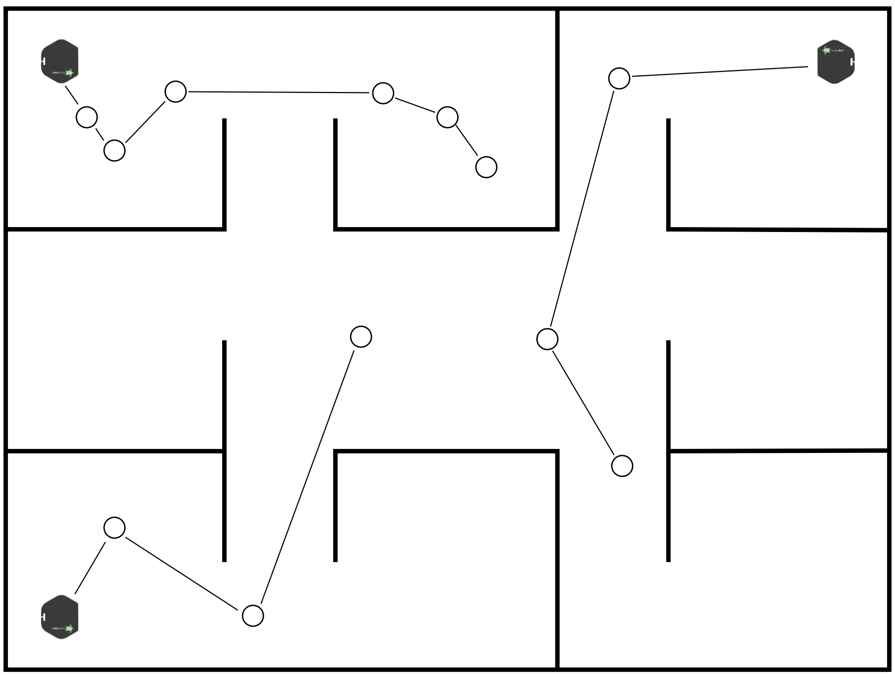


Figure 1.1: A multi-robot routing problem. Black hexagons represent robot locations. Circles represent locations of tasks. In this solution, tasks have all been allocated to robots, who plan routes (straight lines) to visit task locations.

set of rules that govern how tasks should be assigned and a protocol for communicating the availability of tasks to robots, and the values robots have for them. A mechanism enables a virtual marketplace in which tasks can be distributed to robots or exchanged among them.

The most common kind of market-based mechanism for multi-robot MRTA is an *auction*, which compares bids for resources from interested parties and awards them to the highest (or lowest) bidder according to the particular rules of a mechanism. (Auctions are discussed in detail in Chapter 2). It can be expensive to compute an allocation that is optimal for some performance objective, so most auction mechanisms strive for approximately optimal allocations. Designers of auction mechanisms must make trade-offs between the costs of computing an allocation and the performance of the execution of a mission that results from the allocation.

Multi-Robot Routing

This thesis examines auction-based MRTA mechanisms for a class of problems known as multi-robot routing. In a multi-robot routing mission, a team of robots must travel from their starting locations to a number of *task locations* that are distributed over some geographic area. The robots must avoid obstacles such as walls (and each other) as they travel to their assigned task locations. The mission is considered complete when all of the task locations have been visited. An example problem and its solution is shown in Figure 1.1.

In these types of missions, the aim of a task allocation mechanism is to assign tasks to robots such a way that, when robots execute the tasks, some global measure of cost is minimised or some measure of utility is maximised. Cost is often computed as the distance covered by the team as they travel to task locations or the time it takes them to do so. Utility might be measured as a reward for reaching a task location that has a high priority, for example a victim in distress in a search and rescue setting.

For simple routing missions in which task locations are only visited by single robots and there are no constraints like precedence-ordering between tasks, multi-robot routing can be formulated as a *multiple travelling salesperson* (mTSP) [7] or *vehicle routing problem* (VRP) [72]. However, as explained in Chapter 2, multi-robot tasks and environments may become complicated in ways that make traditional methods of solving problems like this infeasible. In such cases, market-based mechanisms such as auctions can be scalable, practical to implement, and still yield good performance compared to optimal solutions.

Task Environments

A *map* defines the physical boundaries of a mission space as well as the obstacles within that space, such as walls or barriers. Robot team members have an initial arrangement–a set of *starting locations*–on the map. Tasks that make up a mission also have a spatial arrangement on the map. Tasks may also have other properties:

- A task may require multiple robots to visit its location simultaneously before it can be considered complete
- There may be precedence-ordering or other constraints between tasks that dictate when or in which order task locations may be visited
- A tasks may “arrive” over time according to a known or unknown schedule
- A task may have a priority, raising the utility of visiting its location

A *task environment* characterises these properties.

In missions more complex than multi-robot routing, the execution of a task, once its location has been reached, may itself require elaborate planning. Previous work has observed that the environments in which auction mechanisms are employed have a large impact on mission execution performance [106, 108, 110]. This impact can lead to performance results that mechanism designs might not be able to consider or predict.

Applications

Task allocation is a fundamental problem in a number of multi-robot applications. Robots exploring remote areas (Figure 1.2) need to navigate autonomously, as delayed signals make real-time control impossible. A future is envisioned in which teams of rovers

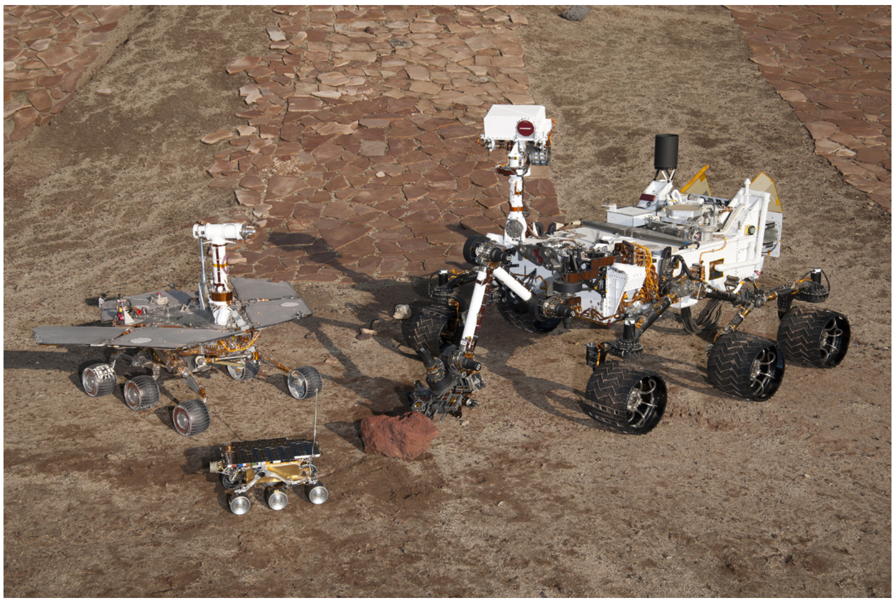


FIGURE 1.2: NASA's autonomous rovers explore the surface of Mars. Courtesy NASA/JPL-Caltech.

explore Mars [22, 39] or perform underwater archaeological surveys [123] by distributing high-level mission workloads among themselves autonomously.

There is also increasing interest in employing robots for search and rescue [17, 55, 56, 85] and disaster recovery [66] missions (Figure 1.3) in environments, such as nuclear disasters [84] or collapsed mines [83], that are not only dangerous for human responders to enter, but also make real-time control difficult.

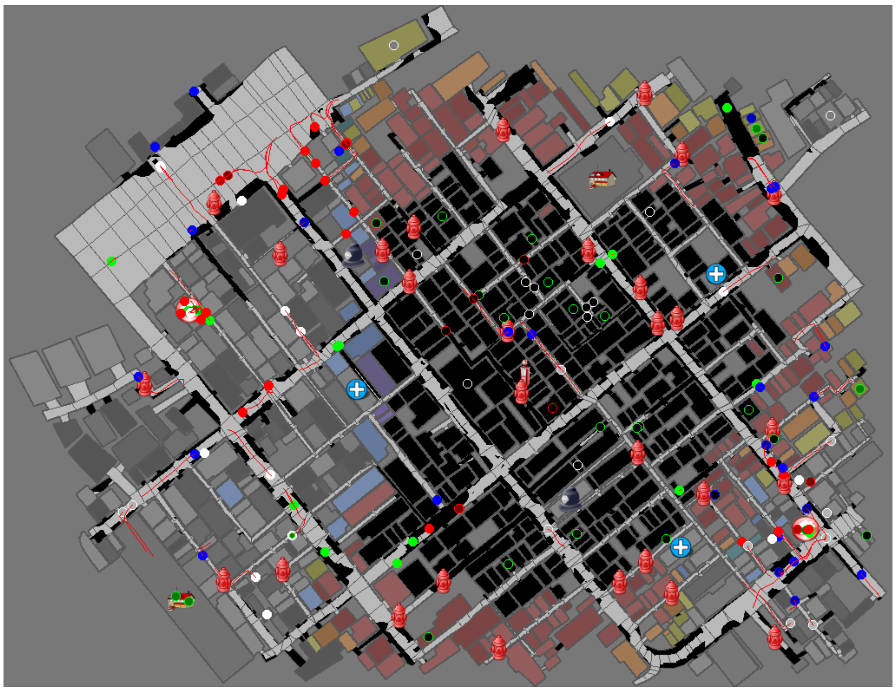


FIGURE 1.3: Hypothetical disaster scenario set in Kobe, Japan. From the Robocup Rescue League Agent Simulation competition [56].

1.1 Research Questions

1. Do theoretical performance guarantees of auction-based task allocation hold up in practice in a multi-robot system? How do factors like inter-robot interference complicate the execution of tasks once an allocation has been made?

2. Does a single auction-based task allocation mechanism perform best across a range of complex environments in which tasks may arrive over time, may require multiple robots to execute simultaneously, or have constraints between them? Do the relative performance rankings of several mechanisms hold across these environments?

3. If certain auction-based task allocation mechanisms perform better in some environments than others, then is it possible to choose a mechanism that performs best for a given environment?

1.2 Contributions

Thesis Statement

This thesis asserts that, as task environments become more complex and varied, no single multi-robot task allocation mechanism will perform best in all task environments in which it is employed.

Research Contributions

This thesis makes four primary contributions:

1. **An experimental software framework** (Chapter 3)
I have developed the MRTeAm software framework[1] to conduct experimental investigation of market-based task allocation mechanisms for multi-robot systems. MRTeAm is a collection of software agents that implement task allocation and execution behaviours for a team of mobile robots. It is used to conduct experiments on both physical and simulated robots with minimal modification to the behaviour between the two. MRTeAm has been deployed on physical robot hardware and in simulations on a massively parallel compute cluster at the University of Liverpool. As part of MRTeAm, I have developed the ROS Master Bridge, an extension to the communication infrastructure of ROS (the Robot Operating System [96], on which MRTeAm is based) that enables inter-robot communication.

2. **Rich Performance metrics** (Chapter 4) Metrics often used to measure the performance of a multi-robot routing mission are the distance travelled by the robot team over the course of a mission and the time taken to reach task locations. Sometimes overlooked is the cost of computing an allocation itself, an important

[1] `http://github.com/nitsuga/mrta`

factor when considering the scalability of missions and teams. Inter-robot interference due to the need to avoid collisions can hamper robots execution of a mission and confound predictions of performance based on allocations alone. The time robots spend idle while team mates execute tasks is also a measure of inefficiency of the team. This thesis defines a set of metrics that measure these factors and tell a much richer story about the performance of a mission than commonly used metrics can.

3. **An empirical investigation of task environments** (Chapter 5–7) This thesis presents a set of multi-robot routing experiments conducted in a series of increasingly complex task environments, with the aim of discovering how theoretical guarantees of mission performance based on allocations alone are borne out in practice. The experiments are conducted on a team of physical autonomous mobile robots when possible, and in high fidelity simulations otherwise. Experimental results reveal how factors like inter-robot interference and the need to re-plan during task execution can complicate expectations of performance that are based on the quality of allocations alone.

4. **A method of mechanism selection** (Chapter 8)
I have developed a data-driven method of selecting a task allocation mechanism from among several options based on reading the arrangements of tasks and robots at the beginning of a mission. The method was inspired by the results of the empirical investigation mentioned above that suggest that, under certain conditions, it may be possible to employ a low cost task allocation mechanism that achieves performance that is competitive with or better than a more expensive alternative. Experiments show that, under certain conditions, selecting a task allocation mechanism using this method can significantly improve the performance of a robot team executing its mission compared to using any single mechanism in the portfolio.

1.3 Thesis Outline

This thesis is structured as follows. Chapter 2 reviews the background in which multi-robot task allocations problems allocation problems are set, states the MRTA formally, and discusses related work that attempts to address it. Chapter 3 details the MRTeAm software framework that has been developed to carry out experimental work for this research. Chapter 4 describes the design of experiments that follow, including the task allocation mechanisms that are evaluated and metrics that measure the performance of a mission.

Chapters 5 – 7 present the results of performance evaluation experiments in a series of increasingly complex environments. Chapter 5 evaluates the comparative performance of several mechanisms in a simple environment in which tasks are allocated in a single phase at the beginning of an experiment, can be executed by single robots, and can be executed in any arbitrary order. Chapter 6 compares this task environment against one in which tasks arrive over time in a fixed schedule. Chapter 7 investigates mechanism performance in yet more complicated environments in which tasks that must be executed by multiple robots at once and may have precedence ordering constraints between them.

Chapter 8 presents the titular contribution of this thesis, a method of selecting a mechanism from a portfolio of options that uses spatial features of tasks and robot locations as features to characterise a mission. This chapter also presents the results of employing the method in the task environment investigated in Chapter 5 and shows that, under some starting conditions, it can lead to a significant increase in performance compared to employing any one single mechanism from the portfolio alone.

Chapter 9 discusses the path of investigation that led to developing the selection method, suggests lines of inquiry for future work, and concludes.

Publications

Portions of the work presented in this thesis have been published in the following:

1. Eric Schneider, Ofear Balas, A Tuna Ozgelen, Elizabeth I Sklar, and Simon Parsons, *An empirical evaluation of auction-based task allocation in multi-robot teams*, Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 1443–1444

2. Eric Schneider, Elizabeth I. Sklar, M. Q. Azhar, Simon Parsons, and Karl Tuyls, *Towards a methodology for describing the relationship between simulation and reality*, Proceedings of the European Conference on Artificial Life (ECAL), MIT Press, 2015, pp. 562–569

3. Eric Schneider, Elizabeth I Sklar, Simon Parsons, and A Tuna Özgelen, *Auction-based task allocation for multi-robot teams in dynamic environments*, Towards Autonomous Robotic Systems: 16th Annual Conference, TAROS 2015 (Clare Dixon and Karl Tuyls, eds.), Springer International Publishing, 2015, pp. 246–257

4. Eric Schneider, Elizabeth I Sklar, and Simon Parsons, *Mechanism selection for multi-robot task allocation*, Towards Autonomous Robotic Systems: 18th Annual Conference, TAROS 2017 (Yang Gao, Saber Fallah, Yaochu Jin, and Constantina Lekakou, eds.), Springer International Publishing, 2017, pp. 421–435

5. Eric Schneider, Elizabeth I Sklar, and Simon Parsons, *Mechanism selection for multi-robot task allocation*, Towards Autonomous Robotic Systems: 18th Annual Conference, TAROS 2017 (Yang Gao, Saber Fallah, Yaochu Jin, and Constantina Lekakou, eds.), Springer International Publishing, 2017, pp. 421–435

Chapter 2

Background and Related Work

This chapter describes the multi-robot task allocation (MRTA) problem and gives an overview of related work. Section 2.1 first discusses multi-robot systems (MRSs) and multi-robot coordination, and where the MRTA problem fits within the landscape of research in this domain. I then state the MRTA problem and introduce definitions and notation that will be used in further discussion. A range of approaches to MRTA is examined that run along an axis from centralised to distributed methods (Section 2.2). Among distributed methods, *market-based methods* of MRTA (Section 2.2.6) are discussed, focusing on *auctions* (Section 2.3). This is followed by a discussion and taxonomy of various kinds of environments in which multi-robot missions are set (Section 2.4).

2.1 Multi-Robot Coordination

The multi-robot task allocation problem fits within a broader category of research in multi-agent systems (MASs) and multi-agent coordination. An *agent* is "a computer system situated in some environment that is capable of autonomous action in this environment to meet its design objectives," [131] or "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators" [100]. In common are the ideas of an autonomous decision-making system situated in some environment that can perceive aspects of the environment and that can act upon the environment. A *multi-agent system* is a collection of agents, perceiving and acting within the same environment, which also includes the other agents in the system. A MAS has potential advantages over single-agent systems intentionally designed to perform some task, in that workloads can be distributed among multiple agents and executed in parallel. However, as discussed in the following sections, this potential parallelism comes with an added potential cost of computation, communication, and complexity.

A *multi-robot system* (MRS) is a type of multi-agent system in which the environment has a *spatial* property. MRSs are distinguished from MASs in that the agents (robots) in a MRS are embodied and can move in a (possibly simulated) physical environment and must interact physically, for example, by locomotion or manipulation [137]. Barnes and Gray (1991) [5] define multi-robot coordination as "joint collaborative behaviour that is directed toward some goal in which there is a common interest or reward". Cao et al. (1997) [16] give a utility-based definition of multi-robot cooperation: "Given some task specified by a designer, a multiple-robot system displays cooperative behaviour if, due to some underlying mechanism (i.e., the 'mechanism of cooperation'), there is an increase in the total utility of the system".

As a review of the complete body of MRS research is outside of the scope of this thesis, the following discussion is restricted to *mobile* robots and does not discuss robotic manipulators and related tasks and planning; human-robot interaction; or competitive (i.e. non-cooperative) systems such as those investigated by Dias and Stentz (2002) [23].

2.1.1 Taxonomies for Multi-Robot Coordination

Several schemes have been proposed to classify the large body of research in multi-robot coordination, each defining categories (or "axes" or "dimensions") of coordination along which a given MRS can be situated. Dudek et al. (1996) [25] focus on the intentional, "task-oriented behaviour" and implementation of a MRS and propose categories for aspects of team architecture such as types of inter-robot communication, team size and composition, and processing ability. Cao et al. (1997) [16] propose categories for team architecture, but also for the "origins"—the motivations and mechanisms—of cooperation, including emergent (e.g., biologically inspired) behaviour; for MRSs that learn behaviours and control parameters; and for different modes of planning for "geometric" (i.e. spatial) tasks. Farinelli et al. (2004) [29] propose broad categories that attempt to decouple abstract properties of multi-robot coordination from categories that relate to team architecture. Parker (2008) [92], covering ideas similar to Farinelli et al., defines three axes that describe how strongly connected and interrelated the goal-seeking behaviours of team members are. Yan et al. (2013) [137] complement these taxonomies, extending Cao's discussion of resource conflicts and including ways that coordination methods can adapt dynamically to changes in the environment.

The following section discusses five of the categories these taxonomies have in common and examples of MRS research that can be found in each. These categories are not fully independent. For example, a strongly-centralised method of coordination might requires some form of explicit communication.

A. Types of Coordination

Parker [92] defines three axes that define a space of the multi-robot coordination domain: robots' awareness of each other; whether or not they share goals (as in e.g., box-pushing), and whether or not the separate goals of different robots have interrelated utility. Farinelli et al. [29] propose a single axis of coordination, covering similar ideas, that runs from "unaware" to "strong coordination". Coordination is governed by the presence (or absence) of a *protocol*, a set of rules that robots must follow to interact with each other. This protocol is used in a process of *deliberation*, or negotiation, in which information is communicated between team members.

In an *unaware* MRS, robots have no knowledge of each other's presence or state. Although explicit coordination or communication does not take place, global, goal-directed behaviour can emerge from the goal-directed behaviours of the individual members. Examples of unaware systems can be found in biologically-inspired designs [18, 67]. With no coordination protocol to follow or sates of other robots to represent and reason about, such systems can scale to large numbers. The goals in an unaware MRS are often simple, such as area coverage or foraging [18]. In a MRS that is *aware but not coordinated*, robots have the ability to perceive and react to their teammates, but lack a coordinating protocol. Reactive behaviours might be repulsive or dispersive [6, 42], where robots seek to avoid interference with one another. These types of systems have been applied to goals that are similar to those of "unaware" systems, including foraging and exploration or area coverage [6]. These types of systems are also capable of being scaled to large numbers of robots without running into communication or computational bottlenecks [61, 99].

Robots in a *weakly coordinated* MRS are aware of each other but still lack an explicit coordinating protocol. In contrast with "no coordination" MRSs, robots may use perception of other robots and simple rules to achieve tasks in a coordinated way. Balch and Arkin (1998) evaluate methods for formation-holding tasks, in which ground-based robots seek to maintain relative distances from each other without explicitly communicating [4]. In a *strongly coordinated* MRS, there exists a coordinating protocol and a mechanism for deliberation. The GOFER architecture of Caloud et al. (1990) [15] coordinates exploration missions using a central planning system that allocates allocates tasks to robots. Gerkey's MURDOCH [34] and Dias's Traderbots [22] architectures feature coordinating protocols that allocate tasks to robots through market-based methods of negotiation and bargaining (Section 2.2.6).

B. Types of Organization

Among MRSs that are strongly coordinated, Farinelli et al. [29] define the "organization level" of a coordinating protocol that distinguishes between centralised and distributed approaches. In a *strongly centralised* system, a single agent in the system acts as a leader that guides the decision making for the whole team. Veloso and Stone [127] describe an architecture for robot soccer in which player agents depend on a global vision system to track the locations of players and the ball. In a *distributed* system, agents are fully autonomous and make their own decisions about how to implement the coordinating protocol. In Parker's ALLIANCE architecture [91], autonomous robots with individual goal-oriented behaviours coordinate by direct interaction without any overseeing agent.

C. Communication

Communication between robots may occur peer-to-peer or within a limited broadcast range. In Wang's "Sign-board" architecture [130] each robot maintains an internal database of information that is broadcast to team mates within communication range. In Konolige et al.'s Centibots architecture [60, 61], leaders of "exploration clusters" aggregate map data from robots in local neighbourhood for an exploration task. Communication may occur globally as in [127]. Blackboard systems [20] maintain a global database that team members use to share information [14]. Publish/subscribe messaging models [34, 96] transmit information globally, but limit bandwidth use by sending messages only to robots interested in certain topics.

D. Team Composition

Homogeneous teams are composed of robots with identical capabilities. In *heterogeneous* [13, 59, 66, 121] teams, robots may have different physical configurations and capabilities [91] or implement different behaviours or roles [3, 33, 119]. Another property of team composition is the team size. Yan et al. [136] present an framework for determining optimal team sizes for exploration missions.

E. Resource Conflict and Resolution

A *resource conflict* occurs when multiple robots request access to the same indivisible resource [16]. Resources might include communication bandwidth [101, 138], space (including access to the same physical object), or sub-goals of a mission. Conflicts over spatial resources require some method of reactive collision avoidance [30, 48, 117, 126], collision-free path planning [9, 129] or explicit negotiation to determine right-of-way [2, 51]. Conflicts over tasks (or sub-goals) that make up a mission can be resolved by a protocol that determines how to assign tasks to robots, and this is discussed in the next section.

2.2 Multi-Robot Task Allocation (MRTA)

In a multi-robot system that is cooperative and intentionally coordinated, multi-robot task allocation is the problem of assigning each of a set of discrete tasks that make up a *mission* to robots in the team, typically in a way that optimises some performance objective (minimising an overall measure of cost or of maximising an overall measure of utility). Several formulations of the MRTA problem have been proposed. In settings where each robot in a team can perform one task at a time and each task can be performed by a single robot, multi-robot task allocation can seen as equivalent to the Optimal Assignment Problem (OAP) from Operations Research and formulated as a linear program [36]. Gerkey and Mataríc also formulate it as a set covering problem [36]. A flexible definition that is adopted here is given by Kalra et al. (2005) [53] and Zlot &

Stentz (2006) [142], itself adapted from a definition of multi-agent task allocation given by Andersson and Sandholm (2000) [1]:

Definition 1. Given a set of robots R, let $\mathcal{R} = 2^R$ be the set of all robot subteams. An **allocation** of a set T of tasks to R is a function, $A : T \rightarrow \mathcal{R}$, mapping each task to a subset of robots responsible for completing it. Equivalently, $\mathcal{R}^T$ is the set of all allocations of the tasks T to the team of robots R. Let $T_r(A),\ \ r \in \mathcal{R}$ be the set of tasks allocated to subteam r in a given allocation A.

Definition 2. The Multi-robot Task Allocation Problem: Given a set of tasks T, a set of robots R, and a private cost function for each subset of robots $r \in \mathcal{R}$ specifying the cost of performing each subset of tasks $c_r : 2^T \rightarrow \mathbb{R}^+ \cup \{\infty\}$, find the allocation $A* \in \mathcal{R}^T$ that minimizes a global objective function $C : \mathcal{R}^T \rightarrow \mathbb{R}^+ \cup \{\infty\}$.

This definition of the MRTA problem considers all possible allocations of subsets (or *bundles*) of tasks to all possible subteams of robots. A feasible allocation is any partition of T that maps subsets of tasks to subteams of robots. Finding an optimal allocation, however, is NP-hard [1, 70].

2.2.1 Cost Functions and Solution Quality

Definition 2 describes two functions for computing costs. A private cost function, c_r, computes the cost incurred by an individual robot or subteam when executing a single task or a subset of tasks. A global cost function, C, which a task allocation method attempts to optimise, is often related to the private cost function, but may be more than just a simple sum.

Valuations of tasks by individual robots or subteams of robots are sometimes expressed in terms of utility rather than cost, in which case an MRTA solution seeks to maximise the global utility of an allocation. In a mapping mission, a location along an exploration frontier that promises high information gain (revealing more of the map) might have a higher utility than a neighbouring location [114]. Kalra et al. [53] give an example valuation that is a combination of cost and utility in an exploration mission, where completing a task incurs a positive utility but each unit of distance travelled moving toward it incurs a negative cost. Chevaleyre et al. (2006) [19] point out that a task can be viewed as a kind of resource to which robots assign negative utility (i.e., cost). In multi-robot routing missions (Section 2.2.3), a natural measure of cost is the distance a robot must travel along a path from its current location to a task location. A cost based on path distance can be thought of as a proxy for a robot's consumption of fuel or battery power. A cost function might also be based on the estimated time to travel to a task location, which may be complicated by congestion, localisation uncertainty, or other factors that can affect navigation.

Two commonly used global cost functions are MiniSum and MiniMax [53, 70, 142]. The MiniSum objective minimises the sum of the costs incurred by individual robots or subteams in an allocation, which might represent the total fuel or battery energy consumed by a team as robots travel to task locations of a routing mission:

$$C(A) = \sum_{r \in \mathcal{R}} c_r(T_r(A)) \tag{2.1}$$

The MiniMax objective, also known as the *makespan*, minimises the maximum cost incurred by an individual robot or subteam in an allocation. It may be desirable to minimise the maximum cost in a search and rescue mission, where the goal is to reach a victim furthest away from aid in the least amount of time:

$$C(A) = \max_{r \in \mathcal{R}} c_r(T_r(A)) \tag{2.2}$$

A third objective, MiniAvg, minimises the average cost of an individual robot or subteam in an allocation [70]. This might be desirable in a search and rescue mission, where the health condition of several victims deteriorates over time [122] or in a patrolling mission where frequent, regular coverage of a patrol path is required [50].

$$C(A) = \frac{1}{m} \sum_{r \in \mathcal{R}} c_r(T_r(A)) \tag{2.3}$$

A allocation may be optimal or approximately optimal. A **ρ-approximate** solution for a multi-robot task allocation problem is no greater than ρ times the cost of an optimal solution to a minimisation problem or $\frac{1}{\rho}$ times the cost for a maximisation problem [53]. For an online problem in which tasks appear dynamically over time, a **ρ-competitive** solution is no greater than ρ or $\frac{1}{\rho}$ times the cost of an optimal *offline* solution of the same problem—accounting for all tasks that eventually arrive—for minimisation and maximisation problems, respectively.

2.2.2 Missions and Tasks

A *mission* is a high-level goal such as "search an area for people in distress" or "deliver a batch of goods to these customers". A "task" is a subgoal of a mission, such as "travel to a location (x, y) and take a picture" or "patrol a route along these waypoints". Some authors define a task as a *role*, or set of behaviours, such as an attacker or defender in robot soccer [32, 33, 119, 125] or individual robot actions [81].

A task specification language is often used to decompose a high-level mission goal into subgoals that can be allocated. Noreils [87] describes how subgoals can be decomposed automatically from a mission-level goal by a general-purpose planner. Simmons and Apfelbaum [113] and Brummit and Stentz [13] define task specification languages that system designers can use to represent tasks explicitly.

In the simplest cases, tasks are *atomic*, in that they can not be further decomposed, and *independent*, in that there are no dependencies or constraints between tasks (e.g., precedence ordering). In more complex missions, tasks may be organised into groups [87], trees [113] or other structures.

2.2.3 Multi-Robot Routing

A canonical example of multi-robot task allocation is found in multi-robot routing problems [58, 70], in which a team of mobile robots must visit a set of task locations to perform some work such as delivery or collection. The routes, or paths, from robots to task locations should optimise criteria like travel distance or time. Movement to a location itself is considered a task and the work performed there is abstracted. In missions with atomic and independent tasks, a multi-robot routing problem is similar to a multiple depot multiple Travelling Salesperson Problem (mTSP) [7] or Vehicle Routing Problem (VRP) [72], where salespeople or vehicles need not return to their depots [70].

A definition of the multi-robot routing problem adopted here is given by Lagoudakis et al. [70], and gives Definition 2 a spatial interpretation:

Definition 3. The Multi-robot Routing Problem: Given a set of robots, R, and their locations, a set of tasks, T, and their locations, and a function $c(i, j) \to \mathbb{R}^+ \cup \{\infty\}$, $i, j \in R \cup T$, which denotes the cost of moving between target (task) or robot locations i and j, find the allocation $A* \in \mathcal{R}^T$ that minimizes a global objective function $C : \mathcal{R}^T \to \mathbb{R}^+ \cup \{\infty\}$.

The cost c, or *path cost*, is defined here for a single robot moving from its current location to a task location or between two task locations. A path cost between locations i and j may be based on Euclidean distance or Manhattan distance, but in a physical setting paths are often planned around obstacles in the environment (e.g., by Dijkstra's Algorithm [24] or A* [45]). The path cost may also be based on an estimated time to travel between locations i and j. Definition 3 assumes costs are symmetric, $c(i, j) = c(j, i)$, are the same for all robots, and satisfy the triangle inequality. A path cost may be infinite if it is impossible to find a path between locations i and j. The global objective function C typically computes, as in Definition 2, the sum of path costs over all tasks assigned to robot sub-teams in a given allocation.

2.2.4 Centralised MRTA

Centralised approaches to MRTA rely on a single "leader" agent to plan, allocate and coordinate the executions of tasks for the entire team. With global information about the environment and robot states, it is straightforward (but not necessarily efficient) for a centralised coordinating agent to allocate tasks optimally. A centralised path planner, for example, can calculate collision-free paths that optimise the MiniSum or MiniMax objective for a multi-robot routing mission by solving a multiple Travelling Salesperson Problem or Vehicle Routing Problem. However, as Kalra et al. [53] note, the complexity of centralised coordination grows exponentially in the number of agents and tasks, suffers from a single point of failure, and may incur high communication costs. Although there are efforts to reduce the search spaces of centralised solvers [129], centralised approaches (e.g., GRAMMPS [13]) generally do not scale as well as distributed approaches and may only be appropriate for smaller teams or missions.

2.2.5 Fully Distributed MRTA

In fully distributed approaches to MRTA, robots operate only on local information. Methods, such as threshold-based task allocation [64], have low computation requirements, require little or no communication, can be robust to robot failure, and may be scalable to large team and mission sizes. However, such fully distributed approaches provide little explicit coordination and may produce suboptimal results without careful or computationally expensive engineering or training [11]. Fully distributed approaches to multi-robot task allocation may be most appropriate for large teams that are faced with relatively simple tasks like foraging [65].

2.2.6 Market-based MRTA

Market-based multi-robot task allocation is somewhat of a hybrid of centralised and distributed approaches. In market-based MRTA, self-interested agents seek to maximise personal gain by trading resources in a "virtual economy" [53] of tasks. Market-based approaches combine some of the advantages of centralised approaches, such as intentional coordination of a global objective, and distributed approaches, such as distribution of of computation and robustness to failure.

The following are some general properties of a market-based approach to MRTA defined by Kalra et al. [53]:

- The team is given an objective that can be decomposed into sub-components achievable by individuals or sub-teams. The team has access to a limited set of resources with which to meet this objective.
- A global objective function quantifies the system designer's preferences over all possible solutions. This function may be complex and take into account multiple criteria. For example, it might be desirable to minimize time and energy spent moving (cost), while maximizing utility (completion of high priority tasks).
- An individual utility function (or cost function) specified for each robot quantifies that robot's preferences for its individual resource usage and contributions towards the team objective given its current state. Evaluating this function cannot require global or perfect information about the state of the team or team objective. Subteam preferences can also be quantified through a combination of individual utilities (or costs).
- A mapping is defined between the team objective function and individual and subteam utilities (or costs). This mapping addresses how the individual production and consumption of resources and individuals' advancement of the team objective affect the overall solution.
- Resources and individual or sub-team objectives can be redistributed using a mechanism such as an auction. In a well-designed mechanism, maximizing the

mechanism-controlling agent's utility (or minimizing its cost) results in improving the team objective function value.

An early example of market-based multiagent task allocation is Smith's Contract Net Protocol [116]. A contract net is a collection of agents that negotiate with each other to distribute tasks by advertising and honouring contracts for work. Agents assume either the role of *manager* or *contractor*. Managers announce the availability of tasks to contractors and solicit *bids* from them. Eligible contractors capable of executing the tasks compute and submit bids to managers. If a bid is deemed "satisfactory", the agent that submitted the bid is awarded the task that it bid on. The protocol defines a model of communication for task allocation but makes no assumptions about the meaning of bid values or how bids are determined to be satisfactory.

Wellman and Wurman [132] introduce ideas from economics to multiagent task allocation. Task allocation is modelled as an *auction mechanism*, where agents can compute and compare internal valuations, or *prices*, for tasks. A price compactly summarizes and encodes the utility or cost of a possibly complex internal process, such as a plan, for executing a task. A valuation encoded as a price also makes it easier to employ established mechanisms like auctions to multiagent task allocation.

Market-based task allocation for multi-robot systems was first proposed by Stentz and Dias [118] and later extended in Dias's Traderbots architecture [22]. In what they describe as a "free market" system, robot agents model both the utilities and costs of executing tasks, and seek to maximize their individual profits by trading tasks with each other using an extension of the Contract Net protocol [116].

2.3 Auctions

Auctions are the most common type of market-based approach to MRTA problems [53]. An MRTA *auction mechanism* is both a process that determines how to allocate tasks based on robots' private valuations and a protocol that specifies how tasks and valuations are encoded and communicated.

Auctions are conducted over a number of *rounds*, which typically have three phases:

- **Announcement**: An *auctioneer* agent advertises one or more unallocated tasks from a pool to eligible robots.
- **Bid Computation and Submission**: Each robot calculates a private valuation (a utility or cost) for the task(s) announced and submits bid(s) to the auctioneer.
- **Winner Determination and Awarding** (or *Clearing*): The auctioneer aggregates all bids received, calculates one or more winners, and awards the winners with the tasks. Awarded tasks are removed from the auctioneer's task pool.

Auction rounds are conducted until all tasks have been allocated and the task pool is empty.

Parsons et al. [93] classify auction mechanisms using Shoham's taxonomy [112] as well as their own parametric model. By this classification, MRTA auctions are *one-sided* in that robots represent multiple "buyers" of tasks but there is only a single "seller" agent (the auctioneer). In a cooperative multi-robot system, auctions are typically *sealed bid* in that robots reveal bids to the auctioneer but not to each other. Auctions may be *single-item*, where one task is announced and awarded per round or *multi-item*, where bundles of tasks are announced and awarded. The "price" that represents a robot's valuation of a task is typically abstract; that is, with some exceptions [22], a robot does not exchange a resource with the auctioneer in return for the acquisition of a task.

Gerkey and Matarić introduced auctions for MRTA problems in their MURDOCH architecture [35] as an extension of the Contract Net Protocol. MURDOCH uses a central auctioneer agent and multiple autonomous robot bidding agents with a global publish-subscribe communication model. MURDOCH allocates cooperative tasks like box pushing using a single-item auction mechanism and utility functions based on metrics like how well positioned a robot is to push a box. In [35], the quality of task execution is measured, but not the global quality of the allocation.

Lagoudakis et al. formalise auction-based MRTA for multi-robot exploration and routing missions [69, 70]. They show that an optimal solution to a multi-robot routing problem is NP-hard and, as an approximate solution, develop bidding rules for the MiniSum, MiniMax, and MiniAvg objectives. They prove that their auction-based bidding rules result in solutions whose worst case performance is bounded by a constant multiple (ρ) of an optimal solution (for MiniSum) or a multiple that is linear in the number of tasks (for MiniMax) or size of the team (for MiniAvg).

2.3.1 Auction Mechanisms

A MRTA auction mechanism typically offers an approximate solution to an optimisation problem, so mechanism designers make trade-offs between the quality of the solution and the costs of computing and communicating it. Wellman and Wurman [132] define an abstract *mechanism space* that considers properties of the messages communicated by an auction protocol. Parsons et al. [93] give a parametric model of mechanism space that considers the eligibility of agents to participate in an auction, rules for accepting or rejecting bids, and functions, including winner determination, that govern how resources are matched with agents. They also provide a model of the processes of an auction mechanism, similar to the phases of an auction round listed in Section 2.3. More concretely, some parameters available to mechanism designers are:

1. *Announcement*: An auctioneer must choose which tasks are announced to which robots from a pool of unallocated tasks. Tasks may be announced singly or in bundles. A sequence of tasks announced over a number of rounds may be governed by ordering or other constraints. Announcements may be broadcast to the entire team or multicast certain groups of robots.

2. *Eligibility*: In a heterogeneous robot team, not all robots might be capable of executing all tasks that are announced. A mechanism might ensure a robot's eligibility to participate in an auction, as in Smith's Contract Net Protocol [116].

3. *Bid selection, computation, and submission*: A robot (or subteam) can choose to bid on all tasks announced in a bundle or a subset of them.

   A robot's (or subteam's) private valuation of a task that it reports in a bid might be accurate or estimated from partial information. The path cost of a routing task, for example, might be subject to a robot's uncertainty about its location or obstacles in the environment.

   A robot (or subteam) may consider the cost of *task insertion*. A bid for a routing task, for example, might report the *cumulative cost* of a tour of task locations that have already been awarded plus the cost of adding a new task. Alternatively, a bid might report the *marginal cost* of adding a new task location to a tour of task locations that have already been awarded [70].

4. *Winner Determination and Awarding*: An auctioneer aggregates all of the bids that are submitted and may choose winners according to an objective like MiniSum or MiniMax. Ties may be broken randomly or by some principled method. More than one robot (or subteam) may win and be awarded tasks in a single round.

Different choices made within this mechanism designs space can lead to widely varying costs both in terms of computation and communication of the auction itself and in terms of mission performance once an auction has been conducted. Phelps et al. [95] investigate evolutionary approaches to the design of auction mechanisms for multiagent systems. In a multi-robot exploration setting, Tovey et al. [122] systematically generate bidding rules for the MiniSum, MiniMax, and MiniAvg objectives and show experimentally that the rules result in team performance that is "good" for their respective objectives. In this space of mechanism designs, auctions for MRTA can be broadly classified as *single-item* or *multi-item*.

Single-item Auctions

In a single-item auction, one unallocated task is announced and awarded per round, and rounds are conducted until all of the tasks of a mission have been allocated. Winner determination involves finding the minimal-valued bid submitted. The number of rounds is equal to the number of tasks in the mission and communication costs are linear in the number of tasks and size of the team. As rounds progress, robots that have already won tasks might evaluate the cumulative or marginal cost of acquiring a new task, and so their bid computation costs can increase.

In a multi-robot exploration mission, Berhault et al. [8] show that single-item auctions can result in suboptimal allocations because they do not take into account "synergies" of related tasks. If a performance objective involves minimising a cost, two tasks are said to exhibit positive synergy if their combined value for the bidder is smaller than the sum of their individual values. In a routing mission, this simply means that tasks are located close to each other. Because task synergies are ignored in single-item auctions, the order in which tasks are auctioned can have a great impact on the quality of the solution [46].

Multi-item Auctions

In a multi-item auction, multiple tasks are announced and awarded per auction round. Multi-item auctions can be further classified into *combinatorial*, *parallel*, and *sequential single-item* mechanisms.

In a combinatorial auction, all unallocated tasks are announced to the robot team and awarded in a single round (although multi-round variations exist [8]). In a combinatorial auction, robots compute and submit bids for every possible combination (bundle) of tasks. During the winner determination phase, synergies between tasks can be discovered and an optimal solution computed for a given objective. However, the optimal winner determination problem for combinatorial auctions has no known polynomial solution and is NP-complete [8, 103]. Bid computation and communication costs for robots are also exponential in the number of tasks. Although heuristics have been proposed to find approximately optimal solutions to winner determination, such as pruning the number of bundles bid upon or evaluated [8] or building incremental solutions [103], combinatorial auctions for multi-robot MRTA can still scale too quickly to be practical. Combinatorial auctions have been employed for the RoboCup Rescue domain [77, 85] and multi-robot exploration missions [8].

In a parallel auction [58], all unallocated tasks are announced to the robot team and awarded in a single round. Unlike a combinatorial auction, robots submit bids for each task independently, ignoring inter-task synergies. Winner determination, likewise, ignores task synergies and simply awards each task to the lowest (or highest) bidding robot. A solution to the winner determination problem can be found in time linear in the number of tasks and size of the team. Bid computation and communication are also linear since there is no computation of task insertion (as in single-item auctions). While computation and communication costs of parallel auctions are relatively low, performance of a parallel auction allocation can be “arbitrarily bad” as Koenig et al. [58] demonstrate.

Sequential single-item (SSI) auctions were developed by Koenig et al. [58], extending on earlier work with Lagoudakis et al. [70] on auctions for multi-robot routing. In a SSI auction, all unallocated tasks are announced to the robot team. Each robot computes the cost of each task but submits a bid only for the minimum cost task computed. The robot that submits the lowest bid is awarded the task that it bid on, and auction rounds continue until all tasks have been allocated.

A SSI auction takes some synergies between tasks into account, but not all. Koenig et al. give an example where an SSI auction produces an allocation for the MINISUM objective that is 1.33 times the cost of an optimal solution and equivalent to an allocation produced by a parallel auction. Despite this, allocations produced by SSI have been proved to have a theoretical worst case performance of 2 times an optimal solution for the MINISUM objective [58].

Extensions of SSI have been developed to improve its solution quality by identifying inter-task synergies missed by standard SSI auctions. With *lookaheads*, robots bid in hypothetical multiple auction rounds but award a single task at a time [140]. *Bundle-bids* have robots bid on fixed-size bundles of tasks per round and award the bundles [140]. *Rollouts* have robots compute the cost of the closest task plus the cost of visiting all unallocated task locations [140]. With *regret clearing* [139], robots submit bids for all unallocated tasks and the auctioneer determines a winner by maximising the difference between the lowest and second lowest bids submitted. These extensions are shown to improve solution quality but raise the cost of bid computation and communication or winner determination. *Sequential single-cluster auctions* [46] attempt to capture inter-task synergies by first clustering tasks geographically into bundles, and then auctioning the bundles.

2.4 Task Environments

Several researchers have devised taxonomies to classify the body of research into multi-robot task allocation. A scheme by Gerkey and Mataríc [36] draws major distinctions between single-task (**ST**) and multi-task (**MT**) robots, which can perform more than one task at a time; single-robot (**SR**) and multi-robot tasks (**MR**), which require multiple robots to execute; and instantaneous assignment (**IA**) and time-extended assignment (**TA**). Instantaneous assignment means that knowledge about the robot team and tasks is limited, so an allocation mechanism can not plan for tasks that may arrive in the future. With time extended assignment, an allocation mechanism may have knowledge about future tasks, such as a schedule. Landen et al. [71] extend Gerkey and Mataríc's taxonomy to classify settings in which tasks are either independent (**IT**) or have constraints between them (**CT**), such as precedence-ordering, and distinguish static allocation (**SA**) and dynamic allocation (**DA**) from Gerkey and Mataríc's *IA* and *TA* to describe static or dynamic environments themselves rather than an allocation mechanism's knowledge about them. Korsah et al. [62] define *iTax*, a taxonomy for multi-robot tasks that classifies tasks as atomic or compound bundles which may have inter-task dependencies or utilities. Table 2.1 gives the dimensions of the taxonomy proposed by Gerkey et al. with the extensions made by Landen et al.

This thesis focuses on three dimensions of the taxonomy given in Table 2.1: Single-Robot vs. Multi-Robot Tasks (**SR vs. MR**); Independent vs. Constrained Tasks (**IT vs. CT**); and Static Allocation vs. Dynamic Allocation (**SA vs. DA**).

Dimension	Description
ST vs. **MT**	Robots can perform a single task (ST) or multiple tasks (MT) at a time.
SR vs. **MR**	A task requires only a single robot (SR) or multiple robots (MR).
IA vs. **TE**	An instantaneous assignment (IA) has no information to reason about tasks that may arrive in the future; a time-extended (TE) assignment may have a model of how tasks are expected to arrive.
UU vs. **IU**	A task with unrelated utility (UU) has value, by itself, to bidders. Tasks with interrelated utility (IU) may only have value when bundled with other tasks.
IT vs. **CT**	Task may be independent (IT) or constrained (CT) by a dependency like precedence-ordering.
EV vs. **IV**	Allocation is performed by an agent or entity outside of the robot team in an external allocation view (EV). In an internal view (IV), allocation is performed by a team member(s) and can itself be considered a task.
SA vs. **DA**	The pool of tasks, constraints between them and team composition are held static in a static allocation environment (SA). In a dynamic environment (DA), they may vary over time.

Table 2.1: “Axes” or dimensions of a task environment [36, 71]

2.5 Summary

Auctions mechanisms for multi-robot task allocation are implemented in multi-robot systems that are strongly coordinated (i.e. an auction mechanism is a coordinating protocol); centralised, decentralised, or a hybrid; and that use explicit communication to submit bids and awards. The following chapters investigate the performance of different auction mechanisms empirically, with a multi-robot system implemented on physical robots and in high-fidelity simulations. A primary interest is in seeing how theoretical guarantees of solution quality made during task allocation, such as SSI's upper bound of two times the optimal team distance for the MINISUM objective (Section 2.3.1), accurately predict or diverge from observed performance during task *execution*. Of equal interest is understanding how performance varies as mechanisms are employed in a range of *task environments* that can be complicated in ways which might not have been considered by mechanism designs. The following chapter describes the architecture of the multi-robot system used to conduct these experiments.

Chapter 3

The MRTeAm Experimental Framework

This thesis investigates the comparative performance of different market-based task allocation mechanisms when employed for the same multi-robot routing missions. I am especially interested in measuring the empirical performance of a mission by a team of real (or realistically simulated) robots in physical environments, with the noise, uncertainty, and inter-robot interference that can arise in a physical system. To conduct this experimental work, I have developed a software framework named MRTeAm, a combination of "Multi-Robot Task Allocation" and "Teamwork". MRTeAm is an implementation of a multi-robot system that runs on both physical and simulated robots and is designed to investigate research problems in multi-robot task allocation [107–110], multi-robot communication quality [141], and other domains.

MRTeAm is an evolution of the HRTeam ("Human-Robot Teamwork") software framework [115], which itself has been used as a research platform for similar problem domains [26–28]. HRTeam is a multi-robot system based on the Player Project [37], which is no longer actively maintained. In an effort to adopt a more mature infrastructure, MRTeAm is based on the *Robot Operating System* (ROS) [96], a pervasive software infrastructure underlying many single- and multi-robot systems in both research and applied industrial settings. ROS provides many services needed by the kind of mobile multi-robot systems that MRTeAm was designed to work with: drivers for sensing, sensor fusion and motor control; map representation;[1] localisation;[2] path planning[3] and execution, including plan repair and recovery;[4] and, importantly, a standard messaging system[5] for inter-process communication, which allows these services to be distributed across multiple hosts in a flexible way.

MRTeAm builds upon ROS by adding several key components. *Robot Controllers* are agents responsible for the bidding and task execution behaviour of robot team members.

[1]http://wiki.ros.org/map_server
[2]http://wiki.ros.org/amcl
[3]http://wiki.ros.org/global_planner
[4]http://wiki.ros.org/move_base
[5]http://wiki.ros.org/Messages

An *Auctioneer* is a coordinating agent responsible for conducting auctions among the robot controllers and guiding the overall course of an experiment. The Robot Controller and Auctioneer agents implement task allocation and execution for missions which are set in the **SR/MR-IT/CT-SA/DA** task environments described in Chapter 2. The *ROS Master Bridge* is an extension to ROS's messaging system that connects multiple ROS communication hubs ("masters"), enabling multi-robot communication

Section 3.1 explains some of the basic concepts behind ROS and its implementation of navigation services for mobile robots, which are relevant to understanding the results of experiments in later chapters. This review of existing software is followed by sections that describe my own work on MRTeAm. Section 3.2 discusses the auctioneer and robot controller agents that implement task allocation and execution behaviours to accomplish multi-robot routing missions. Section 3.3 discusses the ROS Master Bridge and multi-robot communication. Section 3.5 describes the physical platform on which MRTeAm has been deployed and section 3.6 describes its counterpart in simulations. Section 3.7 discusses the HRTeam framework used to carry out the experiments discussed in Chapter 5.

3.1 The Robot Operating System (ROS)

ROS consists of a set of computation units ("nodes"), which run as separate processes, and a structured communications layer based on message-passing [96], which enables communication between them. A ROS *node* is a process that performs some computation and communicates with other nodes via messages. ROS messages are sent between nodes on *topics* using a publish-subscribe model [34]. A ROS *master* is a special process that provides naming and directory services that keep track of nodes and publishers and subscribers to topics. A ROS *message* is an instance of a typed data structure that nodes use to communicate. The set of all ROS nodes that are connected to a single master comprises a *ROS Computation Graph* (Figure B.1, pg. 138).

3.1.1 The ROS Navigation Stack

The ROS *navigation stack*[6] is a collection of nodes which provide services that enable mobile robots to plan and navigate through physical spaces. Physical space is represented by a two-dimensional, grid-based *cost map*, where the value of each grid cell represents the likelihood of occupancy by an obstacle in that location. Obstacles may be inserted or cleared from the cost map based on readings from a robot's range sensors (e.g., a laser or infrared-based camera). A *localisation* node attempts to find the most likely position of the robot on the map from spatial features detected by the robot's range sensors using a particle filter approach [31]. A *global planner* finds the shortest path between two locations through the cost map using an A* search algorithm [45]. A

[6]http://wiki.ros.org/navigation

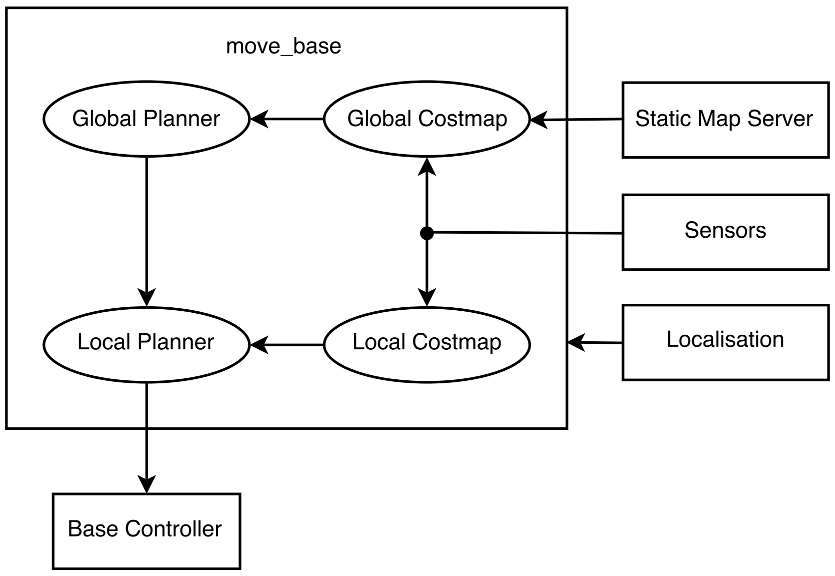


FIGURE 3.1: Components of the ROS navigation stack and their interactions.

path plan produced by the global planner is made up of a series of positions, or *waypoints*, from the robot's current position to the goal. A *local planner* follows a global plan by computing motion trajectories and issuing velocity commands that follow them to a robot's drive motors. Several of these services are coordinated by the `move_base`[7] node (Figure 3.1). Some components of the navigation stack can be visualised for monitoring and debugging (Figure 3.2).

3.2 Agents

Two types of agents carry out missions in MRTeAm experiments. The *Auctioneer* decomposes a mission into discrete tasks and allocates them to *Robot Controllers* using a number of mechanisms.

3.2.1 Auctioneer

The auctioneer is an agent with two main roles. Chiefly, it is responsible for allocating tasks to robots via auctions and other types of mechanisms. The auctioneer sends and receives messages that announce tasks to robots, collects bids from them, determines winners, and awards tasks to robots. The auctioneer is also responsible for the overall coordination of experiments by signalling the transitions of experimental phases (Section 4.5) to a tool that logs results (Section 3.4).

At the beginning of an experiment, the auctioneer loads a pre-defined *scenario*, which specifies a map and a set of tasks. (A scenario is defined in Section 4.3). The auctioneer then places tasks that make up the scenario into a *task pool* of unallocated tasks. Alternatively, the auctioneer can wait for tasks to be introduced by some external source (e.g.,

[7]http://wiki.ros.org/move_base

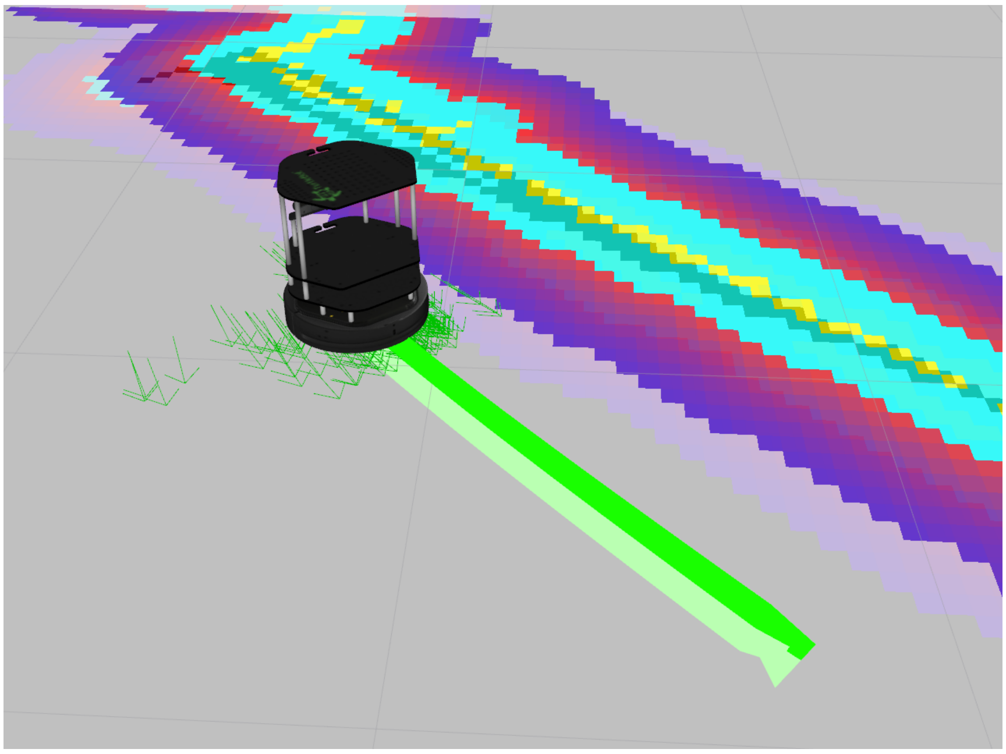


Figure 3.2: Visualisation of the state of navigation stack components running on a simulated robot. A global path plan, shown in green, has a goal location in the lower right. Green arrows show samples of the particle filter used for localisation. Shaded cells of the cost map along the upper right indicate occupancy by an obstacle. Different colours indicate different costs for cells in the cost map. For example, a yellow cell is the most costly as it is the most likely that a robot will crash into a physical obstacle (such as a wall) in that region.

a mission generator). After the task pool is populated, the auctioneer identifies members of the robot team and selects a task allocation mechanism. A mechanism is either chosen manually as a parameter of the experiment, or dynamically based on features of the environment (as in Chapter 8). The auctioneer then advances the experiment to a *deliberation*, or task allocation, phase. An auction-based allocation takes place over a number of rounds, with each round typically having three sub-phases:

1. *Announcement*: One or more tasks from the task pool are announced to the team via ROS messages. The number of tasks to announced depends on the rules of the chosen mechanism.

2. *Bid computation/submission*: Each robot team member computes a bid value (Sections 3.2.2 and 4.4) for one or more of the tasks that were announced and publishes it to the auctioneer via a ROS message(s).

3. *Winner determination*: The auctioneer collects the bids submitted and determines winners (possibly more than one) of task(s) auctioned in the current round according to the rules of the mechanism.

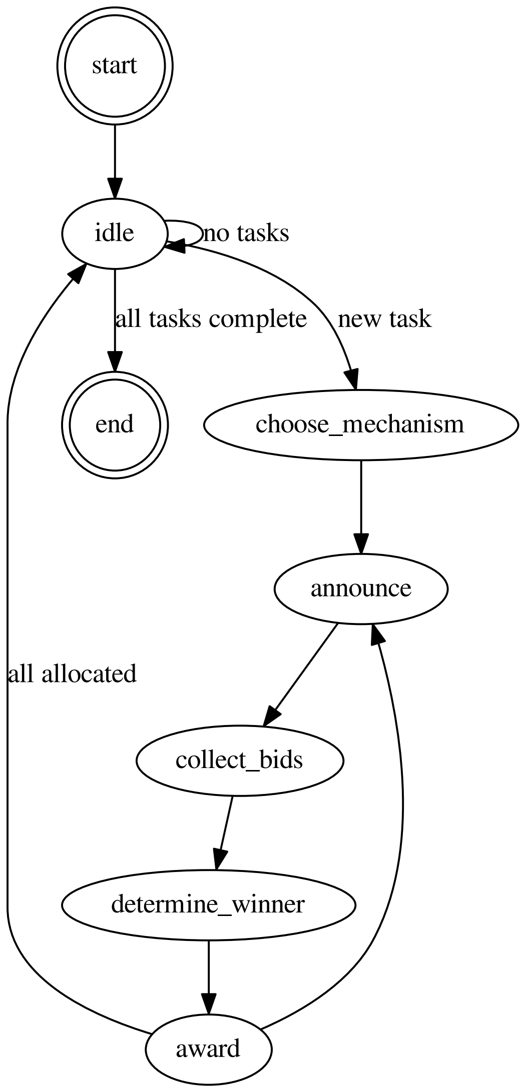


FIGURE 3.3: A simplified version of the auctioneer's state machine.

4. *Award*: The auctioneer awards each winning robot by publishing a ROS message. After a task has been awarded, it is removed from the pool (cleared).

Task allocation proceeds until the task pool is empty. The auctioneer then marks the end of a *deliberation* phase of the experiment and the beginning of an *execution* phase in which robots traval to their task locations. A simplfied version of the state machine that drives the auctioneer's behaviour is shown in Figure 3.3. A diagram of the communication that takes place between the auctioneer and robot controllers is shown in Figure 3.4.

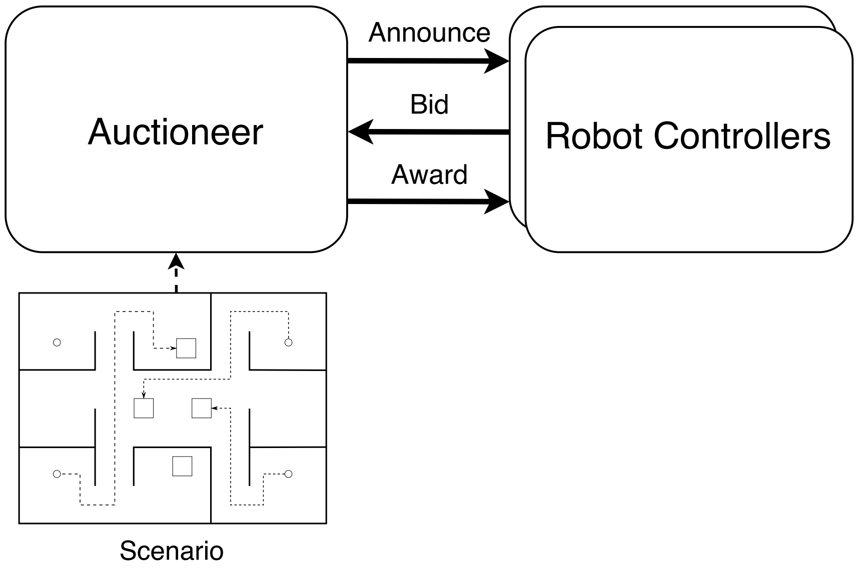


Figure 3.4: Auctioneer communication with robot controllers.

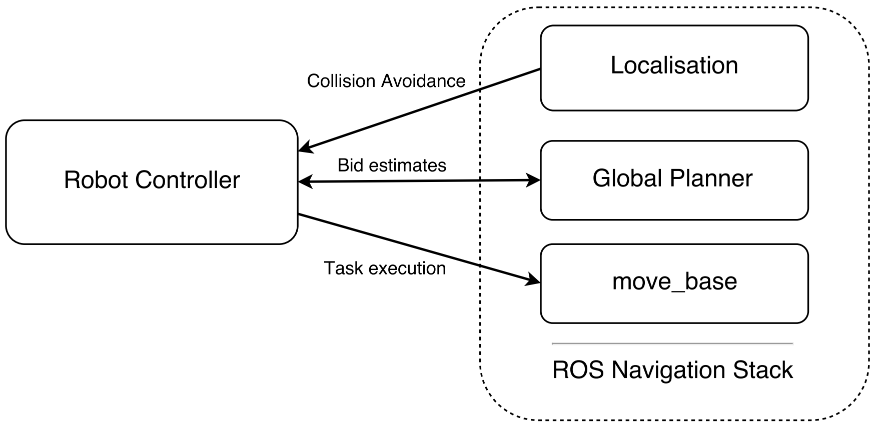


Figure 3.5: Communication between a robot controller and the ROS navigation stack.

The auctioneer implements a number of allocation mechanisms (discussed in Section 4.4), but is extensible so that additional mechanisms (e.g., manual assignment) can be added.

3.2.2 Robot Controllers

A *robot controller* in MRTeAm is an agent that implements a robot's bidding and task execution behaviours. For the multi-robot routing domain investigated in this thesis, task execution entails two procedures: *task selection*—deciding which task location to visit next out of several possible options—and *navigation* to the location of the selected task. While en route to a task location, a robot may need to take precautions against colliding with other robots that lie in its path.

Bidding

In the *bid computation* phase of an auction (Section 3.2.1) a robot calculates and submits bids for tasks that were advertised in an announcement phase. Bids values are based on *path distances* between locations. A path distance between two locations on a map is calculated by invoking the ROS navigation stack's global planner (Figure 3.5), which finds the shortest path through the cost map between the two locations (around obstacles) using an A* search. Distances between the *waypoints* of the path returned by the global planner are then summed to compute total path distance.

When a task is awarded to a robot by the auctioneer, it it placed in the robot's *agenda*, a list of incomplete tasks. A robot may consider the locations of tasks in its agenda to compute bids (Section 4.4).

Task Selection

When an execution phase of an experiment begins, a robot controller must decide which task in its agenda to execute first. Ideally, a robot would compute an optimal path of tasks in its agenda (i.e., a Hamiltonian path) that minimises the total cost of visiting all task locations. In order to compute such an optimal path, the robot controller would need to invoke the global planner $O(|agenda|^2)$ times to find the path distance between between each pair of locations. Instead, the robot finds the path distance between its current location and every task in its agenda, invoking the path planner $O(|agenda|)$ times, and chooses the closest task as a *nearest-neighbour* candidate. (Other heuristic approaches are possible [104].) For single-robot tasks with no precedence-constraints between them (**SR-IT**), this nearest neighbour is selected as the next task to execute.

For multi-robot (**MR**) or precedence-constrained (**CT**) tasks, the robot controller uses the following procedure to select its next task:

1. Construct a directed graph G_A of tasks in the robot's agenda. Each node $t \in G_A$ represents a task, and an edge between two nodes $t_p \rightarrow t_q$ indicates a precedence constraint such that task t_p must be completed before task t_q.
2. $candidates_{LC} \leftarrow$ the list of tasks represented by nodes with no incoming edges, found by topological sort of G_A.
3. $candidates_{MR} \leftarrow$ the list of multi-robot (MR) tasks in the agenda that have at least one team mate en route to or arrived and waiting at their locations.
4. If $candidates_{MR} \neq \emptyset$, then $candidates \leftarrow candicates_{LC} \cap candidates_{MR}$. Else, $candidates \leftarrow candidates_{LC}$.
5. Find the path distance to each task $t_c \in candidates$. Choose the task with the minimum distance, breaking ties randomly.

While en route to task locations, robots may drive close enough to risk a collisions. Collisions are avoided with a simple avoidance protocol similar to that described in

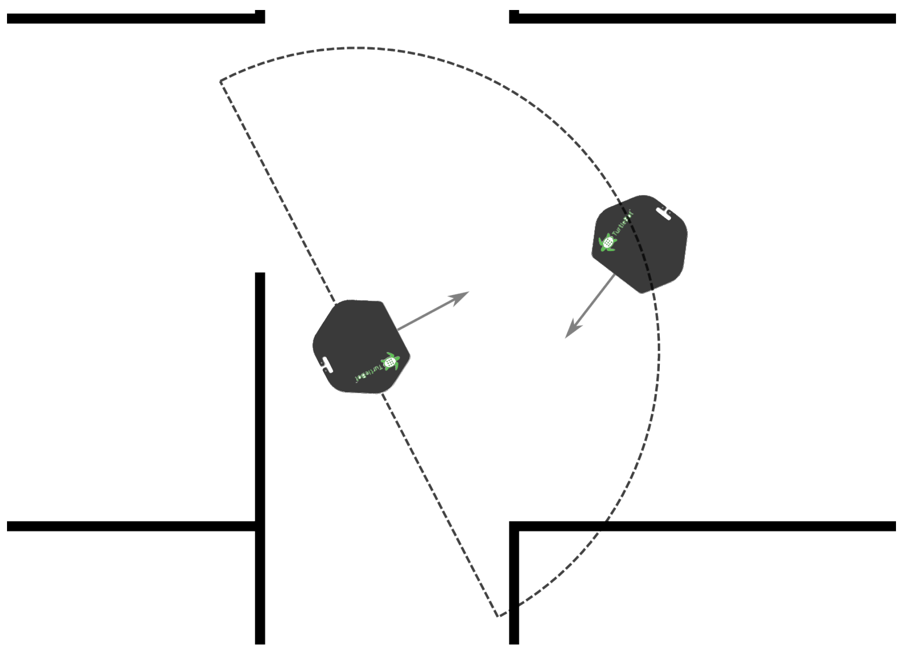


Figure 3.6: Semi-circular "danger zone" of a collision risk between two robots.

[51]. Each robot controller maintains a list of positions of other robots in a list that is updated dynamically. Whenever an update of a team mate's position is received, the robot controller checks whether or not it is inside a semi-circular "danger zone" around it (Figure 3.6). If the team mate's position falls within a "danger zone", both robots pause and calculate the remaining path distance to the task they are currently pursuing. The robot with the shorter remaining path distance (ties are broken randomly) is given the right of way and allowed to resume its movement towards its task location. The team mate robot waits until its danger zone is clean and then resumes moving.

3.3 ROS Master Bridge

The centralised nature of the naming and directory services provided by a ROS master (3.1) poses a challenge for multi-robot systems. The problem of where to locate a single ROS master, physically, among a team of autonomous robots has no clear solution. Certain nodes like those that publish or consume sensor information require a reliable, high throughput connection. Mobile robots are typically connected over a wireless network (e.g., 802.11 Wi-Fi), which can lack both qualities. A problematic connection between, for example, a node that publishes a robot's distance sensor scans and the node that handles its localisation could cause navigation to fail. Thus, it is common for each robot to run its own ROS master on its host computer.

Communication between nodes that are connected to different ROS masters—*multi-master communication*—is not officially supported by ROS in its current versions. When work on MRTeAm first began, several unofficial implementations of multi-master communication existed but were not very robust or mature.[8]

I have developed my own method for multi-master communication named the **ROS Master Bridge**. The ROS Master Bridge provides a publish/subscribe messaging service that connects multiple ROS masters using RabbitMQ,[9] a messaging server that implements the Advanced Message Queueing Protocol [128]. A Master Bridge service runs on a central host (Figure 3.7). Each robot connects to the master bridge using a local *master bridge relay* node, which is responsible for publishing local ROS messages to and receiving messages from the master bridge.

In an MRTeAm experiment, only messages that relate to the experiment are relayed between robots by the master bridge: announcements, bids, awards, and robot positions (Appendix B.2). In this way, the bulk of messaging traffic produced by a robot's navigation stack (such as coordinate frame transformation messages[10]) remains local to the robot, and does not saturate the network.

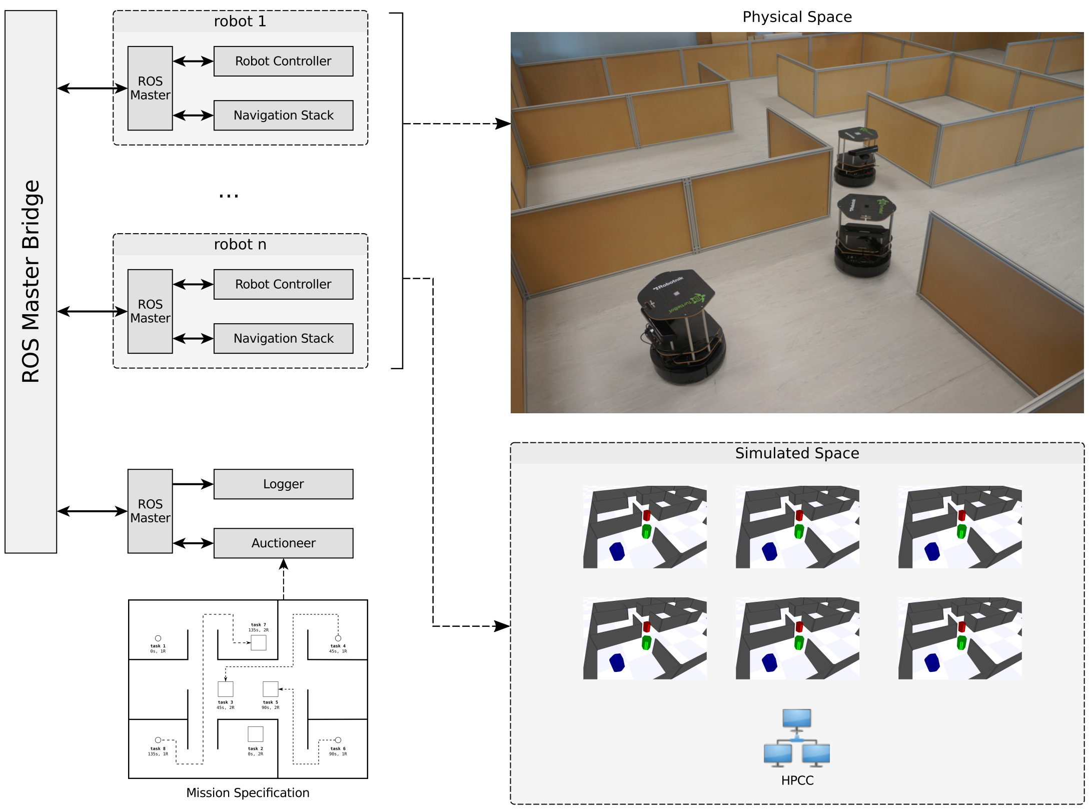


FIGURE 3.7: MRTeAm System Architecture with ROS Master Bridge.

[8]http://wiki.ros.org/sig/Multimaster/
[9]https://www.rabbitmq.com/
[10]http://wiki.ros.org/tf

3.4 Tools

MRTeAm contains several tools to support the generation and analysis of experiments, including a pipeline that converts experimental results into training sets for a machine learning library [94] (discussed in Chapter 8). A *scenario generator* creates scenarios in which task locations and other properties of tasks, such as their arrival times, precedence constraints and the number of robots required ($[1, n]$, where $n = |R|$ the number of robots in the team), can be randomised. Messages and events generated during the course of an experiment are recorded in a ROS bag, a special type of logging file.[11]

3.5 Physical Platform

The physical environment for MRTeAm experiments is an arena that has been built in the smARTLab UGV laboratory at the University of Liverpool. The arena is 8 metres long and 6 metres wide, with walls 0.5 metres high. The layout of the arena is meant to resemble a simplified floor plan of an office-like building with spatial features like rooms and corridors. The arena can be reconfigured in different layouts.

MRTeAm's physical robot platform is the Turtlebot 2.[12] Each robot is equipped with a differential-drive mobile base and a colour- and depth-sensing camera and is controlled by an on-board laptop computer that runs the ROS navigation stack. Each robot is capable of fully autonomous navigation, and only depends on the auctioneer to receive tasks. Robots in the physical arena are shown in Figures 3.8.

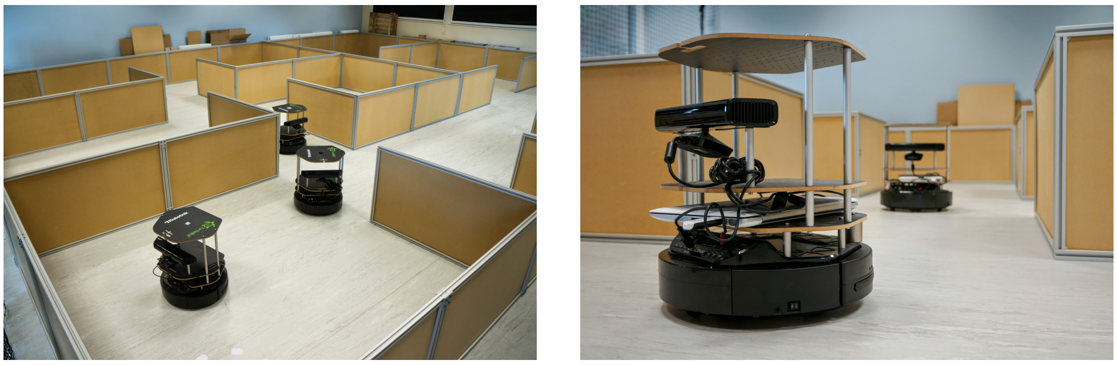


FIGURE 3.8: Turtlebots in the *smARTLab* environment.

3.6 Simulation Platform

The layout of the physical arena is reproduced inside of the *Stage*[13] robot simulator [37]. The Turtlebots' physical properties (e.g., size, shape, acceleration) and sensors are

[11]`http://wiki.ros.org/rosbag`
[12]`http://www.turtlebot.com/turtlebot2/`
[13]`http://rtv.github.io/Stage/`

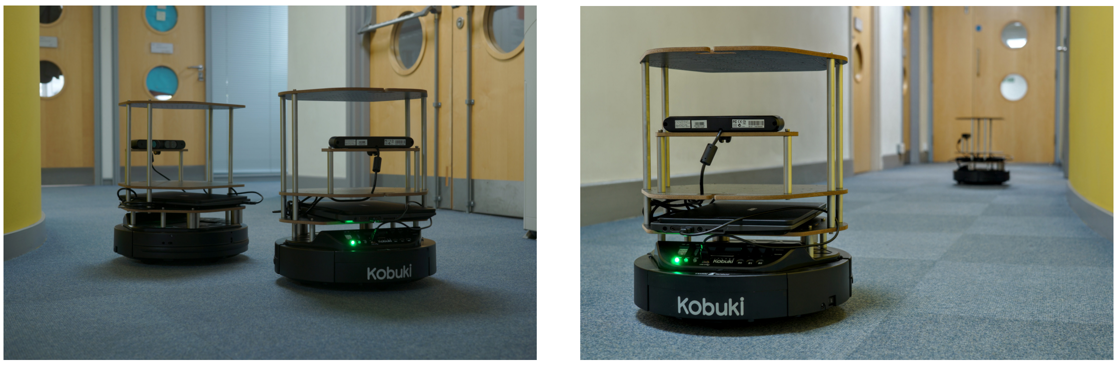
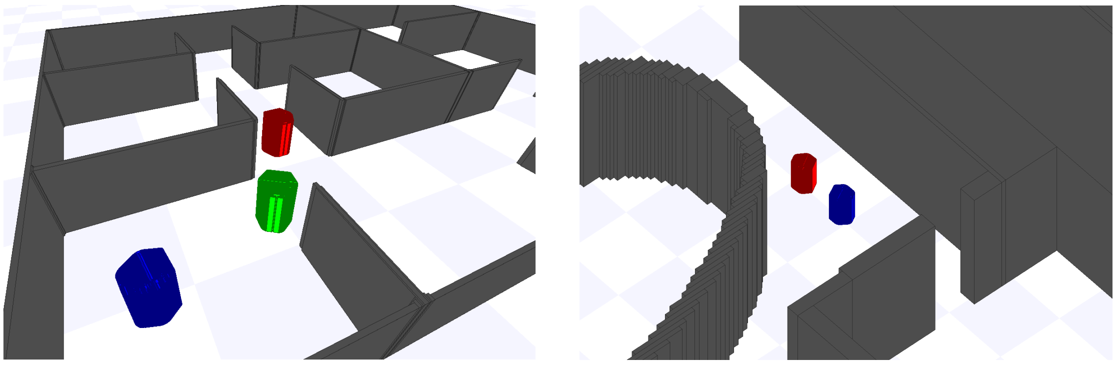


FIGURE 3.9: Turtlebots in an office-like environment.

modelled in Stage. The fidelity of the simulation is important. Most of the experiments presented in this thesis are conducted in simulation, with the goal of validating simulation results on physical robots [107]. Section 4.7 (pg. 47) discusses the differences between simulated and physical settings and how these differences can be measured and mitigated.

I put approximately five months of effort into building the software necessary to run simulations on the University of Liverpool's Chadwick cluster.[14] This effort involved porting ROS and its software dependencies to the cluster's computing environment. The Chadwick cluster is a grid computing resource with 200 compute nodes, each with 2 8-core Intel Xeon CPUs. I am able to run simulation experiments on the Chadwick cluster in parallel on a large scale, which allows for a large number of experiments to be run in a reasonable amount of time.



(a) A simulation of the smARTLab arena. (b) The office environment in Figure 3.9.

FIGURE 3.10: Turtlebots in the Stage simulator.

[14]`http://www.liv.ac.uk/csd/escience/`

3.7 HRTeam

HRTeam [115] is a multi-robot framework with agents that implement auction-based multirobot task allocation mechanisms for multi-robot routing missions, with implementations similar to those of MRTeAm. HRTeam uses software from the Player Project [37] to handle robot navigation and a *central server* to communicate messages between auctioneer and robot controller agents (Figure 3.11).

The robot platform of HRTeam is the Surveyor SRV-1 Blackfin robot (Figure 3.12a), which was modelled in the Stage simulator [37] for the experiments reported in Chapter 5.

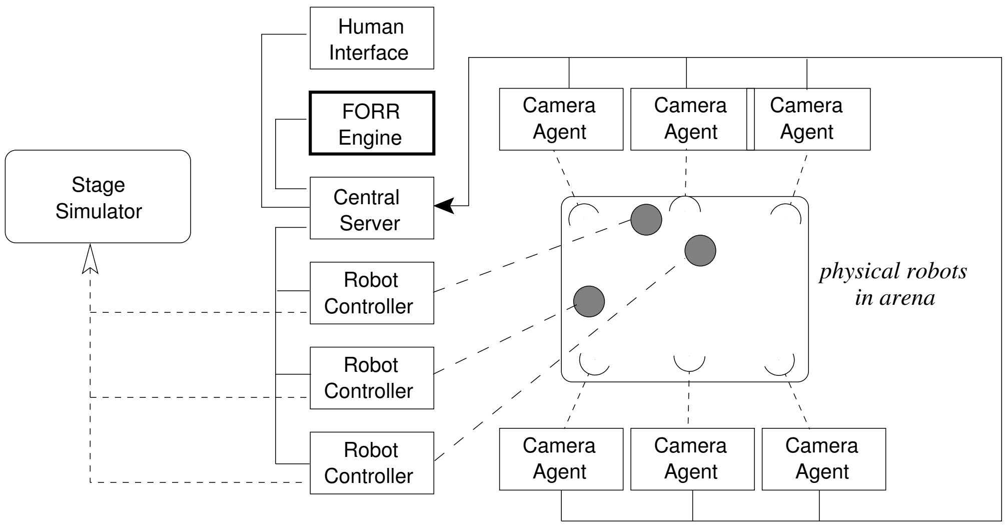
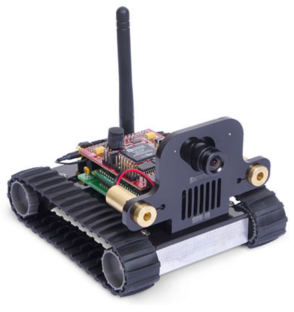
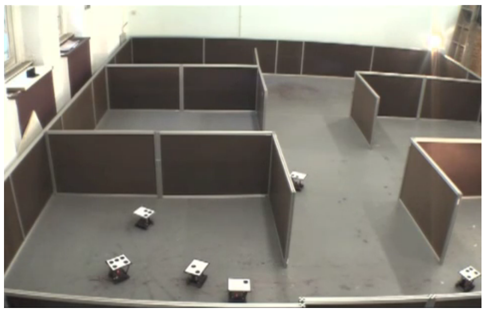


Figure 3.11: HRTeam system architecture



(a) Surveyor SRV-1 Blackfin robot.



(b) The physical arena for HRTeam missions.

Figure 3.12: HRTeam physical platform.

3.8 Summary

This chapter has introduced the MRTeAm software framework, which was used to conduct the majority of experiments presented in this thesis. I have developed MRTeAm to run on both physical and simulated robots with minimal difference in the behaviour of the two. The following chapter describes the design of experiments presented in Chapters 5–8, defining the terminology, notation, and the metrics used to measure performance.

Chapter 4

Experimental Design

This chapter describes common elements of experiments presented in this thesis and a rationale for their design. Section 4.1 describes a hypothetical setting in which experiments take place. Section 4.2 describes the task environments investigated in Chapters 5–8. Section 4.3 defines the terms and notation used to discuss experiments. Section 4.4 defines the task allocation mechanisms employed in experiments, including their expected computation and communication costs. Section 4.5 discusses the phases of an experiment time line. Section 4.6 defines the metrics that are used to measure performance. Section 4.7 discusses the differences between simulated and physical settings and how simulations can be used to develop and test algorithms that will ultimately work on physical robots despite a "reality gap" that exists between the two settings. Finally, Section 4.8 describes the practical factors that affect the design of experiments conducted in Chapters (5–8).

A portion of the results and discussion presented in Section 4.7 were published in Schneider et al. (2015) [107].

4.1 Mission Setting

A primary inspiration for the experiments presented in the following chapters is that of a search and rescue mission performed by a team of autonomous mobile robots. The mission is set on a floor of an office-like building with features like rooms and corridors that link them (Figure 4.1). Multi-stage missions, in which an advance team of surveying robots first explores and maps an area and discovers points of interest in it, have been employed in ground-based [66] and undersea [124] archaeological inspection and recovery settings. Here we assume that the area has been explored and mapped beforehand and that the map is available to the robots. The mission involves sending the robot team into the building from safe entry points at the edges of the map to travel to pre-identified points of interest and perform some action, for example, recording video or offering aid to a person in distress.

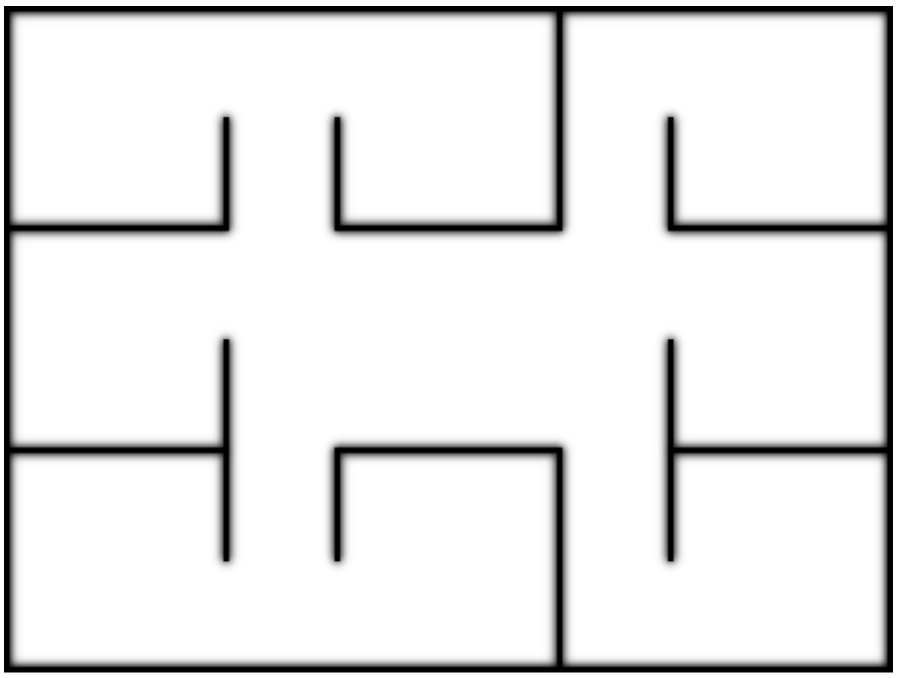


Figure 4.1: An office-like floorplan of the type used in experiments in Chapters 5–8.

4.2 Task Environments

The experiments presented in Chapters 5–8 investigate the performance of task allocation mechanisms employed in a number of **task environments** classified by the taxonomy given in Section 2.4. The "landscape" of task environments investigated is defined by three axes:

- Single-robot (**SR**) vs. multi-robot (**MR**) tasks
- Independent (**IT**) vs. constrained (**CT**) tasks
- Static allocation (**SA**) vs. dynamic allocation (**DA**)

A single task environment in this landscape is identified by a triple, for example, *SR-IT-SA*, which indicates an environment in which all tasks are single-robot tasks, have no constraints between them, and are all allocated at the beginning of an experiment.

4.3 Terms and Notation

The following terms and notation are used to describe the experiments presented in Chapters 5–8:[1]

- A **map** specifies the two-dimensional extents of a space and the arrangements of obstacles within it.
- A **team** is a set of n robots $R = \{r_0, \ldots, r_{n-1}\}$.

[1]These definitions may be different to those found in [106, 108–110]

- A **starting configuration** specifies the location, r_{start}, on a map of each robot in the team at the beginning of an experiment.
- A **scenario** is a set of m tasks $T = \{t_0, \ldots, t_{m-1}\}$ situated on a map. Each task $t \in T$ has the following properties:
  - $t.pos$, a fixed position on the map;
  - $t.arr$, the *arrival time* of the task;
  - $t.req$, the number of robots required to complete the task.

  A scenario also contains *ordering constraints* CT, a set of pairs of tasks (t_p, t_q), $t_p, t_q \in T$ such that t_p must be completed before t_q can proceed.
- A **mission** M comprises a map, a scenario, and a team with a starting configuration $M = \{map, T, R\}$

4.4 Task Allocation Mechanisms

Four task allocation mechanisms are employed in experiments in Chapters 5–8: *Round-robin* (RR) allocation; the *ordered single-item* auction (OSI) auction; the *sequential single-item auction* (SSI) [58]; and the *parallel single-item auction* (PSI) [58]. These are standard in the design space of MRTA mechanisms but do not explore it fully (combinatorial auctions [8, 77, 85, 103], for example, are not investigated here). Sequential single-item (SSI) auctions [58, 111] and their extensions [46, 139, 140], parallel (PSI) auctions [22, 58, 86], and sequential auctions (as in OSI) [10] have been well-investigated. Round-robin is a non-auction based mechanism, developed in [90] and [106] as a kind of baseline against which auction-based mechanisms can be compared.

The four task allocation mechanisms are defined in turn in the following subsections. It it helpful to define here some common terms used in their definitions. In the notation of Koenig et al. [58], let $PC(r, A)$ be the smallest *path cost* of a robot r to visit all of the task locations in its agenda A from its current location. The *task insertion cost* $TIC(r, t')$ of a robot r for task t' is the marginal path cost increase the robot incurs for adding t' to its agenda. That is, $TIC(r, t') = PC(r, A \cup \{t'\}) - PC(r, A)$. Computing a minimal path cost is NP-hard, so the mechanisms presented here use a heuristic method for computing path insertion cost (Algorithm 2, Line 3 and Algorithm 3, Line 4) as described by Lagoudakis et al. [70].

4.4.1 Round Robin (RR)

In *round robin* allocation (Algorithm 1), a cycling iterator identifies each robot by an index and keeps track of a "current" index (ci). Tasks T that make up a mission are sorted in an arbitrarily ordered list. For each task $t \in T$, the cycling iterator is queried for the index of a robot r_{next}, and the task is awarded to that robot wr, who adds

Algorithm 1 Round Robin (RR)

```
 1: $ci \leftarrow 0$
 2: procedure CycleIterator()
 3:     $r_{next} \leftarrow r_{ci}$
 4:     if $ci < (|R| - 1)$ then
 5:         $ci \leftarrow ci + 1$
 6:     else
 7:         $ci \leftarrow 0$
 8:     end if
 9:     return $r_{next}$
10: end procedure
11: for all $t \in T$ do
12:     for $1 \dots t_{req}$ do
13:         $wr \leftarrow CycleIterator()$
14:         $T(wr) \leftarrow T(wr) \cup \{t\}$
15:         $T \leftarrow T \setminus \{t\}$
16:     end for
17: end for
```

Algorithm 2 Ordered Single-Item (OSI) Auction

```
1: for all $t \in T$ do
2:     for all $r \in R$ do
3:         $bid(r, t) \leftarrow PC(r, T(r) \cup \{t\}) - PC(r, T(r))$
4:         $submit(r, t, bid(r, t))$
5:     end for
6:     $wr \leftarrow arg\, min_{r \in R} bid(r)$
7:     $T(wr) \leftarrow T(wr) \cup \{wt\}$
8:     $T \leftarrow T \setminus \{wt\}$
9: end for
```

the task to its agenda. A (RR) allocation has a winner determination phase but no announcement or bid computation/submission phases

This is clearly not a particularly efficient way to approach task allocation. It provides a valid solution to an allocation problem but does not attempt to optimise any performance objective. But it does provides a baseline against which other mechanisms can be tested. We referred to this as the "greedy taxi" policy in our earlier work [90], because this policy emulates the behaviour of a taxi rank.

4.4.2 Ordered Single-Item (OSI)

An *ordered single-item* auction (Algorithm 2) is a type of sequential auction. The tasks T of a mission are sorted in an arbitrarily ordered list, as in an RR allocation. Each task $t \in T$ in turn is announced to all robots, who then bid on it. The value that a robot r bids is the cost of inserting the task into its agenda $T(r)$. The robot that submits the lowest bid is determined the winner wr and awarded the task, which the robot inserts into its agenda.

Algorithm 3 Sequential Single-Item (SSI) Auction

```
 1: while $T \neq \emptyset$ do
 2:     for all $r \in R$ do
 3:         for all $t \in T$ do
 4:             $bid(r, t) \leftarrow PC(r, T(r) \cup t) - PC(r, T(r))$
 5:         end for
 6:         $bid(r) \leftarrow \min_{t \in T} bid(r, t)$
 7:         $target(r) \leftarrow arg\,min_{t \in T} bid(r, t)$
 8:         $submit(r, target(r), bid(r))$
 9:     end for
10:     $wr \leftarrow arg\,min_{r \in R} bid(r)$
11:     $wt \leftarrow target(wr)$
12:     $T \leftarrow T \setminus \{wt\}$
13:     $T(wr) \leftarrow T(wr) \cup \{wt\}$
14: end while
```

Algorithm 4 Parallel Single-Item (PSI) Auction

```
 1: for all $t \in T$ do
 2:     for all $r \in R$ do
 3:         $bid(r, t) \leftarrow PC(r, \{t\})$
 4:         $submit(r, t, bid(r, t))$
 5:     end for
 6: end for
 7: for all $t \in T$ do
 8:     $wr \leftarrow arg\,min_{r \in R} bid(r)$
 9:     $T(wr) \leftarrow T(wr) \cup \{wt\}$
10: end for
```

4.4.3 Sequential Single-Item (SSI)

In the *sequential single-item* auction (Algorithm 3) [58], all unallocated tasks are announced to all the robots simultaneously. Each robot bids on the task with the lowest cost (again computed as in OSI) and the task with the lowest bid is awarded to the robot *wr* that placed the bid. The winning robot inserts the task into its agenda and the process is repeated until all points have been allocated.

4.4.4 Parallel Single-Item (PSI)

In a *parallel single-item* auction (Algorithm 4) (introduced as something of a strawman in [58]) all unallocated tasks are announced to all the robots simultaneously as in SSI. Each robot submits a bids for the cost of a path from its current location to the location of each of the tasks announced. All the tasks are allocated in one round, however, with each point going to whichever robot made the lowest bid on it.

	RR	OSI[*]	SSI[*]	PSI
Bids computed	–	$m \times n$	$\left(\frac{m(m+1)}{2}\right) n$	$m \times n$
Winner Determination	–	$m \times n$	$m \times n$	$m \times n$
Announce messages	–	$m \times n$	$\left(\frac{m(m+1)}{2}\right) n$	$m \times n$
Bid messages	–	$m \times n$	$m \times n$	$m \times n$
Award messages	m	m	m	m

TABLE 4.1: Computation and communication costs of the four task allocation mechanisms used in experiments. $m = |T|$ is the number of tasks in a mission and $n = |R|$ is the number of robots in a team.

[*] Each bid includes the cost of task insertion, which grows linearly in the size of the robot's agenda.

4.4.5 Costs

Table 4.1 gives the computation and communication costs of the four mechanisms in terms of the number of tasks in a mission $m = |T|$ and the number of robots in a team $n = |R|$. Note that although OSI and PSI appear to have the same bid computation costs, each bid computation in an OSI auction calculates a task insertion cost (Algorithm 2, Line 3), which grows in the size of a robot's agenda, while a PSI bid computation does not.

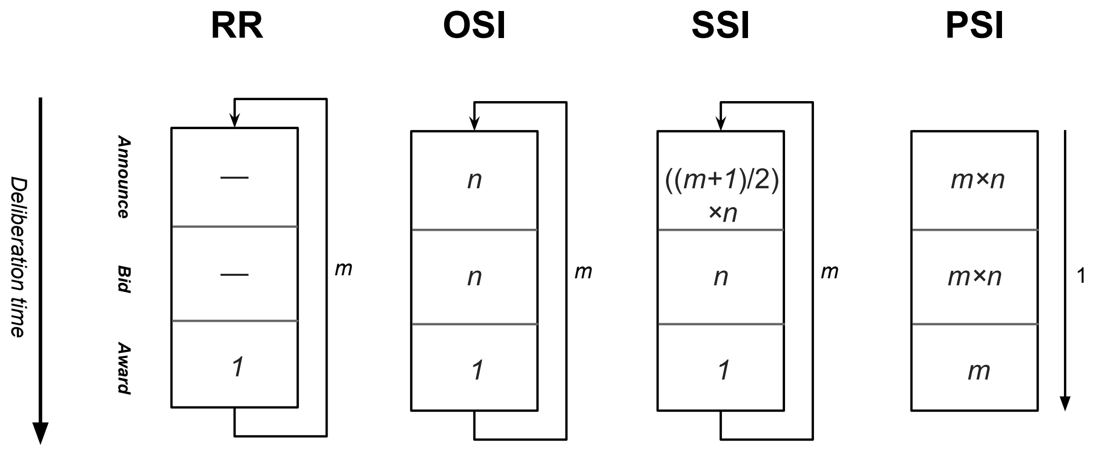
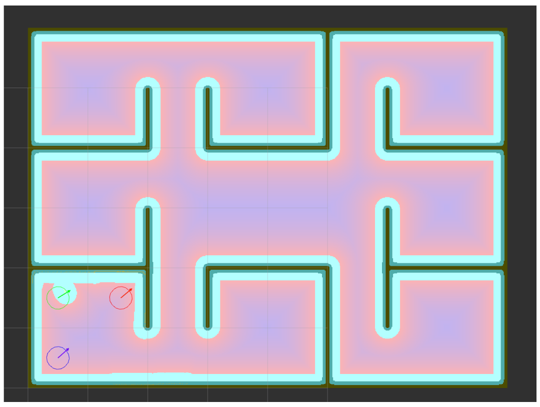
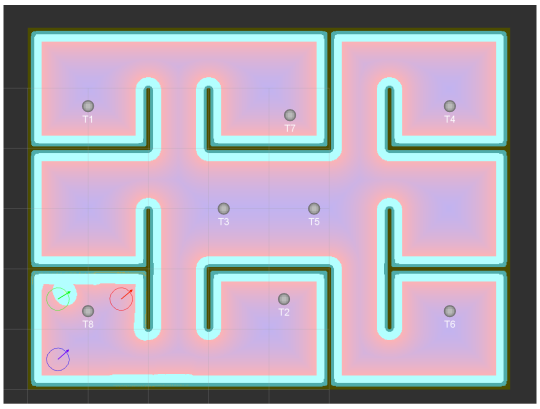


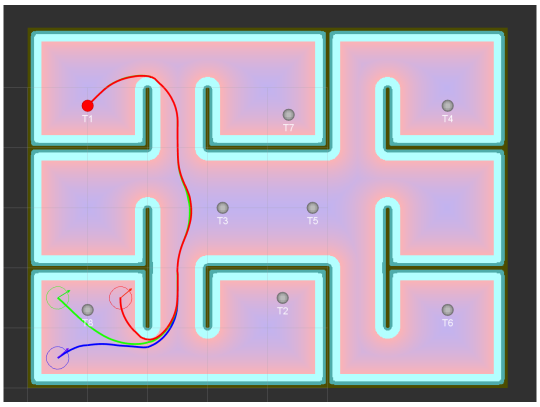
FIGURE 4.2: Number of messages communicated in the deliberation phase of each of the four mechanisms. $n = |R|$ is the number of robots on the team. $m = |T|$ is the number of tasks in a mission.



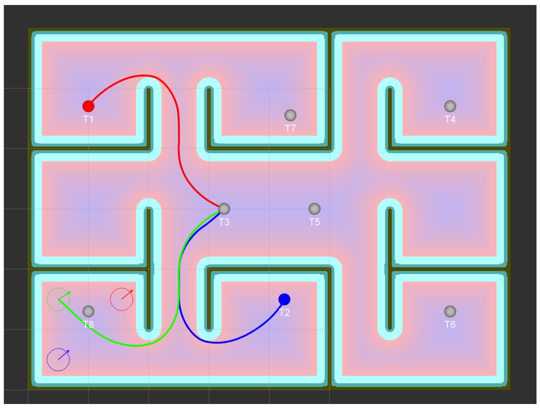
(a) Experiment Start



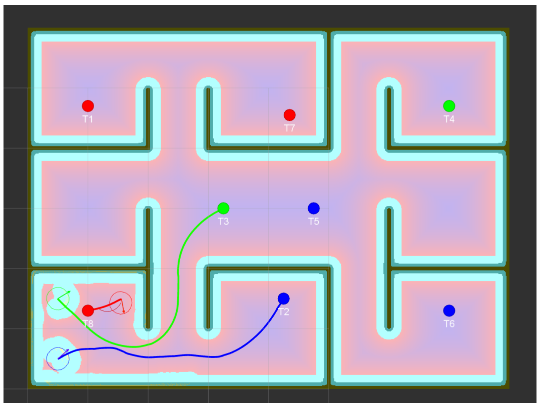
(b) Scenario Loaded



(c) Task 1 awarded to Robot 1 (Red)



(d) Task 2 awarded to Robot 3 (blue)



(e) Execution Phase

FIGURE 4.3: Phases of an experiment.

4.5 Phases of an MRTeAm Experiment

At the beginning of an experiment, tasks are loaded from a scenario (Fig. 4.3a) and placed into the auctioneer's task pool. The experiment then proceeds to a *deliberation* phase, in which tasks are allocated. A task allocation mechanism is chosen by the auctioneer and all tasks in its task pool are assigned to robots according to the rules of the chosen mechanism.

After tasks have been allocated the deliberation phase ends and the experiments proceeds to an *execution phase* (Fig. 4.3). Each robot chooses its next task to execute and moves to that location. Tasks are abstracted so that a robot "performs" a task by moving to its location.

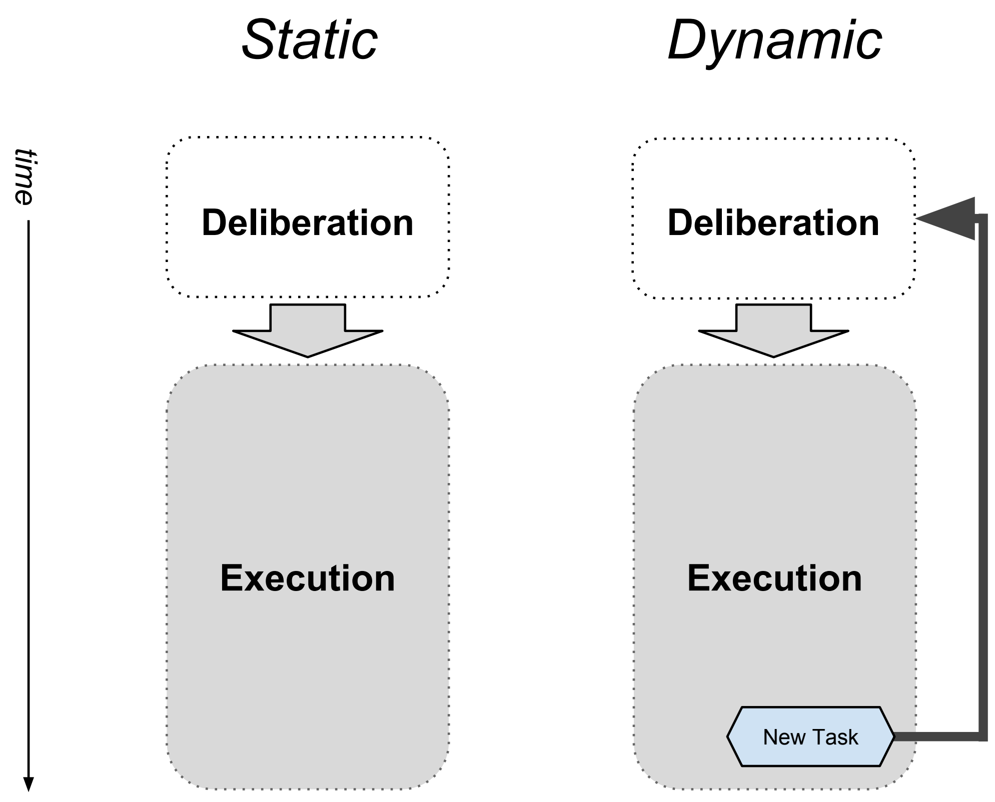


FIGURE 4.4: Phases of an experiment in static allocation (SA) and dynamic allocation (DA) task environments.

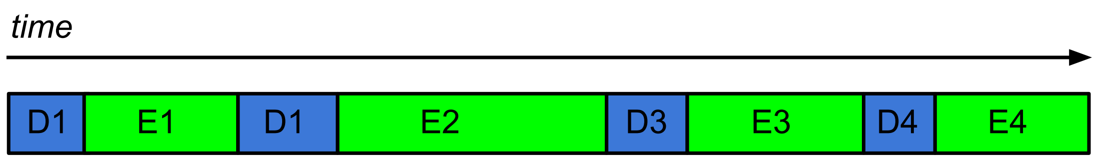


FIGURE 4.5: Multiple deliberation and execution phases in a dynamic allocation (DA) task environment.

In a static allocation environment (**SA**), all tasks in a mission are allocated in a single deliberation phase at the start of an experiment followed by a single execution phase. In a dynamic allocation environment (**DA**), a deliberation phase begins each time a task is added to the auctioneer's task pool (via a delayed timer). If this happens during an execution phase, the execution phase is aborted, the robots are paused, and a new

Metric	Description	Unit
team distance	actual distance covered by robots	metres
run time	total length of a run	seconds
– *deliberation time*	time spent allocating tasks	seconds
– *execution phase time*	time spent executing tasks	seconds
near collisions	number of near collisions between robots	$\mathbb{N}$
delay time	time spent waiting to resolve collisions	seconds
movement time	time spent actually moving towards tasks (per robot)	seconds
idle time	time between when the first robot completes all of its tasks and the last robot to completes all of its tasks	seconds
waiting time	time spent by robots waiting for others to arrive at multi-robot tasks	seconds

Table 4.2: Performance Metrics

deliberation phase begins. The sequence of phases for static and dynamic environments is shown in Figure 4.4. Fig. 4.5 shows an illustration of an multiple deliberation-execution phases of an experiment in a **DA** environment. Figure 4.6 shows a timeline plot of robot activity over the course of an experiment set in a dynamic allocation (DA) environment. Deliberation phases can seen at 45, 90, and 135 seconds into the experiment.

4.6 Performance Metrics

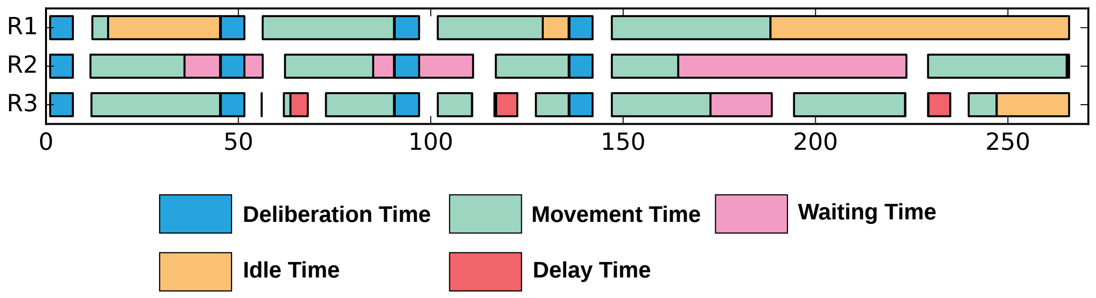


Figure 4.6: An example of an experiment timeline. The experiment begins at time 0 and proceeds to the right. Each row shows the changing state of a robot as it carries out its part of a mission. Time spent moving towards task locations (i.e. actually executing routing tasks) is indicated by green intervals. Other intervals show time spent idle or coordinating with other robots. Gaps indicate *task selection* intervals, when a robot decides which task in its agenda to move to next.

To evaluate the performance of a team, we consider a number of metrics that measure the performance of both individual robots and the team as a whole. In any work with robots, power consumption is the fundamental scarce resource that a robot possesses.

Robot batteries only last for a limited time, and so, all other things being equal, we prefer task allocations and subsequent executions that minimise battery usage. As in [57, 58, 69, 70, 122], therefore, we measure the distance travelled by each robot over the course of an experiment since this is a suitable proxy for power consumption.[2] ***Team distance*** is the sum of the distances travelled by the group.

Time to complete a set of tasks is also important. Time is important in exploration tasks—in search and rescue activities, in patrolling, and possibly in demining[3] —and so we measure ***run time***, the time between the start of an experiment and the point at which the last robot on the team completes the tasks allocated to it.

A component of run time is ***deliberation time***, the time that it takes for the tasks to be allocated amongst the robots. Deliberation time matters because it feeds into the overall time required to complete a set of tasks, but also because it allows us to establish how different allocation mechanisms compare in terms of the computational effort and communication resources required to run them. Another component of run time is ***execution phase time***, the time it takes robots to execute tasks during an execution phase once they have been allocated.

The planned paths of two robots may cross, leading to a potential collision. In such cases, the robots stop and negotiate which of them will receive the right of way. Such ***near collisions*** are counted as well as the ***delay time*** they incur. ***Movement time*** is the time robots spend actually moving, without interruption (e.g., by a near collision), toward tasks. A robot might arrive at a task location before the necessary conditions for executing the task have been met, such as waiting for a team mate to arrive at the location of a multi-robot task. This particular kind of time is measured as ***waiting time***.

Also measured is ***idle time***, the amount of time that robots sit idly during a mission. This is computed as the time that elapses between when a robot completes its last task and when all robots on the team have completed all of their assigned tasks. This gives a way of quantifying how equally tasks are distributed among robots, and it also suggests the extent to which resources are being wasted by a particular allocation. Although a mismatch between the number of tasks and the number of robots means that idle time can be inevitable, idle time represents the use of precious power that is not being directed at task completion.

[2]Note that we compute distance not by looking at the shortest distances between the task locations, but at (as closely as we can establish) the actual distance travelled by the robots during task execution. We collect frequent position updates, compute the Euclidean distance between successive positions, and sum these.

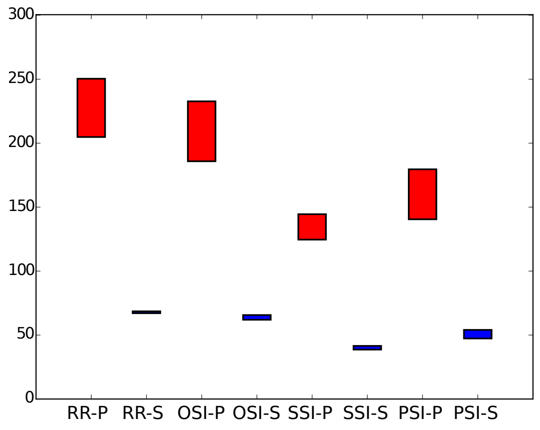
[3]One can easily imagine demining happening against the clock—in humanitarian demining [43], for example, there may be the need to demine an area in order to allow refugees to move safely away from a dangerous situation.

4.7 From Simulated to Physical Robots

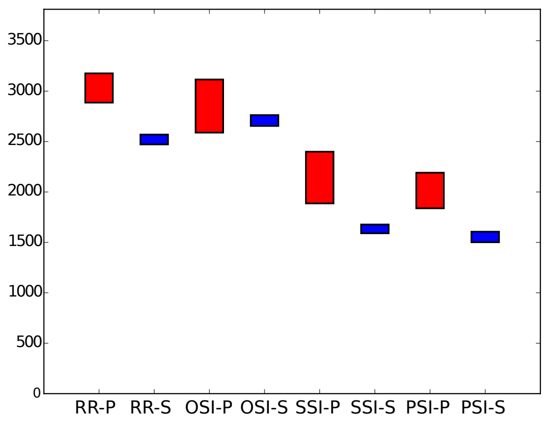
It is common practice in robotics to employ simulation as a means to evaluate an approach which is intended to be deployed in some type of physical environment [12, 52, 80, 82]. The advantage of simulation over reality is that we typically have more control over and easier access to the simulated environment, and experiments are easier to automate and scale to large numbers. This implies that it is simpler to develop and test algorithms, controllers, or whatever we are working on, in the simulated environment first—i.e., before they are deployed on physical robots.

Especially in cases where testing in reality is risky (e.g., nuclear cleanup) and/or expensive (e.g., planetary exploration), it would be very useful to know how much we are gaining by the knowledge obtained in the simulation environment. As it is sometimes impractical to develop robot behaviours on physical hardware, there has been a good deal of investigation into developing behaviours in simulations. An early example of this is the work of Koza (1991) [63], which used genetic programming to recreate the kind of navigation that Matarić (1990) [79] had hand-coded, and led some to conclude that it would be straightforward to use evolutionary techniques to learn robot controllers that could be dropped into real robots that would then operate as desired in the real world. Responding to this position, Brooks (1992) [12] raised concerns about the transferability of behaviours learned in simulations due to significant differences between simulation and physical environments. First, working purely in simulation, that is without regularly checking the results of the simulation against what happens in the real world, could lead to focusing on problems that just don't exist in the real world. Second, if simulators do not accurately model the errors that occur in sensing and actuation, techniques that evaluate their output only in simulation are unlikely to evolve controllers that will work on real robots. Jakobi et al. (1995) [52] introduced the term *reality gap* to describe the differences between reality and simulation that Brooks had described, and went on to provide evidence both of the existence of the gap and of the possibility of overcoming it. They evolved controllers under three conditions: no noise, noise equivalent to that measured in the real world ("observed noise" in their terminology), and much more noise than is observed in reality. Their results showed that controllers that evolved with no noise were too brittle, and relied on behaviours that could not be reliably replicated on a physical robot, while those that evolved with too much noise ended up relying on the existence of the noise in order to work. Controllers that evolved with "observed noise" could be transferred to physical robots that behaved similarly to their simulated counterparts.

Considering the issues presented by the *reality gap* between simulated and physical settings, I have investigated the extent to which the approach of using results obtained from simulation can be applied to physical robots with the MRTeAm and HRTeam frameworks, with the ultimate goal of developing a method of mechanism selection (Chapter 8 on physical robots. In Schneider et al. (2015) [107], I showed that, while differences in multi-robot performance certainly exist between simulation and physical



(a) Execution Time



(b) Distance Travelled

FIGURE 4.7: Comparison of results obtained from simulation and physical experiments. Each plot compares results from physical (red) and simulation (blue) experiments between the Round Robin (RR), Ordered Single-Item (OSI), Sequential Single-Item (SSI), and Parallel Single-Item (PSI) mechanisms. The left plot compares execution times in seconds; the right plot compares distance travelled in centimetres.

settings, the *relative* performance differences among task allocation mechanisms largely hold across the two settings.

Figure 4.7 shows absolute performance differences observed among four task allocation mechanisms in the same set of experiments conducted with the HRTeam framework, similar to those presented in Chapter 5, performed in physical (red) and simulated (blue) settings for the *execution time* (Figure 4.7a) and *distance travelled* (Figure 4.7b) metrics. The simulated robot team clearly travels shorter distances and requires less time to complete its mission than its physical counterpart. Execution time is lower for the simulated robots because they consistently moved faster than their physical counterparts, while the distance they travelled is lower because, although they moved through a map with the same spatial configuration as their physical counterparts, they encountered fewer localisation errors and travelled in straighter paths.

The *relative* performance differences of the four mechanisms, however, which is of primary interest for the work presented in this thesis, largely hold across the two settings. These relative relationships can be seen more clearly in the rank-ordering of performance for the same experiments, shown in Figure 4.8. The top row of each plot shows, in the physical setting, a rank ordering of the four task allocation mechanisms according to a particular metric, while the bottom row shows the same for the simulated setting. While the rank ordering is the same across the two settings for the *deliberation time*, *execution time* and *distance travelled* metrics, the primary performance metrics investigated in this thesis, differences in rankings do exist. Figures 4.8d–4.8f show that rankings for the *idle time*, *near collisions* and (consequently) *delay time* metrics differed across the two settings. While the differences are "misaligned" by at most one rank-order, they suggest that these metrics may be more sensitive to the ability of the simulator to represent the type of "observed noise" discussed by Jakobi et al. [52]. The results of

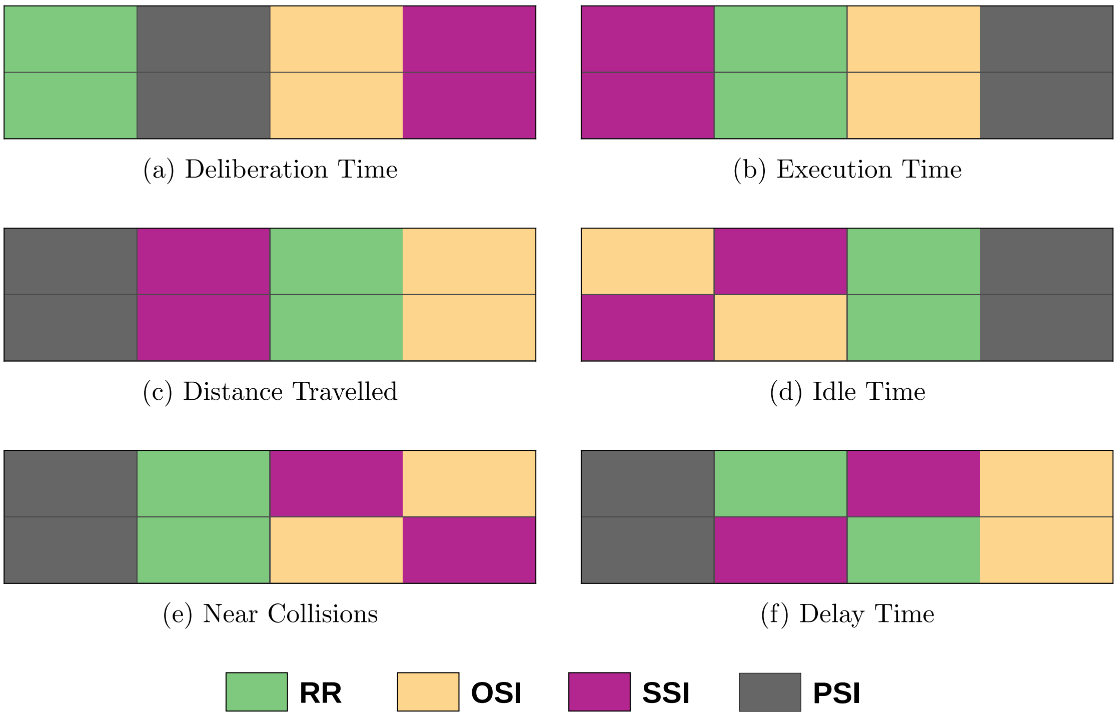


FIGURE 4.8: Comparison of rank-ordered results obtained from simulation and physical experiments for the Round Robin (RR), Ordered Single-Item (OSI), Sequential Single-Item (SSI), and Parallel Single-Item (PSI) mechanisms. The top row of each plot shows rank-orderings of results from physical experiments while the bottom shows results from corresponding experiments performed in simulation. Values are ordered lowest to highest from left to right and measure time (a, b, d, f), distance (c), and counts (e).

simulation experiments conducted with the MRTeAm framework presented in Chapter 5 show similar rank-ordering relationships to those of physical experiments described in Appendix A.2 (pg. 129).

4.8 Practical Considerations

In the experiments presented in Chapters 5–8, the size of the team is fixed at $n = 3$ and the number of tasks, $m = \{8, \dots, 16\}$, is relatively small. As discussed in the previous section, it is desirable to maintain a consistent experimental design across simulated and physical settings since we want to develop and test algorithms in simulation but ultimately field them on physical robots. It is also important to employ a high-fidelity simulator that reproduces as much of the "observed noise" of the physical setting as is practical and to employ the same navigation controllers, which operate on streams of noisy sensor data as input, in both settings.

In simulation, the size of the team is constrained by the computing resources available on a single workstation or HPCC compute node, with $n = 3$ being the practical upper limit on the number of instances of the ROS navigation stack (Section 3.1.1) that can run simultaneously. It was found to be infeasible to coordinate multiple HPCC compute nodes, running instances of the navigation stack in parallel, to increase this limit. In the physical setting, this constraint is somewhat relaxed since each physical robot runs an instance of the navigation stack on its own computing hardware. Nevertheless, team size in the physical setting was constrained by the desire to keep consistent with simulations. The number of tasks, m, was limited mainly by running time. The ROS navigation stack has a practical limit on the rate at which it can read streams of sensor data and issue motion commands to a robot, whether simulated or physical, and thus the simulator can not run faster than real-time, as in a discrete event simulation. In the type of experimental setup presented in Chapters 5–8, given the size of the team, the number of tasks, and the size of the map, a mission with $m = 16$ tasks typically takes about 10–15 minutes to run. The large number of simulations (on the order of several thousand) required by the method of mechanism selection described in Chapter 8 thus limited the number of tasks that could be allocated and executed in a single mission.

The main line of inquiry pursued in this thesis is not evaluate how the performance of task allocation mechanisms varies as team sizes and numbers of tasks are scaled, but rather to investigate how performance of the mechanisms diverges for a single mission (or small set of missions) as that mission is carried out over a range of task environments. The results of experiments presented in Chapters 5–8 show that the effects of task environments on mechanism performance can be readily seen despite practical constraints on team size and number of tasks.

4.9 Summary

This chapter has described common elements of experiments presented in the following chapters and puts forth a rationale for their design. It has explained the operation and computation and communication costs of the four task allocation mechanisms employed in experiments presented in Chapters 5–8, explained the timeline of a typical experiment, and defined the metrics used to measure performance. It has also discussed the differences between simulated and physical settings and how simulation is a valid setting for developing approaches that are ultimately fielded on physical robots. Finally, it has discussed some practical constraints that limit the scale of missions. These limitations do not, however, prevent the investigation of the performance of task allocation mechanisms as they are carried out over a range of task environments. The following chapter presents a set of experiments conducted in a SR-IT-SA (single-robot, independent task, static allocation) task environment.

Chapter 5

A Static Task Environment (SR-IT-SA)

5.1 Introduction

This chapter compares the performance of the four task allocation mechanisms defined in Section 4.4 in two multi-robot routing scenarios in a task environment (SR-IT-SA) in which tasks are independently executed by single robots and all tasks can be allocated at the beginning of a mission. The aim of the experiments presented here, and of the investigation overall, is to understand how the performance of task allocation mechanisms is affected by the environments in which they are employed, with the goal of establishing the suitability of the mechanisms for different kinds of environments. Dynamic task environments that require on-line problem solving (SR-IT-DA) and tighter coordination between robots can be considerably more complex and are investigated in Chapter 6–7. However, in a SR-IT-SA environment there is still a great deal of variation in the arrangements of task locations, robot locations, and free spaces and obstacles on a map.

Section 5.2 details the design of the experiments performed, including the arrangements of robot and task locations, the size and properties of the robot team, the metrics used to measure performance, and the system platform used to carry out the experiments. Section 5.3 points out key results and Section 5.4 discusses their significance. Section 5.5 summarises the research presented in this chapter.

The experiments and a portion of the results presented in this chapter were published in Schneider et al. (2014) [106].

5.2 Experiments

5.2.1 Mechanisms Tested

The task allocation mechanisms discussed in Section 4.4 were tested in these experiments:

- Round-robin (RR)
- Ordered single-item (OSI)
- Sequential-single item (SSI)
- Parallel single item (PSI)

The implementations of OSI and SSI used in these experiments calculate their bid costs differently to the definition given in Algorithm 2 (lines 3 and 4, respectively). During these auctions, after a robot wins a task it updates a *bid-from* position to that of the task it has just won. In subsequent rounds, the bid value a robot calculates is the distance of a path from its *bid-from* position to a task location. This simple task insertion heuristic does not explicitly attempt to optimise the MiniSum objective as in Tovey et al. [122] and Koenig et al. [58].

The tasks in both scenarios are labelled with an (arbitrary) order. The ordering does not affect the SSI or PSI auctions, but it specifies the order in which tasks are assigned by the RR mechanism and announced to the team in the OSI auction.

5.2.2 Metrics

Each experiment recorded some of the metrics described in Section 4.6:

- *Team Distance* – The sum of the distances travelled by all robots.
- *Deliberation time* – The time taken for the tasks to be allocated to the robots.
- *Run time* – The time between the start of an experiment and the time the last robot on the team completes the tasks allocated to it. This includes deliberation time.
- *Near collisions* – The number of times two robots travelled close enough to detect a risk of collision and triggered a negotiation to avoid colliding.
- *Delay time* – The time robots spent replanning and yielding the right of way while negotiating to avoid a collision; and
- *Idle time* – The time that elapses between when a robot completes its last task and when all robots on the team have completed all of their assigned tasks
- *Movement time* – The time a robot actually spends moving, uninterrupted, towards task locations.

5.2.3 Platform

Experiments were conducted with the HRTeam multi-robot framework (Section 3.7) using the Stage simulator [37]. The simulated robots are modelled on the Surveyor SRV-1 Blackfin robot described in Section 3.7 and have the same characteristics as their physical counterparts (size, shape, acceleration, and maximum speed).

One aspect of the simulation to point out is that all agent processes—the auctioneer and robot controllers—run on a single computer. These experiments do not model the communication quality of physical robots that communicate over wireless networks.

5.2.4 Experimental Setup

Two scenarios are investigated, shown in Figure 5.1c and 5.1d, set on the same map and each with $m = 8$ tasks. In Scenario 1, task locations are distributed more or less uniformly through the map, while Scenario 2 has a different arrangement.

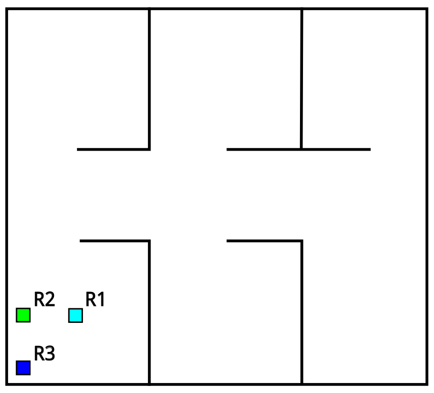
The size of the robot team is fixed at $n = 3$. Two starting configurations were chosen for the team. Figure 5.1a shows the *clustered* configuration, where robots start in the same "room" in the lower left corner of the map. In the *distributed* configuration, robots start at three different corners of the map.

Four task allocation mechanisms were employed in each of 4 missions and there were 10 trials for each combination, for a total of 160 experimental trials:

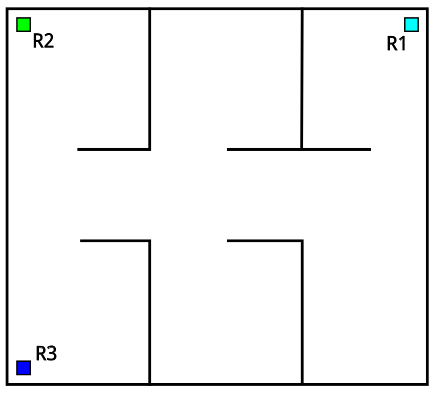
$$160 = 4\ \textit{missions}\ (\ \{\textit{clustered},\ \textit{distributed}\} \times\ \{\textit{Scenario1},\ \textit{Scenario2}\}\ );$$
$$\times 4\ \textit{allocation mechanisms} \times 10\ \textit{trials}.$$

5.3 Results

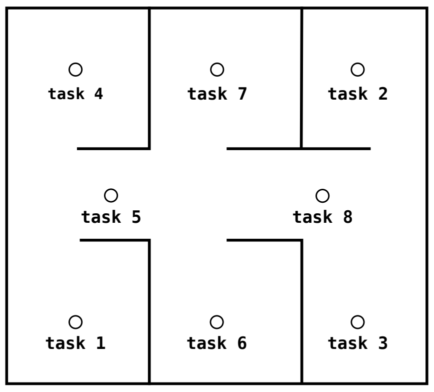
The results of the experiments can be seen in Figures 5.3–5.14. Figures 5.3 and 5.4 show the paths taken by the robots in individual runs of Scenario 1 as solved by different allocation mechanisms. The routes taken by different robots are given in different colours. The aim of the figures is to give a sense of allocations produced by the different mechanisms and the effect that these have on the routes taken by the robots. These trajectories capture many of the key points about the allocations that are echoed in the metrics. The arbitrary allocation of the RR allocation mechanism shows the lack of a clear pattern in the allocations to each robot. The tendency of PSI to allocate tasks unevenly between robots—first noted by Koenig et al. [58]—is clear when comparing it with other mechanisms for the clustered starting configuration (Figure 5.3d). The propensity for SSI to produce tight groupings is clear in its handling of the distributed starting configuration. The allocation of tasks from Scenario 1 to robots is also shown in Figure 5.2, revealing the patterns in each allocation mechanism. Note that each robot visited task locations in the order in which they were allocated, which wasn't necessarily the shortest path amongst the complete set of tasks allocated to the robot.



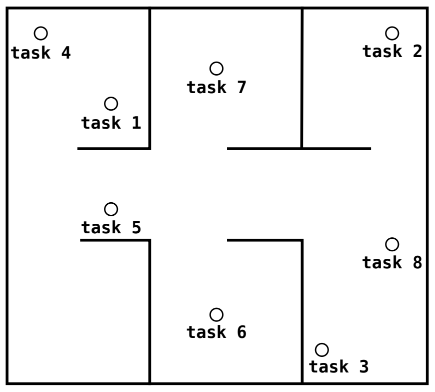
(a) Clustered starting configuration



(b) Distributed starting configuration



(c) Scenario 1



(d) Scenario 2

FIGURE 5.1: Robot starting locations and scenarios. The top row shows two sets of starting locations, *clustered* (a) and *distributed* (b) superimposed on a map of the test environment. The plots on the bottom show task locations in two scenarios.

Figures 5.5–5.14 then plot the average values over 10 runs of each metric for each combination of the four mechanisms, two scenarios, and two starting configurations. Looking at the results from the scenarios side by side makes it clear that the clustered starting configuration provides a steeper challenge for an allocation mechanism than the distributed set. While deliberation times (Figure 5.5 and 5.6) are comparable, team distance and run time (5.7 and 5.8) are reduced for the distributed case, as are the number of collisions and delay times shown in Figures 5.9 and 5.10 (naturally this pair of metrics will be strongly correlated). Tables 5.1 and 5.2 give the same information as Figures 5.5–5.14 but in tabular form, making numerical comparisons possible. The tables also give 95% confidence intervals for the metrics.

Finally, Figures 5.11–5.14 give the metrics that are computed on a per-robot basis, as opposed to those computed for the team as a whole. These individual metrics are distance, delay time (Figure 5.11 and 5.12), travel time, and idle time (Figures 5.13 and 5.14), and the figures give these for each of the three robots for each mechanism and both starting configurations. Since the same robot starts in the same position each time, the distribution across the robots tells us something about the mechanisms. For example, they expose the skewed nature of the results for PSI in the case of the clustered

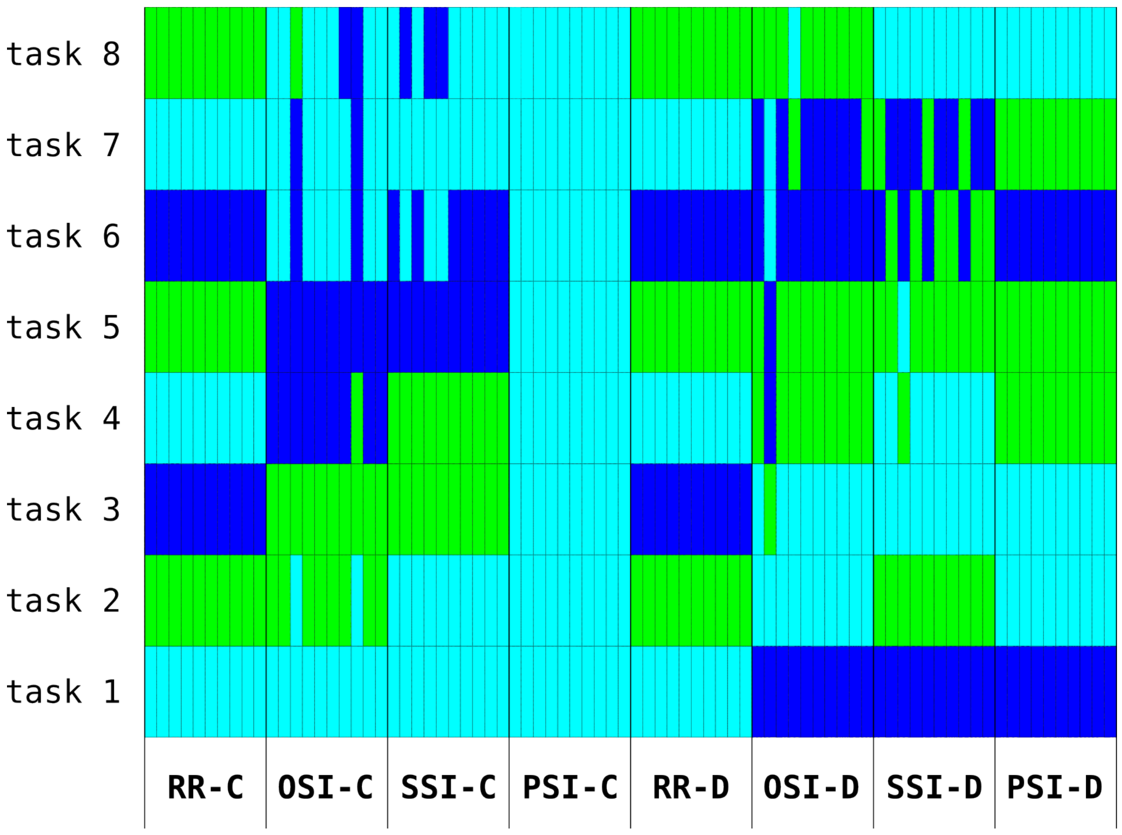


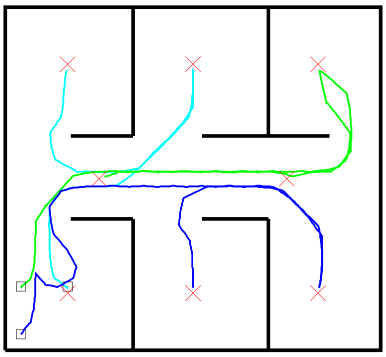
FIGURE 5.2: Allocation of tasks from Scenario 1 to robots over all trials. Each column represents an allocation for one trial, grouped by mechanism and starting locations. Each row represents a task. Colours indicate individual robots.

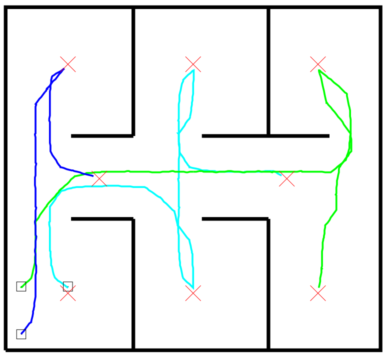
		Team Distance	Run time	Delib. time	Idle time	Delay time
Clustered	RR	35.39 ± 0.95	478 ± 82	**0.05 ± 0.0005**	251 ± 52	41 ± 27
	OSI	31.33 ± 0.95	**421 ± 18**	0.95 ± 0.06	**245 ± 56**	38 ± 23
	SSI	**28.65 ± 0.34**	435 ± 86	1.08 ± 0.06	249 ± 106	42 ± 23
	PSI	32.33 ± 2.07	1056 ± 82	0.22 ± 0.017	2113 ± 164	**0 ± 0**
Distributed	RR	44.68 ± 1.68	605 ± 28	**0.05 ± 0.003**	286 ± 84	82 ± 30
	OSI	21.08 ± 1.42	300 ± 27	0.89 ± 0.01	189 ± 46	**10 ± 8**
	SSI	**17.72 ± 0.35**	**229 ± 25**	0.98 ± 0.043	**107 ± 63**	12 ± 8
	PSI	25.64 ± 1.23	381 ± 39	0.23 ± 0.015	290 ± 71	15 ± 10

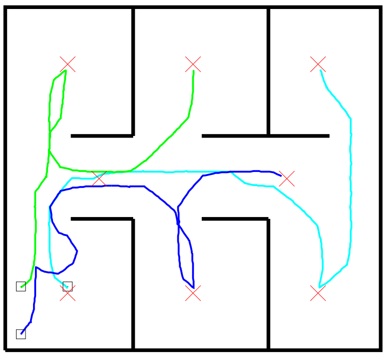
TABLE 5.1: Metrics for Scenario 1. Time is in seconds. Distance is in metres. The values given are means with 95% confidence intervals.

starting configuration, with two robots (robot 2 and robot 3) travelling no distance and reporting high idle time.

The next section discusses the most interesting of the results.


(a) RR


(b) OSI


(c) SSI

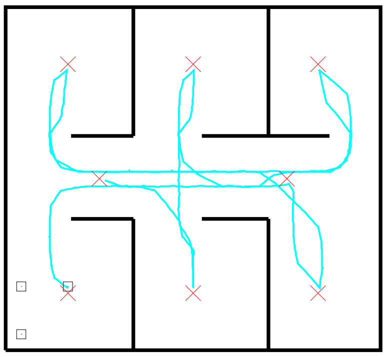

(d) PSI

FIGURE 5.3: Sample trajectories for SR-IT-SA Scenario 1 with clustered starting locations

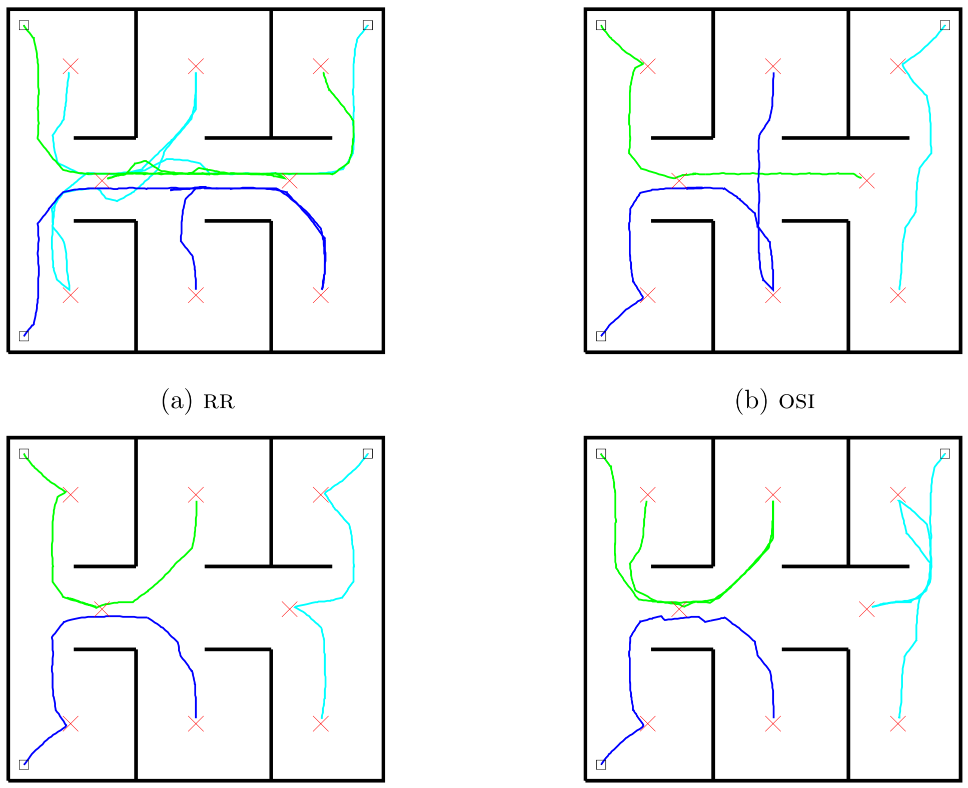
		Distance	Run time	Delib. time	Idle time	Delay time
Clustered	RR	40.23 ± 1.48	530 ± 38	**0.05 ± 0.0003**	234 ± 61	70 ± 19
	OSI	38.19 ± 1.21	513 ± 32	1.14 ± 0.07	**190 ± 62**	100 ± 72
	SSI	**36.20 ± 1.37**	**508 ± 48**	1.37 ± 0.12	250 ± 98	84 ± 21
	PSI	37.46 ± 2.14	1382 ± 140	0.25 ± 0.012	2764 ± 279	**0 ± 0**
Distributed	RR	40.51 ± 3.21	570 ± 49	**0.81 ± 0.5**	287 ± 89	65 ± 20
	OSI	26.23 ± 3.51	412 ± 48	0.96 ± 0.04	353 ± 47	12 ± 12
	SSI	**20.20 ± 1.20**	**375 ± 44**	1.12 ± 0.10	425 ± 123	**3 ± 4**
	PSI	29.22 ± 1.29	377 ± 25	0.25 ± 0.02	**127 ± 61**	13 ± 11

TABLE 5.2: Metrics for Scenario 2. Time is in seconds. Distance is in metres. The values given are means with 95% confidence intervals. Bold values are lower than those for other mechanisms, but not necessarily significantly.

5.4 Discussion

The analysis focuses on the comparative performance of the mechanisms. We start by considering the results for Scenario 1.

Overall the analysis supports the results of Tovey et al. [122], Lagoudakis et al. [57, 69, 70], and Koenig et al. [58], showing the effectiveness of the sequential single-item



(a) RR (b) OSI

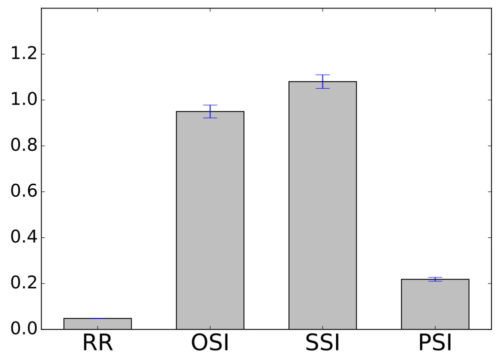
(c) SSI (d) PSI

FIGURE 5.4: Sample trajectories for SR-IT-SA Scenario 1 with distributed starting locations

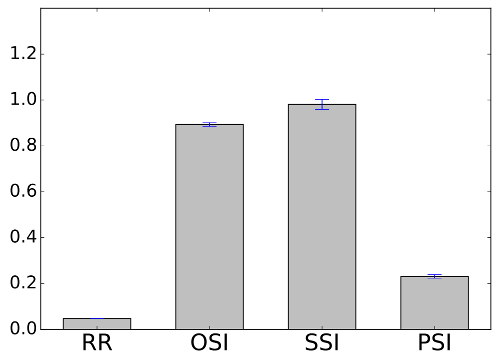
auction in finding solutions to the multi-robot routing problem when the overall distance covered (the MINISUM objective) is the most important performance metric. For both the clustered and distributed starting configurations, SSI generated solutions which required the team to travel the smallest overall combined distance, on average, by a significant amount (Figure 5.7). This means that the solutions generated by SSI were executed quickly in comparison to those generated by the other allocation mechanisms, though on average in the clustered case the SSI solutions take marginally longer to execute than the OSI allocations (Figure 5.7c).

The cost for this performance can be seen in the deliberation times. SSI, which requires bids from all robots for all unallocated tasks in every round, involves much more bidding than any of the other approaches, and this translates into the longest time spent in the allocation process (deliberation time). However, for the scenarios considered here, the deliberation times are all less than 1 percent of the total goal of executing the set of tasks (Figure 5.5).

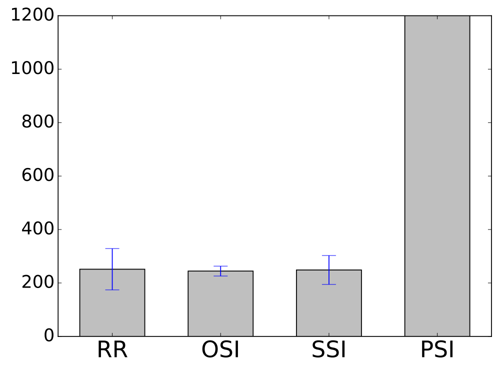
SSI also performs well in terms of idle time. An individual robot accumulates idle time when it finishes visiting its allocated tasks before other robots finish visiting theirs, so across the team it is a measure of wasted resource. For the clustered starting configuration, SSI outperforms OSI and RR but not significantly (though these results are dominated by the terrible performance of PSI on this metric). For the distributed starting



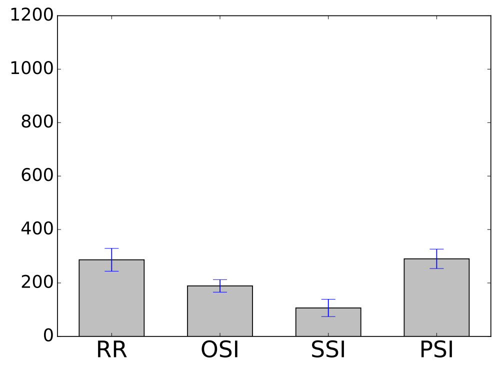
(a) Deliberation Time, Clustered



(b) Deliberation Time, Distributed



(c) Idle Time, Clustered



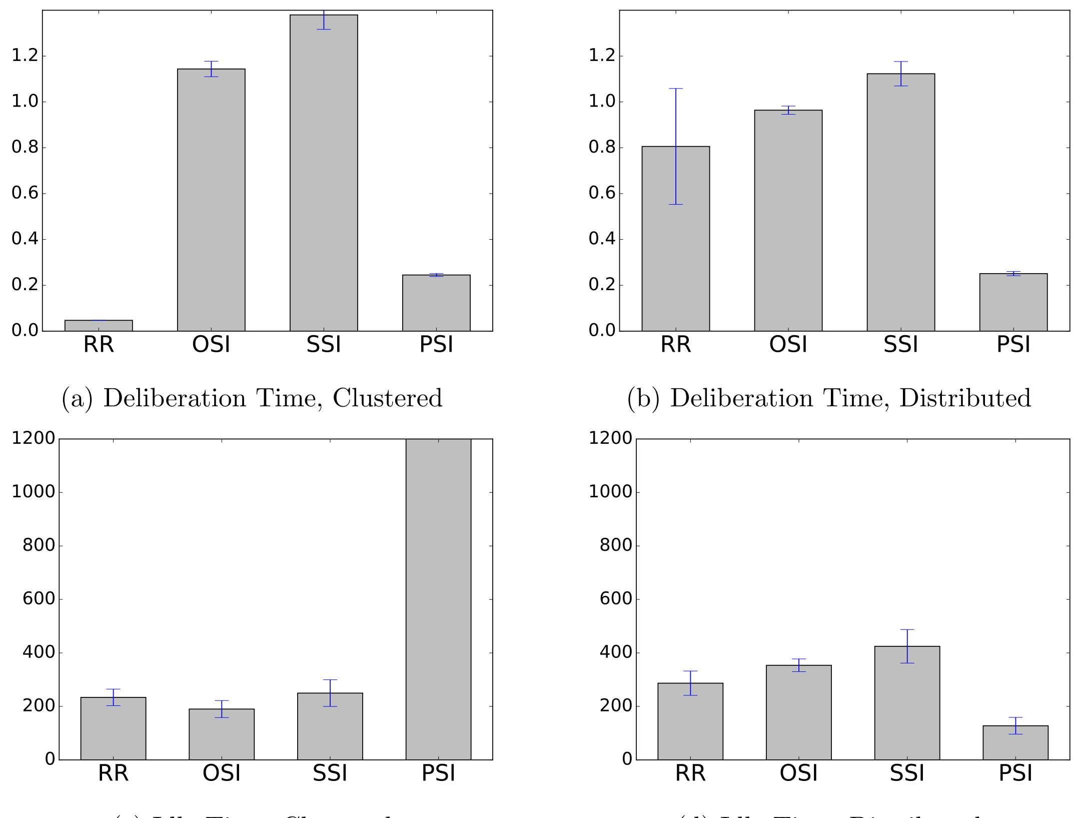
(d) Idle Time, Distributed

FIGURE 5.5: Deliberation Time and Idle Time for Scenario 1, with clustered (left) and distributed (right) starting locations. Time is given in seconds.

configuration, SSI has, on average, about half the idle time of the second-best performing mechanism.

Indeed, in terms of the metrics assessed, SSI can only be considered to have poor comparative performance in terms of near collisions in the case of the clustered starting configuration. The reason for this is the slightly higher number of near collisions that occur in SSI allocations for both clustered and distributed starting configurations and the consequent delay time (Figure 5.9). During an allocation with any of the mechanisms tested here, robots compute bids based on path plans independently with no knowledge of other robots' agendas. Thus, it can be difficult to predict inter-robot interference during the execution of tasks at the time of allocation, unless a method of conflict-free, joint path planning is employed, as in [129]. The increased delay time in this case is an order of magnitude below the run time, so is not a significant factor in the terms of task completion.

As Koenig et al. [58] point out, PSI can come up with arbitrarily poor allocations because it does not take synergies between task locations into account. As the results for the clustered start configuration show, it can also skew the distribution of tasks between robots — Figures 5.3d and 5.12a reveal that in the clustered start case, PSI allocates all of the tasks to one robot. This skew means that although PSI is not much worse than



(a) Deliberation Time, Clustered

(b) Deliberation Time, Distributed

(c) Idle Time, Clustered

(d) Idle Time, Distributed

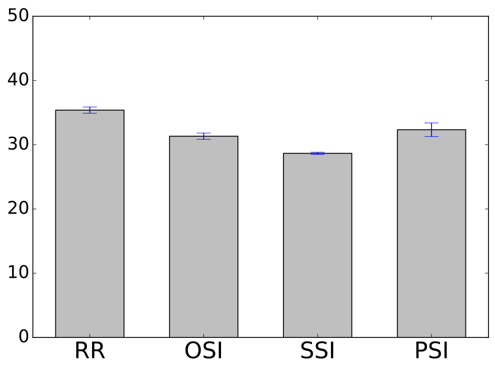
FIGURE 5.6: Deliberation Time and Idle Time for Scenario 2, with clustered (left) and distributed (right) starting locations. Time is given in seconds.

SSI or OSI on team distance, it is much worse on run time and on idle time.[1] Naturally, since PSI only allocates tasks to one robot, there are no near collisions and hence no delay time. Even on the distributed start configuration where this skew does not occur — as Figure 5.7b shows — PSI performs rather poorly, where team distance travelled is more than 40% greater than that travelled, on average, in a SSI allocation.

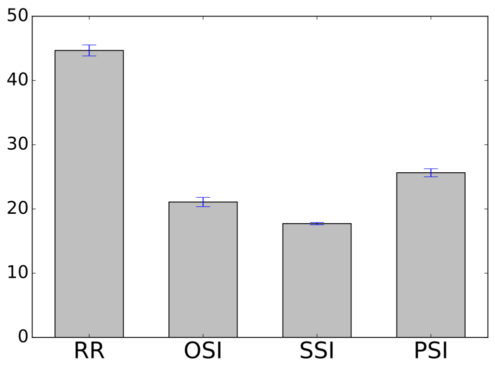
Turning to the results for Scenario 2, they largely agree with those from Scenario 1. SSI again produces good results, broadly outperforming the other mechanisms for the clustered starting configuration on all metrics except deliberation time. As in Scenario 1, PSI performs considerably worse than the other mechanisms (including the arbitrary allocations produced by RR) on run time (Figure 5.8a) and idle time (Figures 5.6c and 5.14c) when the starting locations are clustered.

One interesting result, however, is how well PSI performs on run time when starting locations are distributed. The run-time for PSI (377 seconds) is basically identical to that for SSI (375 seconds, Figure 5.8b) with a considerably lower idle time (127 versus 424 seconds, Figure 5.6d).

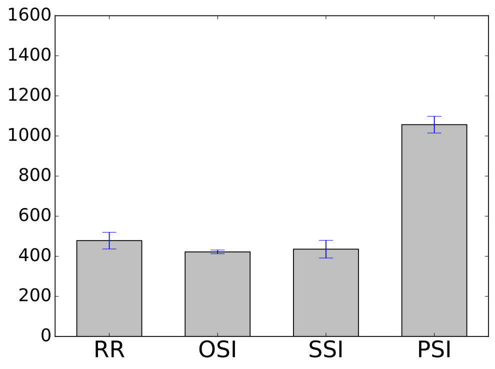
[1]The distance for PSI is 13% larger than that for OSI and 23% larger than that for SSI, but the runtime for PSI is 2.4 times that for SSI and 2.5 times that for OSI.



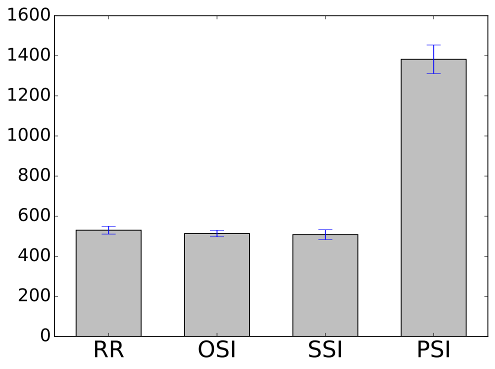
(a) Team Distance, Clustered



(b) Team Distance, Distributed



(c) Run Time, Clustered



(d) Run Time, Distributed

FIGURE 5.7: Team Distance and Run Time for Scenario 1, with clustered (left) and distributed (right) starting locations. Distance is given in metres. Time is given in seconds.

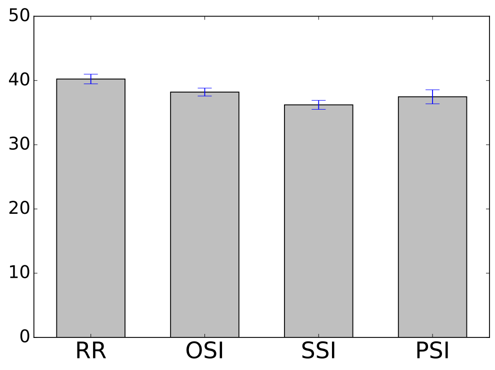
More evidence that PSI performs competitively with the other auction-based mechanisms from distributed starting locations can be found in results from a similar experiment in Appendix A.1.

Overall, the results tend to confirm the strong performance of the SSI auction mechanism in multi-robot routing tasks. The one area in which SSI performs worse than other mechanisms is in deliberation time, the time that it takes to allocate tasks to robots. For the scenarios considered here, the cost of allocating the tasks is negligible, with the deliberation time being less than 1 percent of the total time for completing the set of tasks. The total number of bids to allocate m tasks to n robots is:[2]

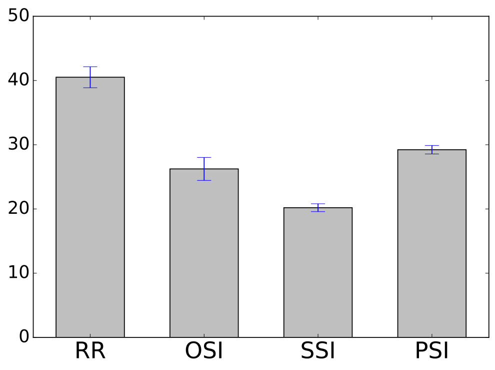
$$n\left(\frac{m(m+1)}{2}\right)$$

and it is conceivable that this could become big enough to be problematic. For example, consider the case of one hundred robots — as in the Centibots project [60] — which have to allocate 500 tasks. In such a case the SSI auction would require over 12 million bids, a 12,000-fold increase over what is required in these experiments, and enough to make

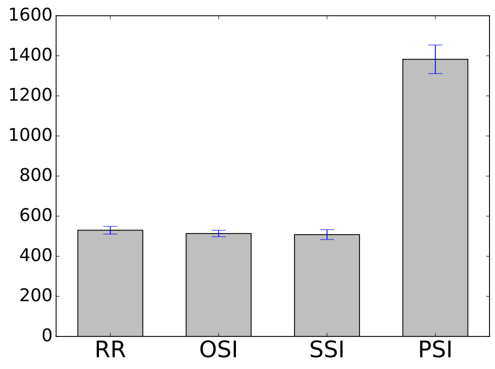
[2]One bid from each robot for m tasks in the first round, one bid from each robot for $m-1$ tasks in the second round, and so on.



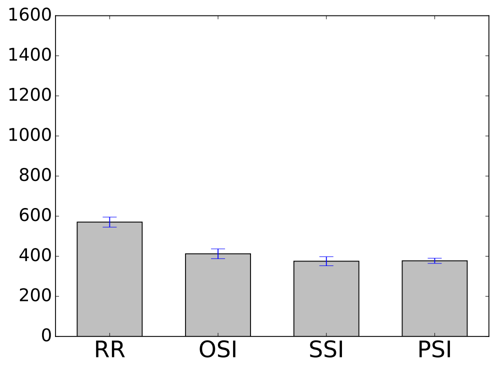
(a) Team Distance, Clustered



(b) Team Distance, Distributed



(c) Run Time, Clustered



(d) Run Time, Distributed

FIGURE 5.8: Team Distance and Run Time for Scenario 2, with clustered (left) and distributed (right) starting locations. Distance is given in metres. Time is given in seconds.

deliberation time a significant contributor to the time for task completion. In addition, since each bid has to be transmitted wirelessly — either to a centralised auctioneer, or to all other robots in a distributed auction — the number of messages can be a factor in robot deployments where communication bandwidth is limited [105].

In addition to communication, the cost of bid computation should also be considered. In the experiments reported here, the simple task insertion heuristic discussed in Section 5.2.1 makes the cost of computing a bid during a round of a SSI or OSI auction relatively inexpensive (it is the cost of a single path between a robot's *bid-from* location and a task location). A task insertion heuristic designed to optimise the MINISUM or MINIMAX objectives, such as the TSP insertion heuristic discussed by Lagoudakis et al. [70], would make the cost of every bid more expensive the more tasks a robot wins. While this may increase the quality of a solution for a distance-based objective, the increased cost of bid computation also increases the time it takes to compute a solution, which also affects the scalability of such SSI implementations.

Both of these aspects might make the OSI or PSI mechanisms worth considering for larger deployments. The OSI auction results in distances that are 10% (clustered start) to 20% (distributed start) worse than SSI in terms of distance to the tasks in Scenario 1, and for the Centibots example would require several hundred times fewer

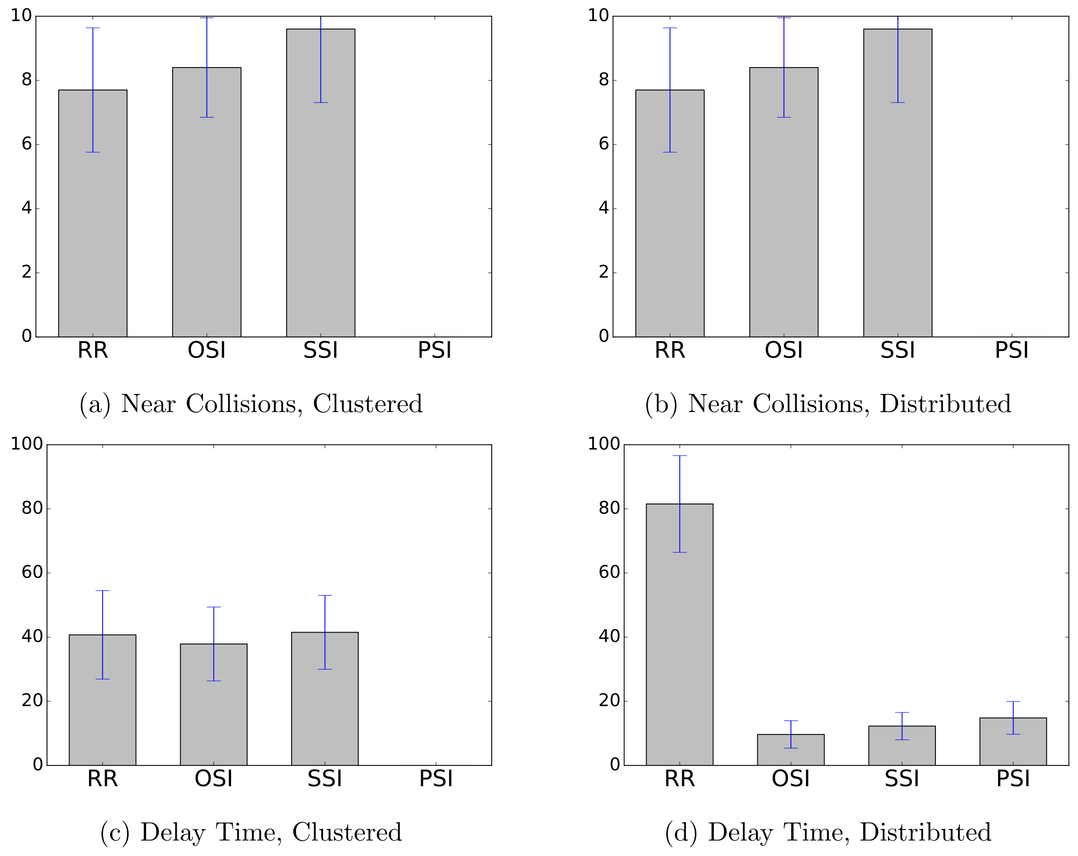
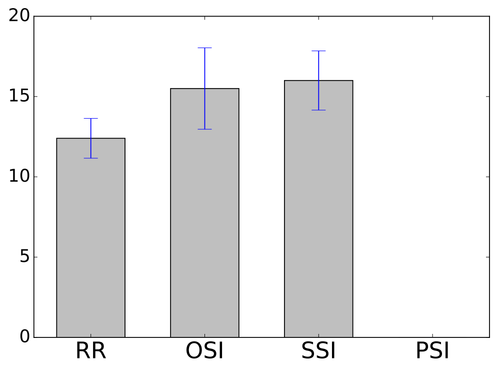


Figure 5.9: Near Collisions and Delay Time for Scenario 1, with clustered (left) and distributed (right) starting locations. Time is given in seconds.

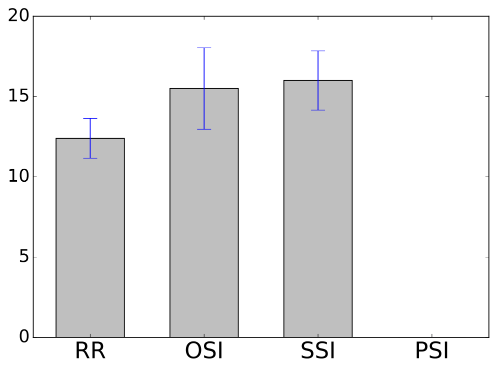
bids (50, 000 rather than 12 million). The PSI auction is about 40% worse in terms of total distance for the distributed case with Scenario 1, but could do the task allocation for the Centibots with just 500 bids (one for each task). In Scenario 1 with the clustered starting configuration, as pointed out above, PSI is only about 23% worse than SSI in terms of distance travelled, but the long run time that results from the skewed allocation needs to be taken into account. This point is reinforced by the results for Scenario 2, which show that there are situations in which PSI can equal or outperform SSI (at least this implementation of it) on some metrics. However, while this result is encouraging in this respect, an analysis of more scenarios would be required before reaching any firm conclusions about whether PSI can predictably outperform SSI.

5.5 Summary

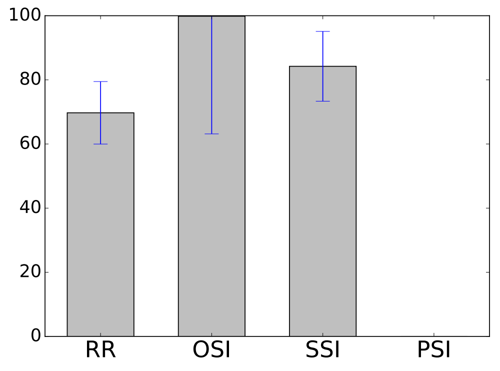
The experiments discussed in this chapter have studied the performance of a number of task allocation mechanisms on a version of the multi-robot routing problem in which tasks are independently executed by single robots (SR-IT-SA). The missions were carried out in simulation. The experiments discussed in this chapter do not attempt to characterise the SR-IT-SA environment as a whole, which includes general multi-robot routing problems. The aim is to investigate factors like inter-robot interference (near



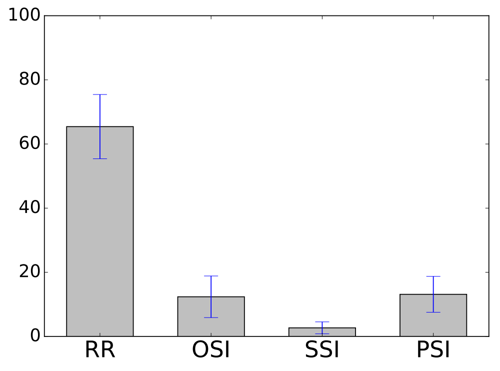
(a) Near Collisions, Clustered



(b) Near Collisions, Distributed



(c) Delay Time, Clustered

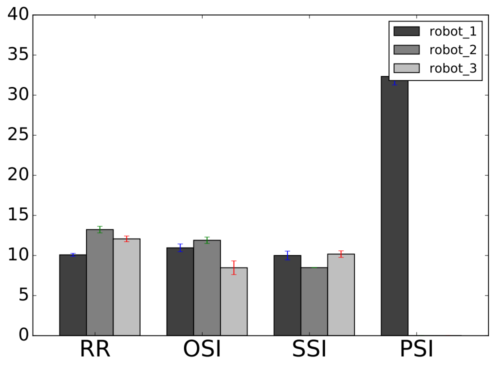


(d) Delay Time, Distributed

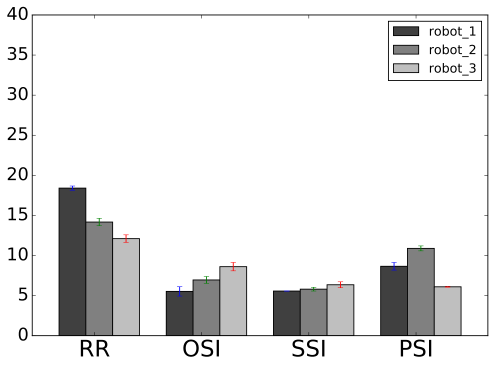
Figure 5.10: Near Collisions and Delay Time for Scenario 2, with clustered (left) and distributed (right) starting locations. Time is given in seconds.

collisions and delay time) that can complicate theoretical predictions of mechanism performance when applied to a realistic multi-robot system even in a simple setting. Chapter 8 presents a more systematic investigation of multi-robot routing missions in the SR-IT-SA environment.

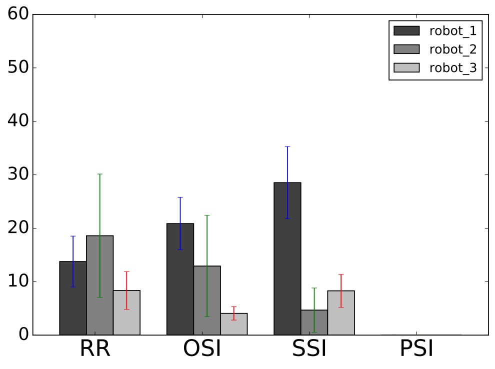
The main result is that the sequential single item (SSI) auction broadly outperforms other single-item auctions and parallel auctions across our range of metrics, though it does not perform best on all of them for all scenarios. However, there do seem to be trade-offs, especially in terms of the total number of bids required by the mechanisms. This suggests that the SSI auction might have issues with scaling to larger routing problems than we study here, especially if communication bandwidth is restricted. In other words, the high performance of SSI comes at a cost that might be hard to pay for some missions. Other mechanisms tested, which can scale better, might be preferable on such scenarios despite their poorer performance otherwise. The next chapter examines the performance of the same four task allocation mechanisms as they are employed in both SR-IT-SA and SR-IT-DA (dynamic allocation) environments.



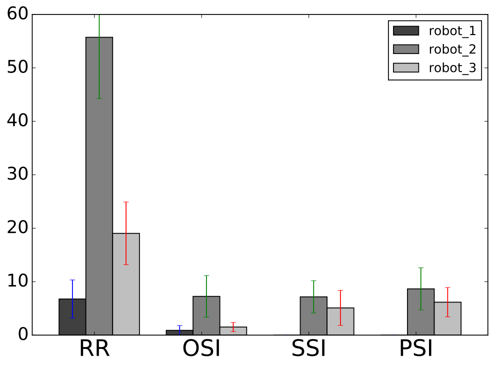
(a) Distance, Clustered



(b) Distance, Distributed

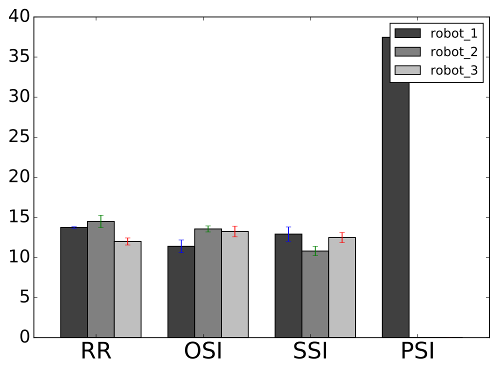


(c) Delay Time, Clustered

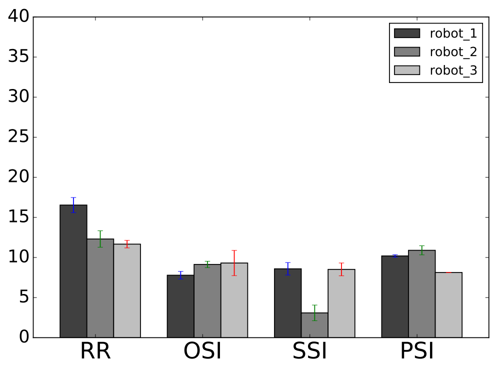


(d) Delay Time, Distributed

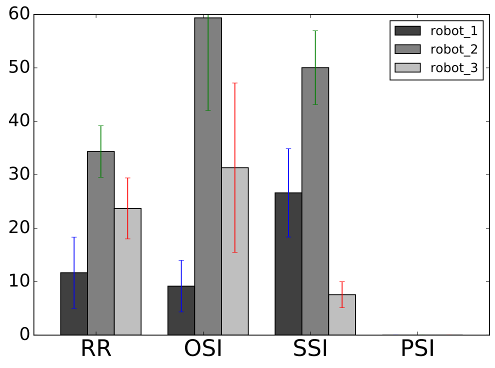
FIGURE 5.11: Distance Travelled and Delay Time for each robot in Scenario 1 with starting locations *clustered* (left) and *distributed* (right). Results are grouped by mechanism and by robot.



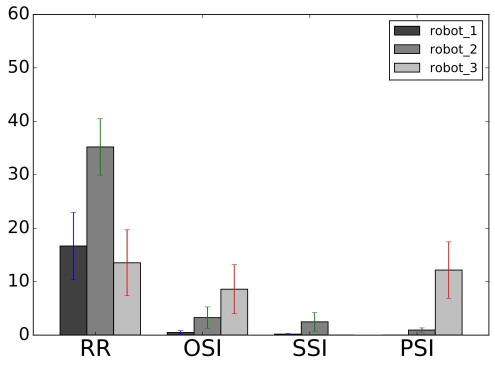
(a) Distance, Clustered



(b) Distance, Distributed

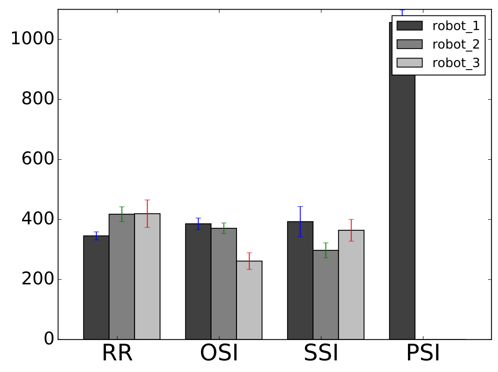


(c) Delay Time, Clustered

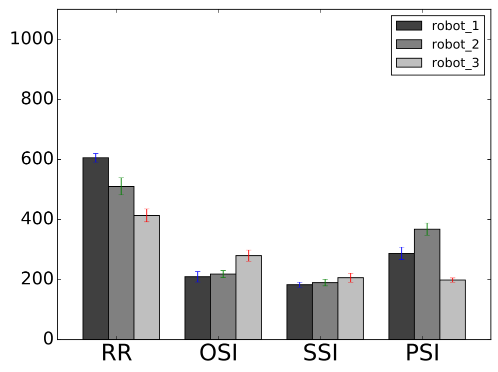


(d) Delay Time, Distributed

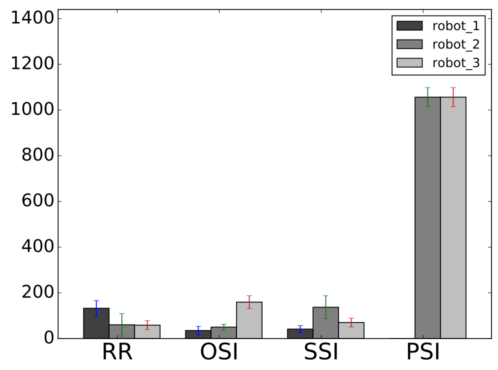
FIGURE 5.12: Distance Travelled and Delay Time for each robot in Scenario 2 with starting locations *clustered* (left) and *distributed* (right). Results are grouped by mechanism and by robot.



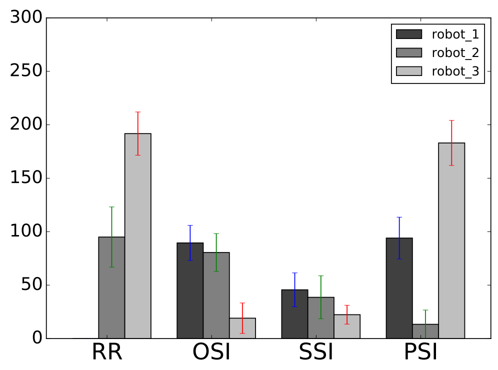
(a) Movement Time, Clustered



(b) Movement Time, Distributed

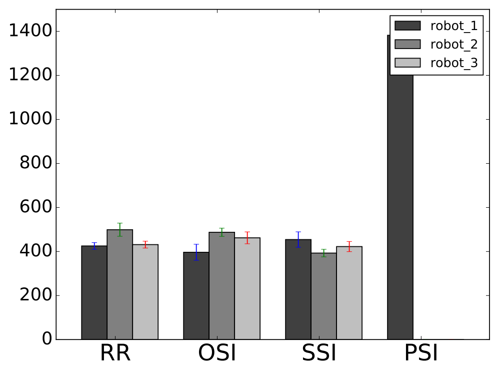


(c) Idle Time, Clustered

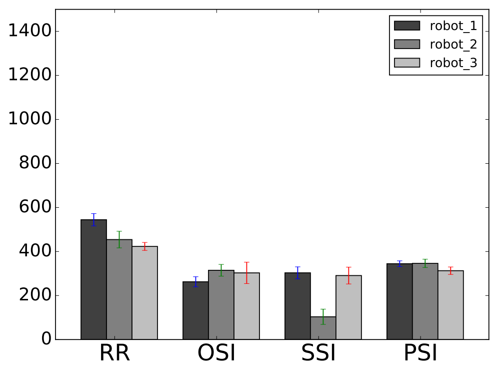


(d) Idle Time, Distributed

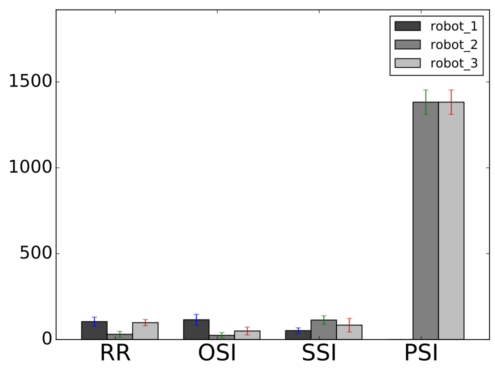
FIGURE 5.13: Travel and Idle time for each robot in Scenario 1 with starting locations *clustered* (left) and *distributed* (right). Results are grouped by mechanism and by robot.



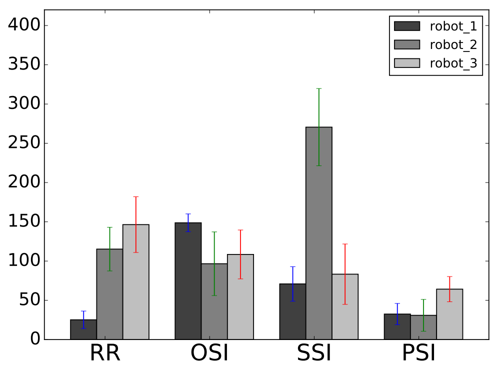
(a) Movement Time, Clustered



(b) Movement Time, Distributed



(c) Idle Time, Clustered



(d) Idle Time, Distributed

FIGURE 5.14: Travel and Idle time for each robot in Scenario 2 with starting locations *clustered* (left) and *distributed* (right). Results are grouped by mechanism and by robot.

Chapter 6

A Dynamic Task Environment (SR-IT-DA)

6.1 Introduction

This chapter investigates how the performance of task allocation mechanisms varies when the mechanisms are employed across different task environments. Experiments are presented that compare the performance of the four task allocation mechanisms defined in Section 4.4 on multi-robot routing problems set in two different task environments: a static environment (*SR-IT-SA*), in which all tasks can be allocated at the beginning of a mission, and a dynamic environment (*SR-IT-DA*), in which tasks arrive over time. In both task environments, tasks are performed by single robots in any order they choose after they are allocated. Section 6.2 describes the design of the experiments, Section 6.3 presents their results, Section 6.4 discusses the significance of the results, and Section 6.5 concludes.

The experiments and a portion of the results presented in this chapter were published in Schneider et al. (2015) [110].

6.2 Experiments

6.2.1 Mechanisms Tested

The task allocation mechanisms discussed in Section 4.4 were tested in these experiments:

- Round-robin (RR)
- Ordered single-item (OSI)
- Sequential-single item (SSI)
- Parallel single item (PSI)

6.2.2 Metrics

Each experiment recorded some of the metrics described in Section 4.6:

- *Team Distance* – The sum of the distances travelled by all robots.
- *Deliberation time* – The time taken for the tasks to be allocated to the robots.
- *Run time* – The time between the start of an experiment and the time the last robot on the team completes the tasks allocated to it. This includes deliberation time.
- *Near collisions* – The number of times two robots travelled close enough to detect a risk of collision and triggered a negotiation to avoid colliding.
- *Delay time* – The time robots spent replanning and yielding the right of way while negotiating to avoid a collision; and
- *Idle time* – The time that elapses between when a robot completes its last task and when all robots on the team have completed all of their assigned tasks

6.2.3 Platform

Experiments were conducted with the MRTeAm framework (Section 3) using the Stage simulator [37]. The simulated robots are modelled on the Turtlebot 2[1] robot described in Section 3.5 and have the same characteristics as their physical counterparts (size, shape, acceleration, and maximum speed).

6.2.4 Experimental Setup

Two versions of a scenario pictured in Figure 6.1 are investigated with $m = 9$ tasks in the same locations. The map in which the scenarios are set is 8×6 metres, with rooms and corridors arranged as shown in Figure 6.1. In the *static scenario* (SA), all tasks arrive at the beginning of an experiment and can be allocated to the robots in a single deliberation phase, followed by a single execution phase.

In the *dynamic scenario* (DA), the value of each task's arrival time, $t.arr$, is shown in Figure 6.1. Arrival of a task triggers a deliberation phase followed by an execution phase (Figure 4.5). If a task arrives while robots are currently executing tasks awarded in a prior deliberation phase, the current execution phase is aborted and a new deliberation phase begins, in which the newly arrived tasks are allocated. Robots retain tasks they have been awarded between deliberation and execution phases.

In the dynamic scenario investigated here, tasks arrive in pairs on a fixed schedule at 45-second intervals. For example, tasks 1 and 8 arrive at time 0 and can be allocated at the beginning of an experiment. Tasks 3 and 4 arrive after 45 seconds have elapsed,

[1]`http://www.turtlebot.com/turtlebot2/`

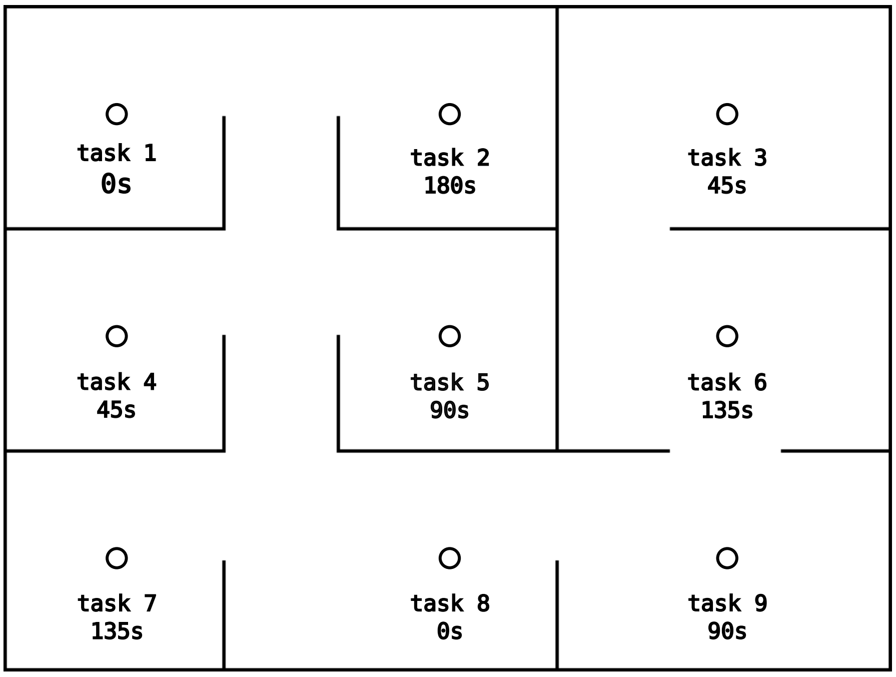
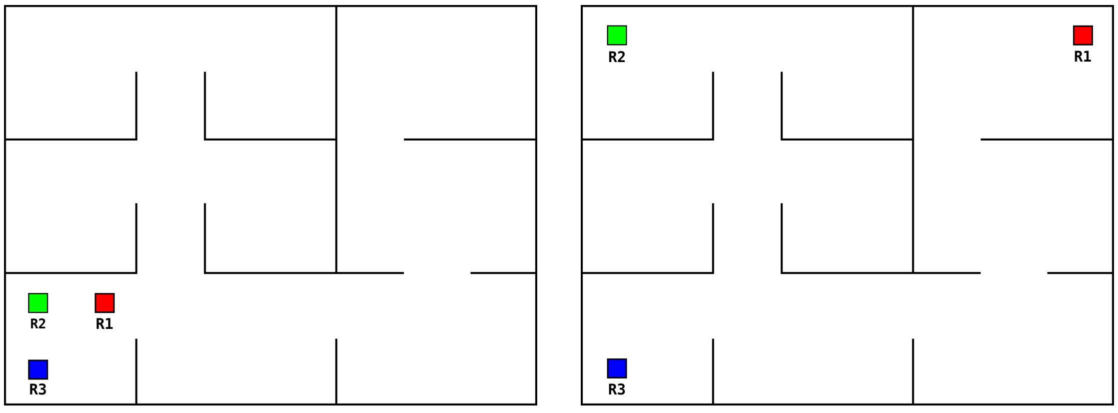


Figure 6.1: The scenario tested, with two variations. In the static scenario (SA), tasks are all allocated at the beginning of an experiment. In the dynamic scenario (DA), tasks arrive and become available for allocation at the times indicated in the figure.



(a) Clustered (b) Distributed

Figure 6.2: Robot starting locations

triggering a second deliberation phase, and so on. The dynamic scenario is loosely based on one used in a study of task complexity in human-robot teams [89].

The size of the robot team is fixed at $n = 3$ as in the experiments in Chapter 5. Two sets of robot starting configurations are used, *clustered* (Figure 6.2a) and *distributed* (Figure 6.2b). These starting configurations are similar to those used in experiments in Chapter 5, but the map used here has a different size and configuration.

Experiments were conducted with each of the two scenarios and two sets of starting configurations using each of the four task allocation mechanisms, and each was run 10

		Team Distance	Run time	Delib. time	Idle time
Clustered	RR	43.92 ± 1.12	262.62 ± 9.03	**0.003** ± 0.0003	118.69 ± 20.95
	OSI	42.17 ± 1.23	238.17 ± 11.55	5.08 ± 0.05	70.87 ± 15.73
	SSI	**32.30** ± 0.26	**181.95** ± 7.08	7.87 ± 0.05	**58.61** ± 10.42
	PSI	34.51 ± 0.24	405.74 ± 7.98	1.14 ± 0.03	389.80 ± 8.02
Distributed	RR	46.80 ± 0.39	233.77 ± 9.83	**0.004** ± 0.0006	131.19 ± 16.49
	OSI	23.81 ± 0.15	138.93 ± 5.31	4.20 ± 0.09	97.62 ± 11.36
	SSI	24.23 ± 0.20	**137.84** ± 5.41	6.43 ± 0.05	**77.70** ± 10.78
	PSI	**22.57** ± 0.15	169.12 ± 4.65	1.34 ± 0.02	207.94 ± 9.42

		Team Distance	Run time	Delib. time	Idle time
Clustered	RR	57.03 ± 0.93	234.65 ± 9.15	**0.0153** ± 0.001	**123.31** ± 18.69
	OSI	37.52 ± 3.64	**211.79** ± 1.39	9.06 ± 0.34	125.56 ± 6.92
	SSI	**35.78** ± 0.16	212.88 ± 1.61	9.38 ± 0.33	125.42 ± 3.08
	PSI	43.23 ± 3.26	220.62 ± 12.75	6.11 ± 0.18	127.4 ± 37.57
Distributed	RR	69.7 ± 1.69	233.54 ± 8.72	**0.0143** ± 0.0007	101.04 ± 9.54
	OSI	28.2 ± 0.14	204.72 ± 2.43	9.17 ± 0.22	89.88 ± 4.42
	SSI	28.25 ± 0.12	204.0 ± 2.04	9.35 ± 0.24	**88.45** ± 4.48
	PSI	**28.14** ± 0.26	**203.43** ± 0.52	6.05 ± 0.14	89.21 ± 2.08

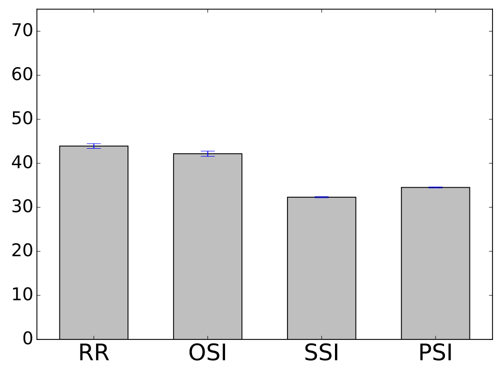
Table 6.1: Team metrics for the static (top) and dynamic (bottom) scenario allocations. Distance is given in metres. Time is given in seconds. The values given are means with 95% confidence intervals.

times. 160 experimental trials were conducted in all:

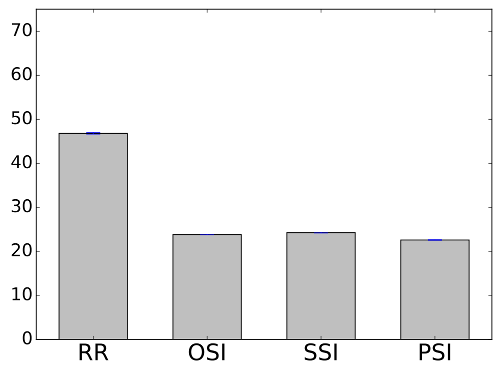
$$160 = 4\ \textit{missions}\ (\{\textit{clustered}\,,\textit{distributed}\} \times \{\textit{static}\,,\textit{dynamic}\}) \\ \times 4\ \textit{allocation mechanisms} \times 10\ \textit{trials}$$

6.3 Results

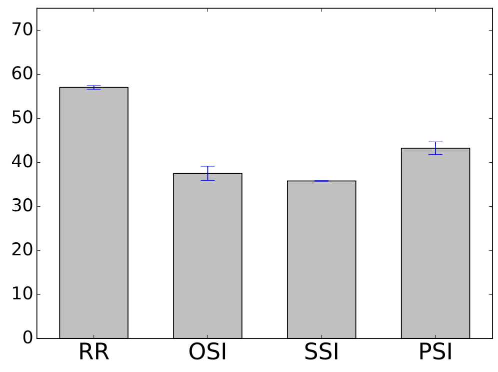
The results of the experiments can be seen in Table 6.1 and Figures 6.4–6.12. Table 6.1 gives the value of the metrics for each of the four task allocation mechanisms — RR, OSI, SSI and PSI in each of the four missions of static and dynamic scenarios with clustered and distributed starting configurations. The table gives average values across the 10 runs with 95% confidence intervals. Figure 6.3 gives average distances travelled by the team. This is one metric considered here since it is the one that SSI is looking to optimise task allocations against (the implementation of SSI used in these experiments attempts to optimise the MiniSum objective) and it is the metric that we might expect it to perform best on. The other main metric is run time (Figure 6.4), another important measure of performance. Team distance does not necessarily give an indication of run time. Note a result from Chapter 5, shown in Figure 5.8, where a PSI allocation was competitive with the other mechanisms in terms of team distance, but its run time was



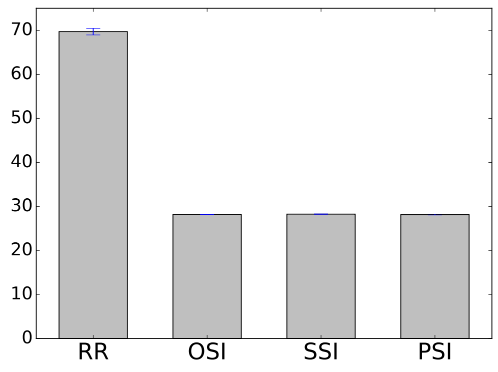
(a) Team Distance, Clustered (SA)



(b) Team Distance, Distributed (SA)



(c) Team Distance, Clustered (DA)



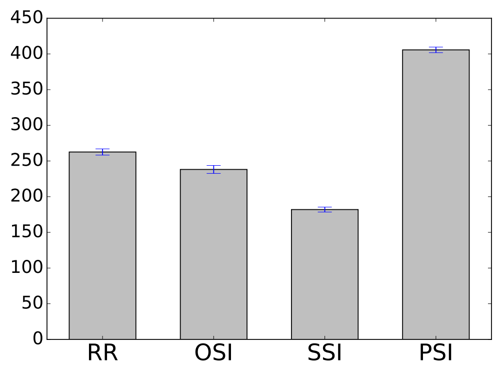
(d) Team Distance, Distributed (DA)

FIGURE 6.3: Team distances for clustered (left) and distributed (right) starting configurations in the static (top) and dynamic (bottom) scenarios.

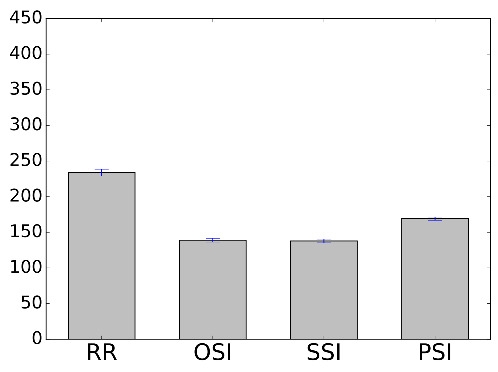
almost three times greater, on average. (Run time is more proportional to the maximum robot distance travelled by any robot during a mission, also known as the *makespan*). Figures 6.5–6.8 show the remaining metrics.

Figures 6.9–6.12 are timeline plots that show robot activity over the course of an experiment in each of the experimental configurations. Each figure plots the activity of robots in the team recorded during a single example trial. These figures help clarify the time-based metrics that we measure (run time, deliberation time, idle time, and delay time) and provide a way to inspect each robot's activity in parallel with its team mates as an experiment unfolds.

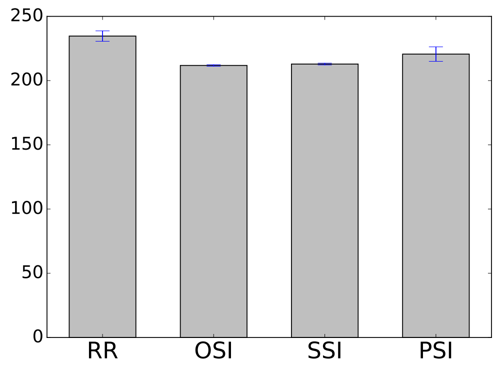
In the static scenario, the results for team distance (Figure 6.3) show that SSI performs as we expect it to given the analysis in [58] and the results from experiments with similar SR-IT-SA scenarios in Chapter 5. From the clustered configuration, SSI generates allocations that result in shorter total distances for the team than any of the other mechanisms (Figure 6.3a). The results also show that SSI allocations yield lower run time (Figure 6.4a), execution phase time (Figure 6.5a), and idle time (Figure 6.7a). Run time and execution time can be complicated by inter-robot interference. In this regard, SSI allocations also resulted in paths with fewer near collisions than any mechanism besides



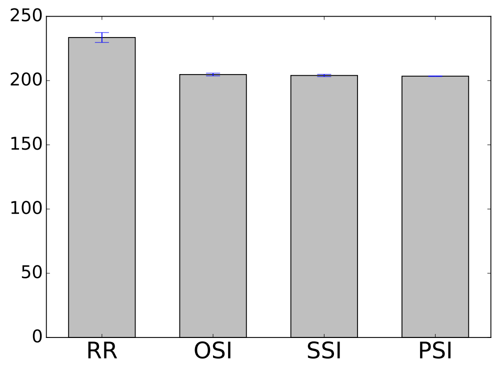
(a) Run Time, Clustered (SA)



(b) Run Time, Distributed (SA)



(c) Run Time, Clustered (DA)



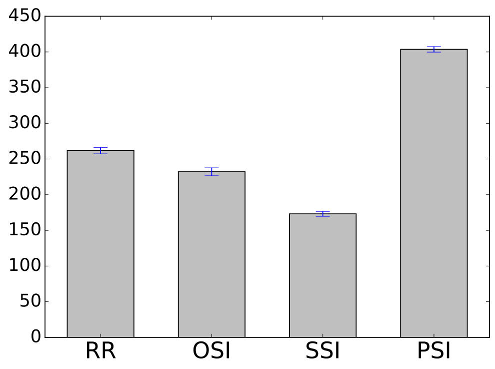
(d) Run Time, Distributed (DA)

FIGURE 6.4: Run times for clustered (left) and distributed (right) starting configurations in the static (top) and dynamic (bottom) scenarios.

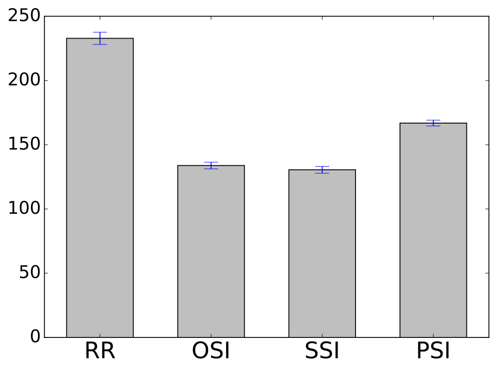
PSI and correspondingly low delay time (Figure 6.8a). Low idle time (Figure 6.7a) suggests a more even allocation of tasks to robots than other mechanisms. PSI is notably poor in this regard in the static scenation from the clustered starting configuration, a point noted by Koenig et al. [58].

The timelines in Figure 6.9 illustrate SSI's performance in these regards visually. Especially clear is the imbalance in allocation between SSI and PSI, which allocated most of the tasks to a single robot (Figure 6.9d). Also apparent is that the OSI allocation in this trial led to more inter-robot interference (Figure 6.9b). A number of near collisions were detected by robot 3 as it attempted to leave the starting area and execute its tasks in the first 60 seconds of the trial. Note that timelines are plotted from individual trials and so don't show average performance across all trials.

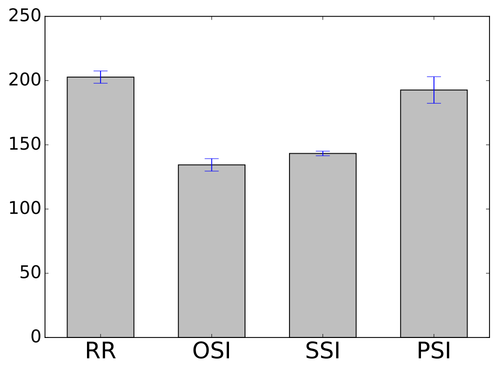
When moving to consider the static scenario with the distributed starting configuration, note from Figure 6.3b that SSI doesn't provide an obviously better allocation in terms of team distance than either OSI or PSI. In fact, Table 6.1 reveals that SSI does fractionally worse than OSI. SSI also does not produce significantly lower run times than OSI on average (Figure 6.4b). SSI however, continues to perform at least as well as the other mechanisms on other metrics, barring its higher deliberation time (Figure 6.6b).



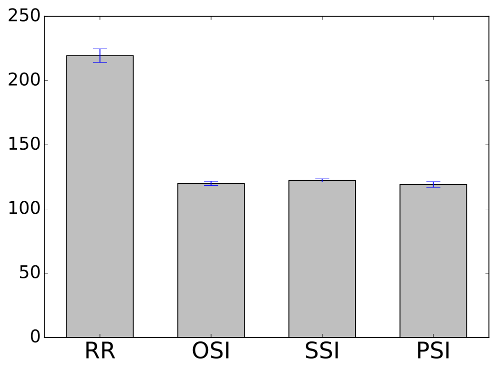
(a) Execution Phase Time, Clustered (SA)



(b) Execution Phase Time, Distributed (SA)



(c) Execution Phase Time, Clustered (DA)

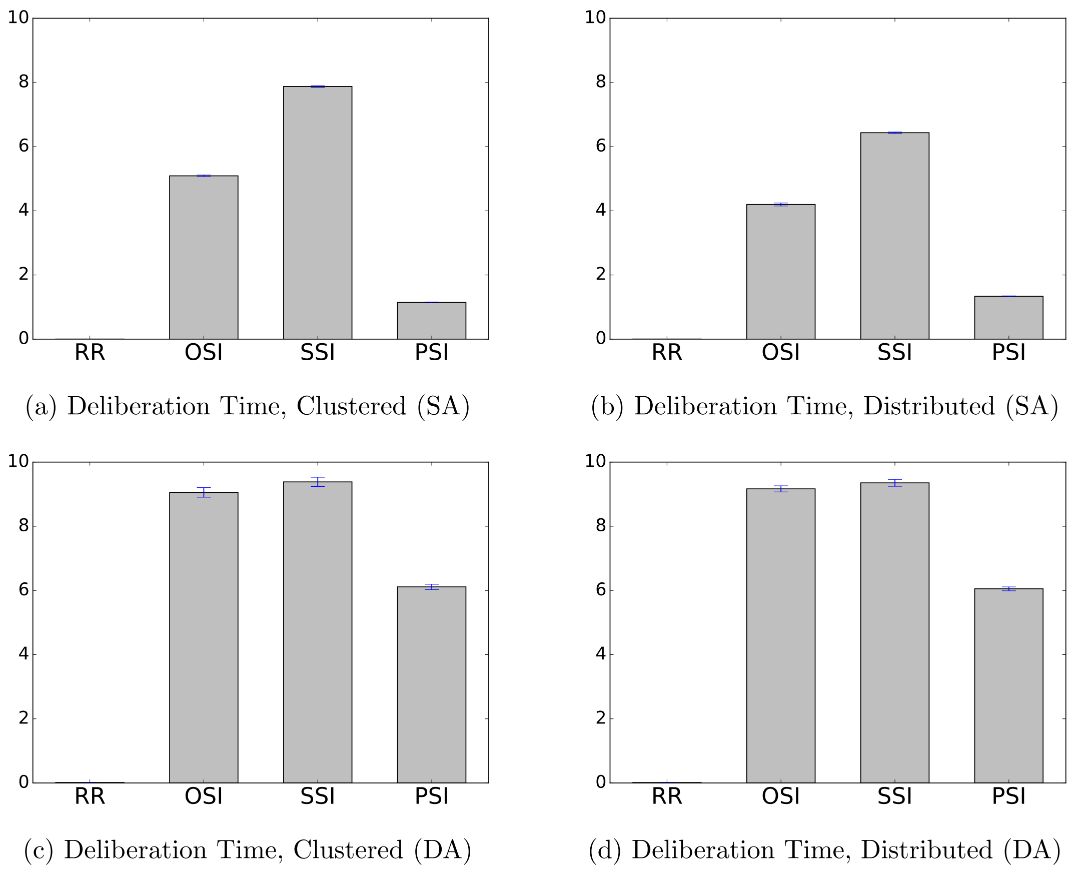


(d) Execution Phase Time, Distributed (DA)

FIGURE 6.5: Execution times for clustered (left) and distributed (right) starting configurations in the static (top) and dynamic (bottom) scenarios.

Moving from the static (SA) to the dynamic (DA) scenario, in terms of team distance (Figure 6.3c), SSI again produces allocations that leads to the smallest team distances, but its average run time (212.88 seconds) is actually slightly higher than OSI's run time (211.79 seconds) (Figure 6.4c). In the dynamic scenario, allocation spreads the robots out in time much the same way as the distributed start locations do in space — because they start moving to the locations of tasks allocated in the first deliberation phase (at time 0), the robots are physically spread out by the time that later tasks are allocated. In the dynamic scenario with the distributed starting configuration in particular, all three auction mechanisms outperform RR about equally, and across other metrics including team distance (Figure 6.3d), run time (Figure 6.4c), and idle time (Figure 6.7d).

Additional evidence that PSI performs competitively with the other auction-based mechanisms from distributed starting locations in a SR-IT-DA task environment can be found in results from a similar experiment carried out on physical robots given in Appendix A.2.



(a) Deliberation Time, Clustered (SA)

(b) Deliberation Time, Distributed (SA)

(c) Deliberation Time, Clustered (DA)

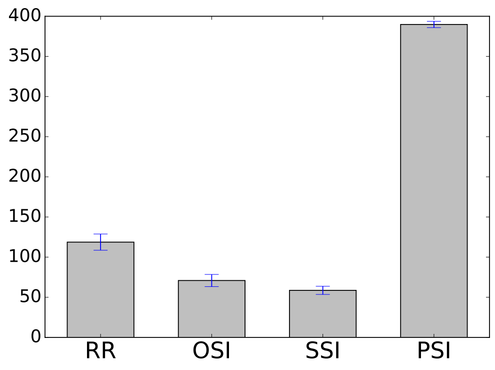
(d) Deliberation Time, Distributed (DA)

FIGURE 6.6: Deliberation time for clustered (left) and distributed (right) starting configurations in the static (top) and dynamic (bottom) scenarios.

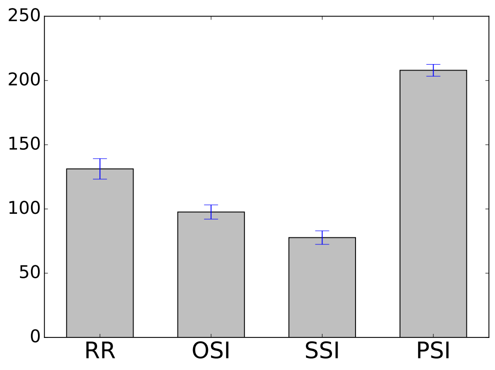
6.4 Discussion

The aim of the experiments presented in this chapter is to show how the performance of mechanisms varies when they are employed *across* task environments (from SA to DA), in addition to comparing how mechanisms perform within a single environment, as in Chapter 5. When considering team distance as a measure of performance, the results show that across task environments, SSI consistently produced the lowest team distance of any mechanism from the clustered starting configuration in both the static (Figure 6.3a) and dynamic (Figure 6.3c) scenarios. However, the relative performance of OSI and PSI in terms of team distance did change when moving across task environments. In the static scenario with the clustered starting configuration, PSI produced a lower team distance on average (34.51 seconds) than OSI (Figure 6.3a). When the mechanisms were employed in the dynamic scenario with the same starting configuration, relative performance was reversed, with OSI producing a lower team distance on average (37.52 seconds) than PSI (43.23 seconds).

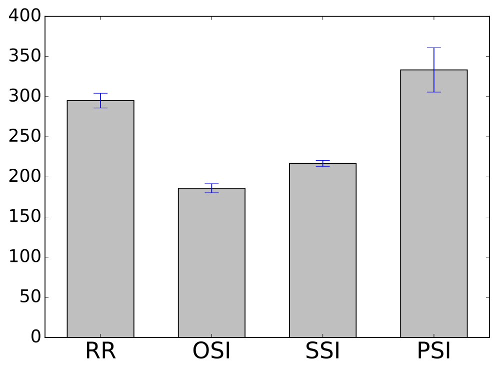
A similar effect can be seen when run time is considered (Figure 6.4). In the static scenario, SSI produced the lowest run time from the clustered starting configuration (Figure 6.4a), but in the dynamic scenario its run time was not significantly better than OSI's (Figure 6.4c). Similarly for execution phase time (as a component of run time),



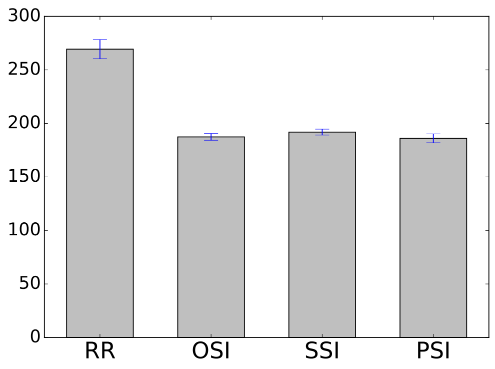
(a) Idle Time, Clustered (SA)



(b) Idle Time, Distributed (SA)



(c) Idle Time, Clustered (DA)



(d) Idle Time, Distributed (DA)

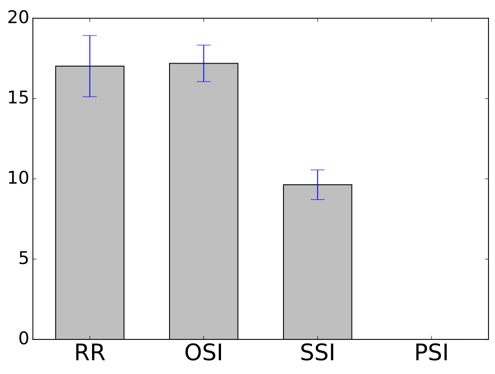
FIGURE 6.7: Idle time for clustered (left) and distributed (right) starting configurations in the static (top) and dynamic (bottom) scenarios.

the relative performance between SSI and OSI was reversed when moving from the static scenario (Figure 6.5a) to the dynamic scenario (Figure 6.5c). As with team distance, the relative performance differences between the auction-based mechanisms in terms of run time and execution phase time was diminished or removed when moving from the clustered starting configuration to the distributed starting configuration (Figures 6.4d and 6.5d).

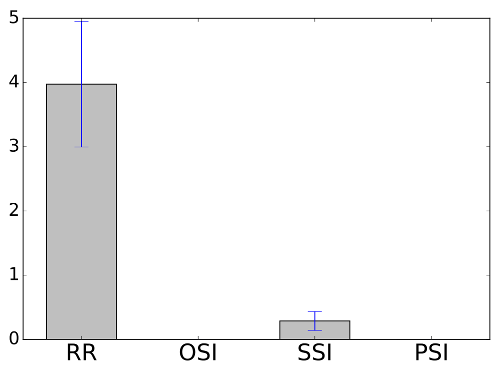
An observation that can be made from these results is that not only can the performance of a single task allocation mechanism vary as as it is employed across task environments, but the *relative* performance of a group of mechanisms can change as well. This idea is explored more fully in the next chapter.

6.5 Summary

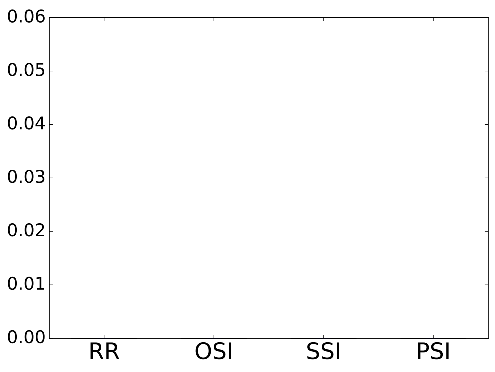
This chapter has presented experiments that investigate the performance of a number of task allocation mechanisms on two examples of a multi-robot routing problem set in different task environments: a static environment (SR-IT-SA), in which all tasks could be allocated at the beginning of a mission, and a dynamic environment (SR-IT-DA), in which tasks arrived over time. The results show that not only can the



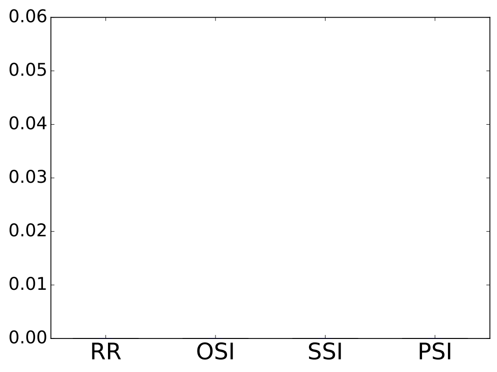
(a) Delay Time, Clustered (SA)



(b) Delay Time, Distributed (SA)



(c) Delay Time, Clustered (DA)



(d) Delay Time, Distributed (DA)

FIGURE 6.8: Delay time for clustered (left) and distributed (right) starting configurations in the static (top) and dynamic (bottom) scenarios.

performance of a single mechanism vary across task environments, but also the relative performance among the group of mechanisms. The next chapter explores the idea of relative performance ranking of mechanisms in other task environments.

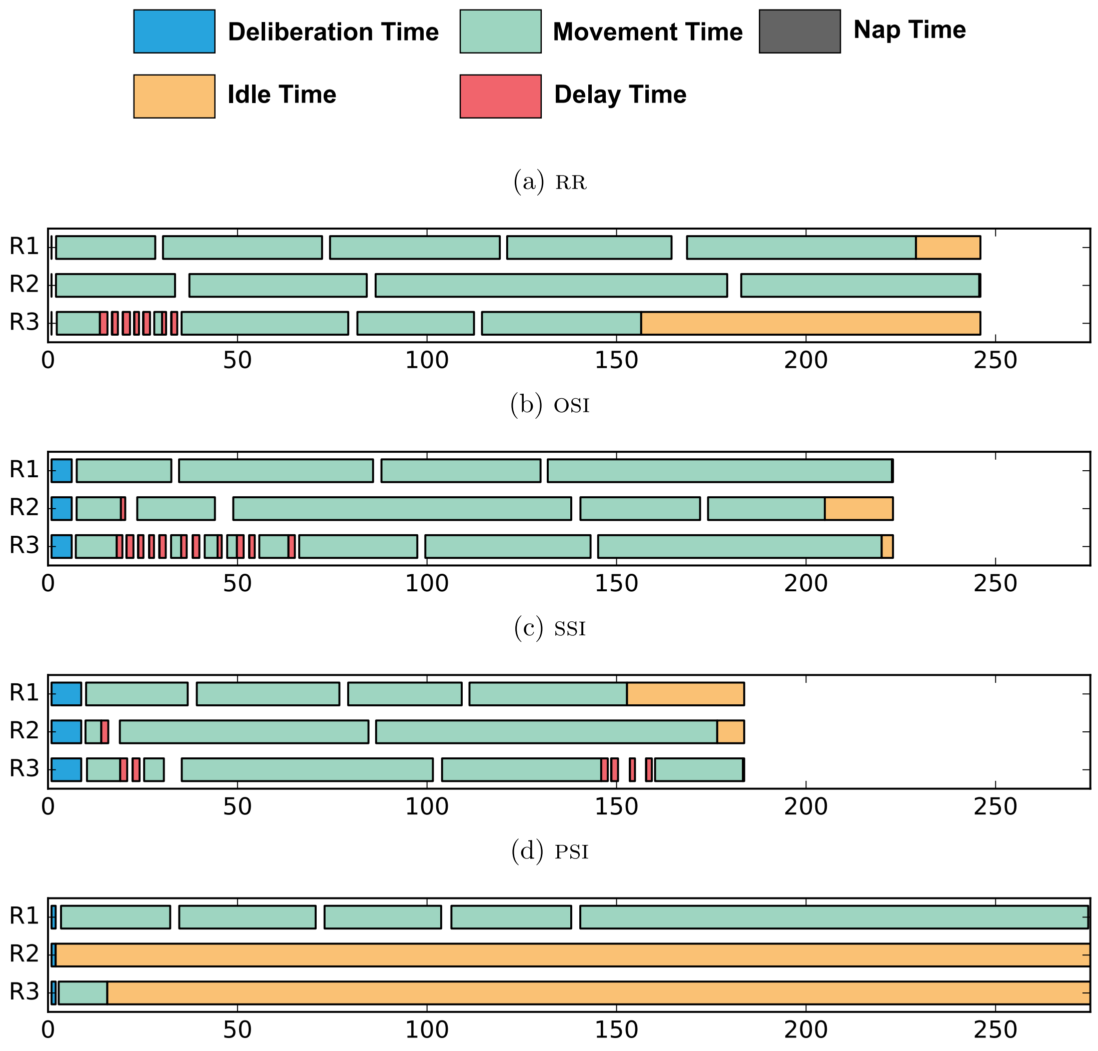
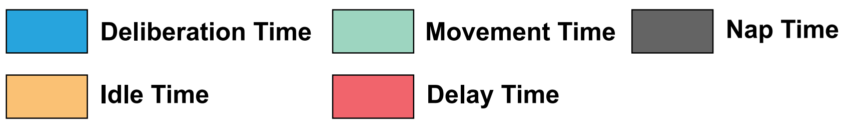
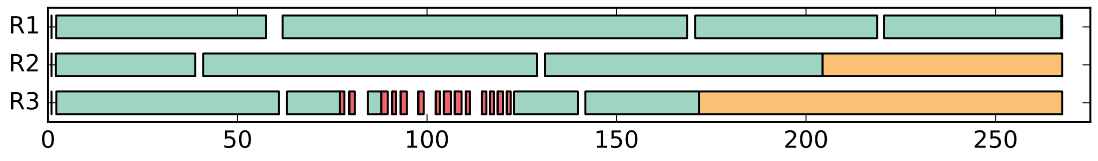


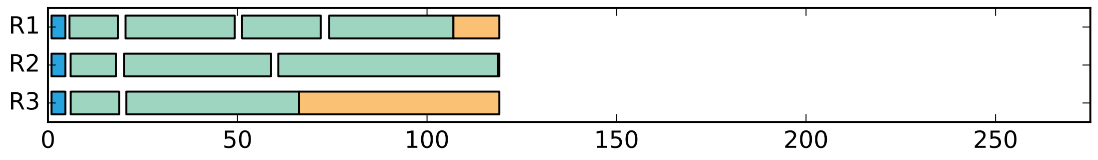
FIGURE 6.9: Timelines showing robot activity in a single trial of the **static** scenario with **clustered** start locations. An experiment begins at time 0, and moves along the positive x-axis. PSI run times were about 400 seconds (Figure 6.1) and run past the end of the timeline at the scale shown here.



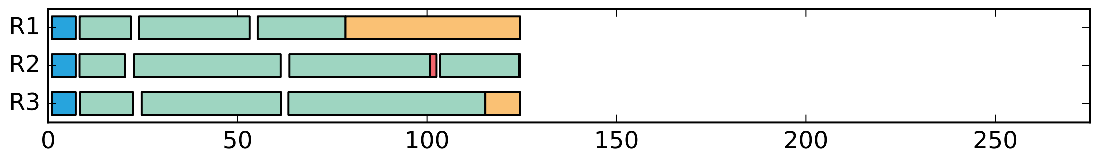
(a) RR



(b) OSI



(c) SSI



(d) PSI

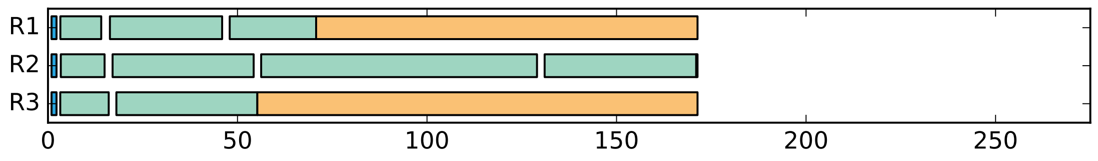


FIGURE 6.10: Timelines showing robot activity in a single trial of the **static** scenario for **distributed** start locations.

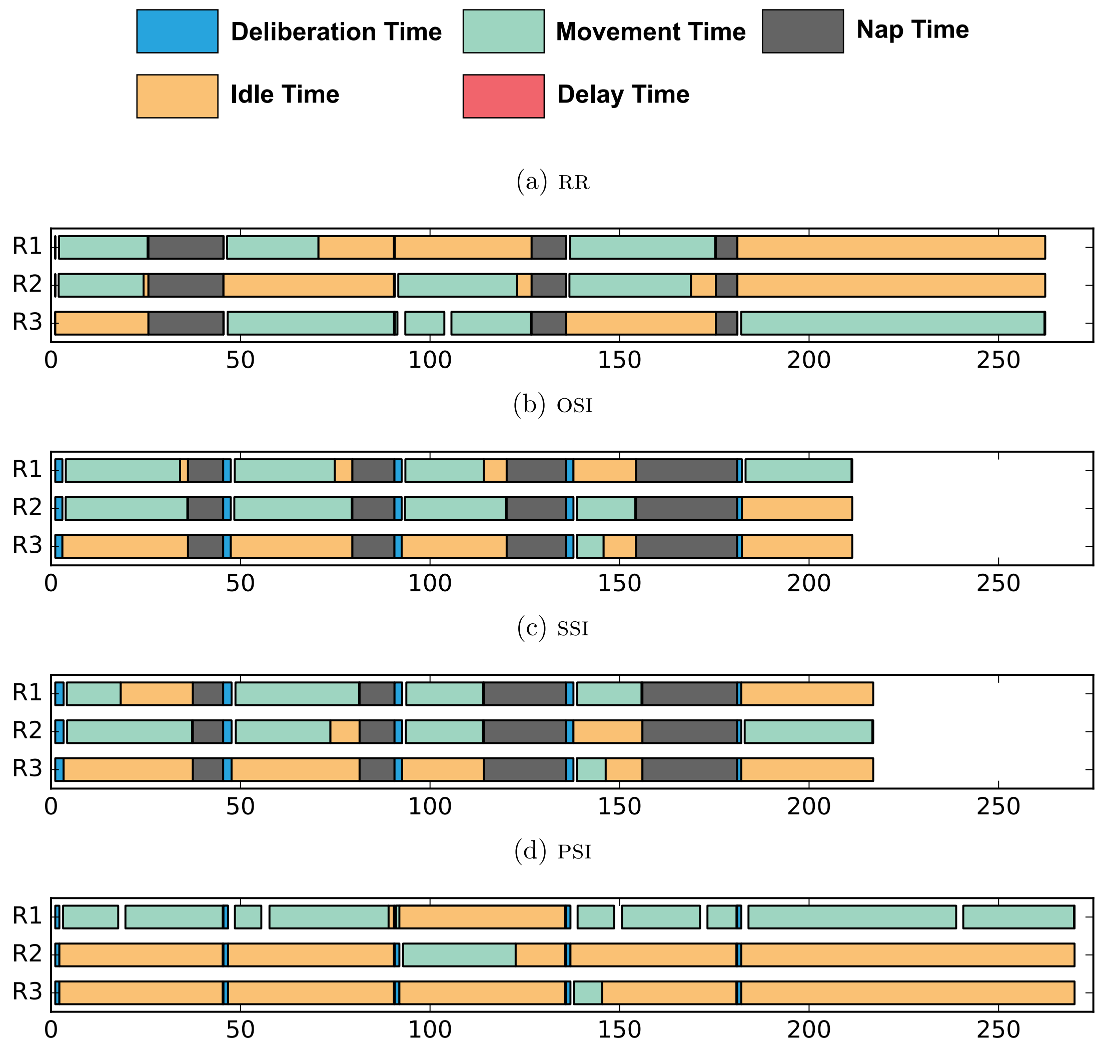


FIGURE 6.11: Timelines showing robot activity in a single trial of the **dynamic** scenario for **clustered** start locations.

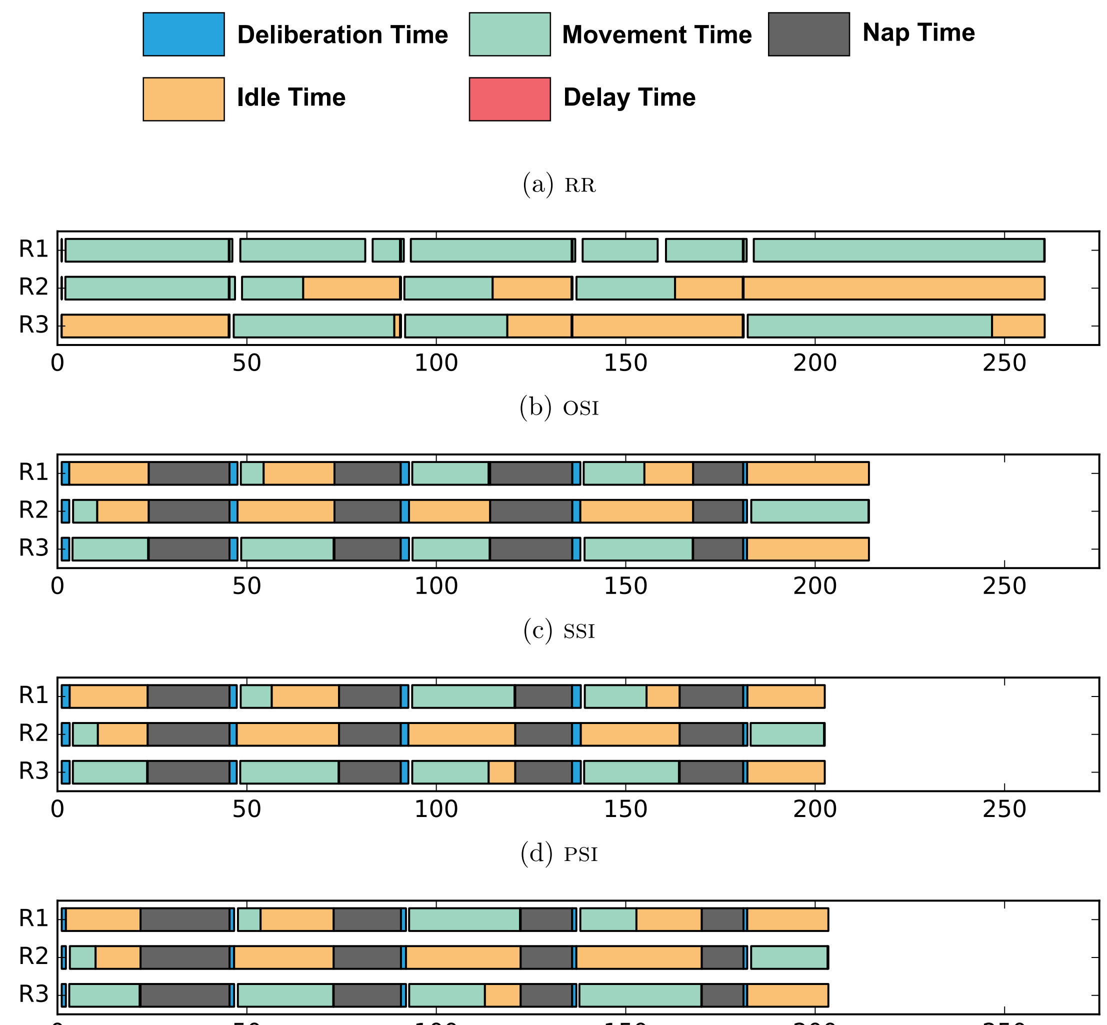


FIGURE 6.12: Timelines showing robot activity in a single trial of the **dynamic** scenario for **distributed** start locations.

Chapter 7

Multi-Robot and Constrained Tasks (MR-CT-DA)

7.1 Introduction

This chapter continues the investigation of Chapter 6 into how performance varies when task allocation mechanisms are employed across different task environments. Experiments are presented that compare the performance of the four task allocation mechanisms defined in Section 4.4 on multi-robot routing missions set in four different task environments, in which tasks may require more than one robot to be present before they they can be executed (*MR*) and which may have precedence-ordering constraints between them (*CT*) that dictate the order in which they can be performed. In all four environments, tasks arrive dynamically over time (*DA*).

Two hypotheses are tested. The first hypothesis is that *within* a single environment, the different mechanisms evaluated here produce statistically significantly different results, according to particular performance metrics. Thus, *for any one point* in the environment landscape, we can identify one task allocation mechanism that reliably performs the best for a given metric. The second hypothesis is that *across* multiple environments, there is no definitive or consistent ranking of these mechanisms across the metrics. Thus, *across all points* in the environment landscape, none of the task allocation mechanisms evaluated performs the best for a given metric. The results of experiments presented in this chapter show evidence that supports both of these hypotheses through empirical results obtained on physical robots, backed up with results obtained in simulation experiments.

Section 7.2 describes related work in the task environments investigated here, Section 7.3 describes the experiments that were run, Section 7.4 presents their results, Section 7.5 analyses how the results support the hypotheses, and Section 7.6 summarises the chapter.

The experiments and a portion of the results presented in this chapter were published in Schneider et al. (2016) [108].

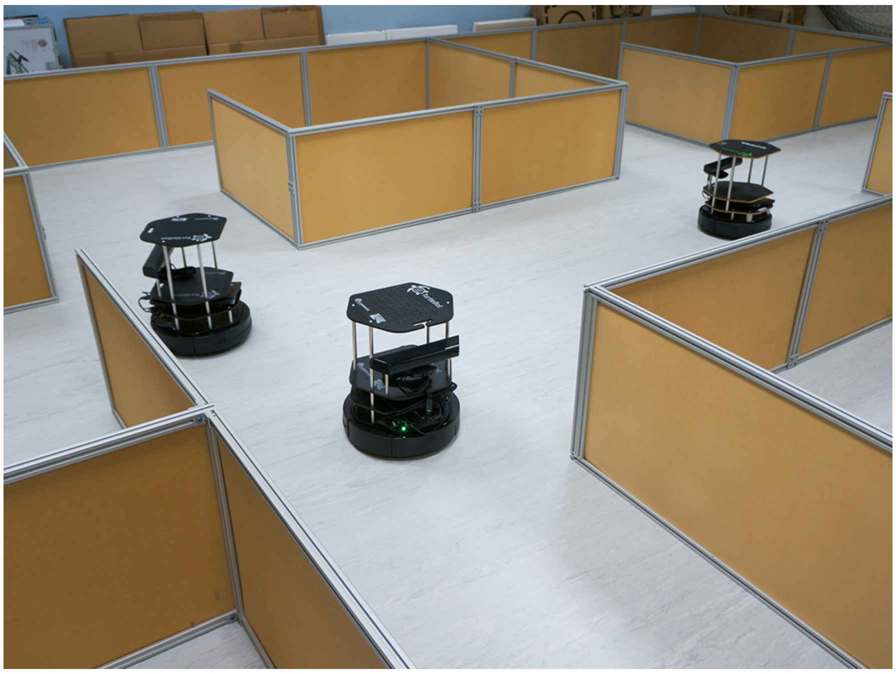
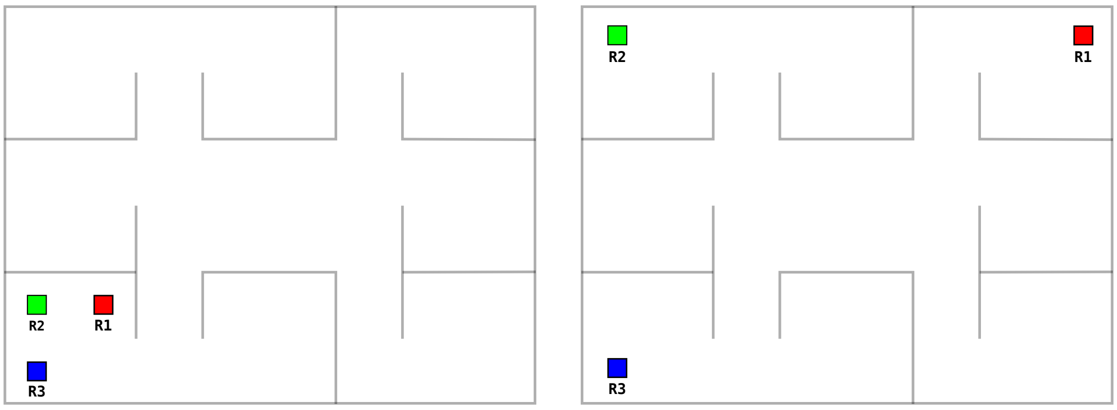


FIGURE 7.1: Physical robots in the test environment

7.2 Related work

Most of the existing work around multi-robot task allocation studies environments in which tasks are known ahead of time, are independent, can be completed in any order, and require only one robot. Existing taxonomies [36, 71] discussed in Chapter 2 suggest three task dimensions, labelling well-studied environments as *single-robot* (*SR*), *independent* (*IT*) and *static* (*SA*). The experiments presented in this chapter investigate task allocation in more complex environments. Work in Chapter 6 evaluated *static* versus *dynamic* task allocation factors, comparing situations where tasks were all known ahead of time and were allocated before execution of any task commenced (*SR-IT-SA*, Chapter 5) and situations where tasks appeared *during* execution, meaning that allocation occurred dynamically, after some tasks had commenced (SR-IT-DA, Chapter 6). Here, two additional confounding factors are considered: *multi-robot* (*MR*) tasks, where more than one robot is required (e.g., moving a heavy object); and *constrained* (*CT*) tasks, where a task may be dependent on others to be completed before it can be executed (e.g., clearing debris from a doorway before being able to enter a room).

This chapter is a further contribution to the body of work around SSI, extending the use of SSI and related mechanisms to task environments that are, according to the taxonomies developed by Gerkey and Mataríc [36] and Landén et al. [71]: *multi-robot* (*MR*), *constrained* (*CT*) and *dynamic* (*DA*). Auction-based approaches to task allocation have been proposed for tasks with precedence [78], with temporal [40, 88] constraints, and for dynamic environments [47, 86, 110, 111] with single robot tasks. Environments that contain multi-robot tasks, with and without constraints, are less well investigated than their single-robot counterparts [62].



(a) Clustered (b) Distributed

FIGURE 7.2: Robot starting locations

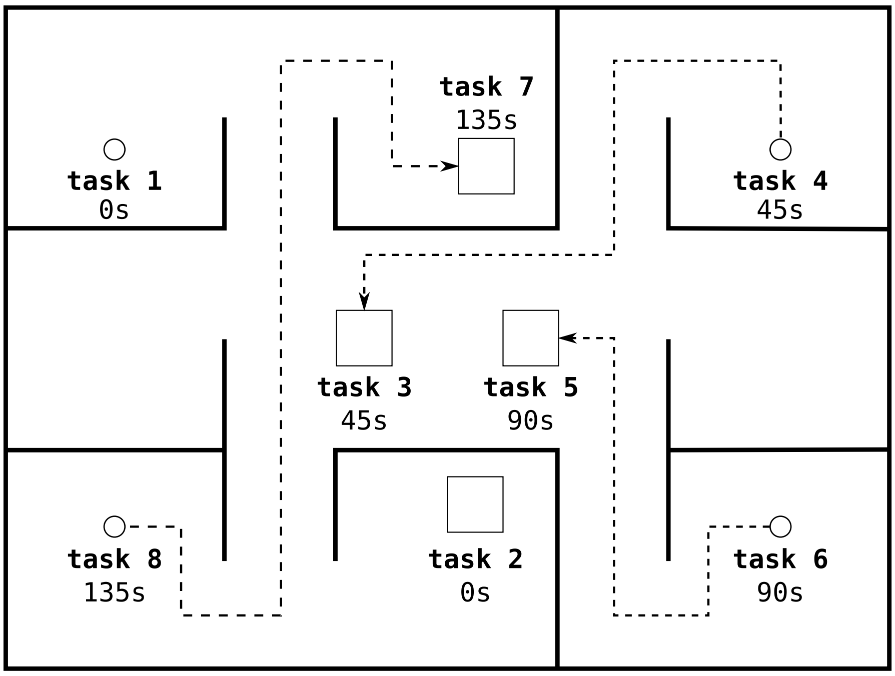


FIGURE 7.3: A dynamic scenario with single-robot tasks (circles), multi-robot tasks (squares) and precedence constraints. A dotted line from task t_p to task t_p means that t_p must be completed before task t_q can be executed. Tasks arrive and become available for allocation at the times indicated in the figure.

7.3 Experiments

Experiments were conducted to compare task allocation mechanisms in a structured set of $\langle SR|MR\rangle\langle IT|CT\rangle\langle SA|DA\rangle$ environments. Here we describe the mechanisms tested, the metrics used to measure performance, the system platform used to conduct these experiments, and the experimental setup.

7.3.1 Mechanisms Tested

The task allocation mechanisms discussed in Section 4.4 were tested in these experiments:

- Round-robin (RR)
- Ordered single-item (OSI)
- Sequential-single item (SSI)
- Parallel single item (PSI)

7.3.2 Metrics

Each experiment recorded some of the metrics described in Section 4.6:

- *Team Distance* – The sum of the distances travelled by all robots.
- *Deliberation time* – The time taken for the tasks to be allocated to the robots.
- *Execution phase time* – The time it takes robots to execute tasks during an execution phase once they have been allocated.
- *Run time* – The time between the start of an experiment and the time the last robot on the team completes the tasks allocated to it. This includes deliberation time and execution phase time.
- *Movement time* – Te time robots spend actually moving, without interruption (e.g., by a near collision), toward tasks.
- *Delay time* – The time robots spent replanning and yielding the right of way while negotiating to avoid a collision; and
- *Idle time* – The time that elapses between when a robot completes its last task and when all robots on the team have completed all of their assigned tasks
- *Waiting Time* – The time robots spend waiting at a task location before the necessary conditions for executing the task have been met, such as waiting for a team mate to arrive at the location of a multi-robot task.

7.3.3 Platform

Experiments were conducted with the MRTeAm framework (Section 3) using both physical Turtlebot 2[1] robots in the smARTLab UGV laboratory at the University of Liverpool and a simulation of the laboratory's arena in the Stage simulator [37]. The simulated robots have the same characteristics as their physical counterparts (size, shape, acceleration, and maximum speed).

[1]`http://www.turtlebot.com/turtlebot2/`

7.3.4 Experimental Setup

An *experimental condition* is defined by the starting locations of the robots and the task scenario (defined by task locations, task arrival times, constraints and robot requirements). This work investigates routing tasks—a robot executes a task simply by driving to the task's location. All of the experiments reported here involve a team of $n = 3$ robots. We used two sets of *starting configurations* (Figure 7.2) for the robot team: one *clustered* the robots in the "room" in the lower left corner of the arena, while the other *distributed* the robots at three corners of the map.

We examined four different task environments, all with dynamic allocation (DA), combining single-robot (SR) vs. multi-robot (MR) and independent (IT) vs. constrained (CT) tasks: *SR-IT-DA, SR-CT-DA, MR-IT-DA* and *MR-CT-DA*. Two scenarios were employed. Figure 7.3 shows a diagram of the first scenario.

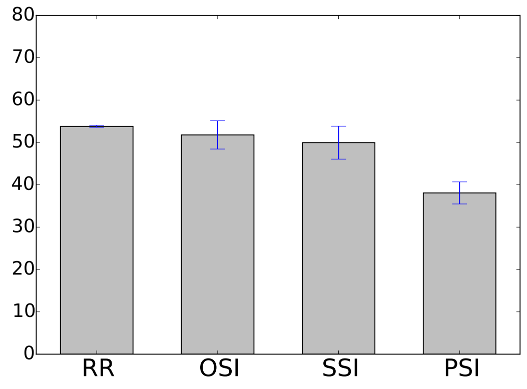
The aim in choosing this combination of task environments was to see how performance of the four task allocation mechanisms varied along the MR/SR and CT/IT dimensions. In total, 192 physical and 960 simulation trials were performed:

$$2\ \textit{starting configurations} \times 4\ \textit{task environments}\ \times 2\ \textit{scenarios} \times$$
$$4\ \textit{allocation mechanisms} \times \{3\ \textit{physical} \mid 15\ \textit{simulation}\}\ \textit{trials}.$$

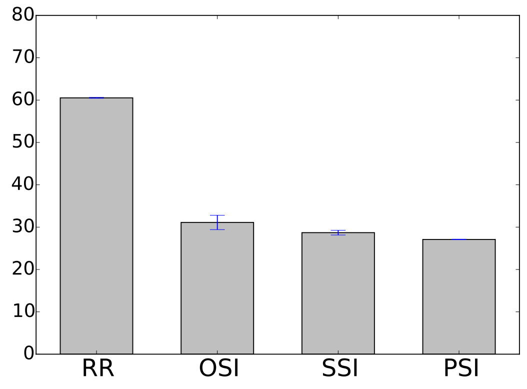
7.4 Results

Figures 7.5–7.6 and Table 7.1 contain representative results from the experiments. Figure 7.5 shows the average *team distance* by in eight variations of the scenario shown in Figure 7.3. In each plot, average travel distances resulting from allocations produced by RR, OSI, SSI, and PSI are shown from left to right.

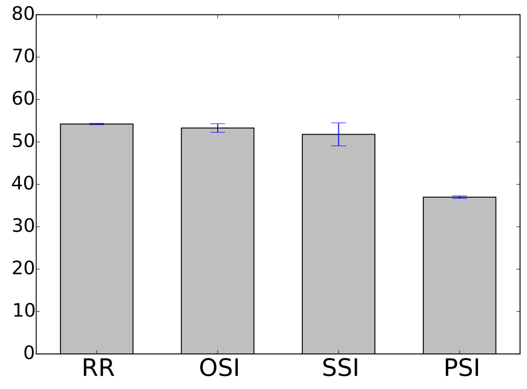
Figures 7.4a and 7.4c show how, in the SR-clustered conditions of this scenario, PSI allocations result in distances that are significantly shorter than those produced by the other mechanisms. As we move to distributed-start conditions of the scenario (Figures 7.4b and 7.4d), differences among three of the mechanisms diminish but remain statistically significantly different, while RR continues to lead to dramatically longer distances. This result is similar to those reported in Chapter 6, where it was shown that a starting configuration that distributes team members more evenly amongst tasks tends to lessen the advantages of mechanisms such as SSI that exploit clustering properties of task locations. In MR conditions of the same scenario, the results are somewhat different. For example, RR doesn't always result in the longest distances (Figures 7.5a and 7.5b) nor does PSI always result in the shortest (Figure 7.5c). The relative rankings of the mechanisms are much less predictable than in the SR environments. The second experimental scenario produced similar results.



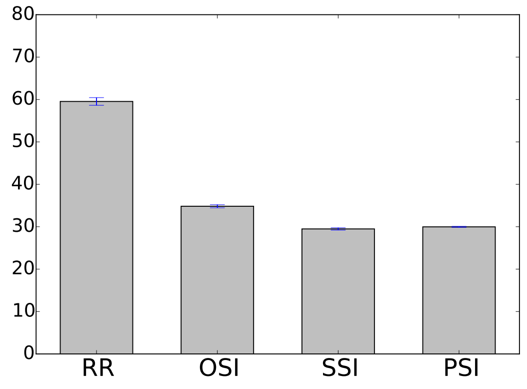
(a) SR-IT-DA, clustered



(b) SR-IT-DA, distributed



(c) SR-CT-DA, clustered



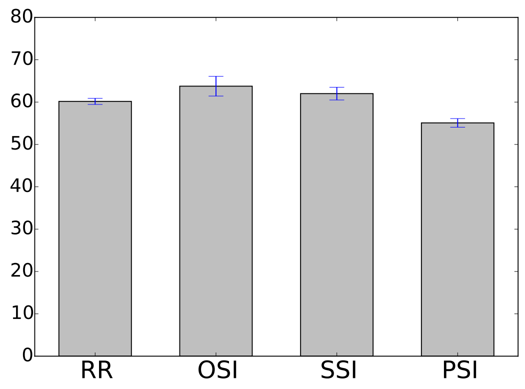
(d) SR-CT-DA, distributed

FIGURE 7.4: Average distance (metres) travelled in physical experiments for the single-robot (SR) variations of the scenario shown in Figure 7.3.

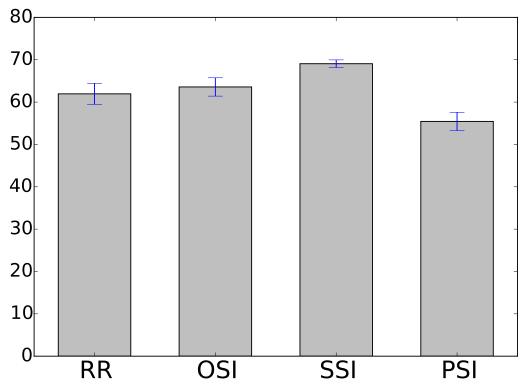
We can choose other of our performance metrics to examine individually. But with nine metrics and a large number of combinations of environments and experimental configurations, we want to make sense of the results as a whole. Do any of the mechanisms produce the best performance across environments or experimental configurations? Do clear patterns emerge? I address these questions in the following section by examining the data in aggregate.

7.5 Analysis

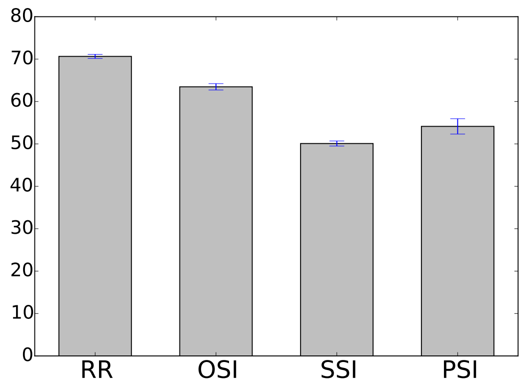
Here, we focus on five different performance metrics. *Deliberation time* is a component of overall run time and a good measure of how well an allocation mechanism scales with the number of tasks and the size of the team. *Execution phase time* is another component of run time and one of the main measures we would like to minimise, the other being *team distance*. We also look at *idle time* as a measure of how well balanced the task load is among the team. Finally, we look at *waiting time*. This is a feature specific to the MR and CT environments. A key contribution of this work is extending experimental results, particularly with physical robots, into MR and CT environments.



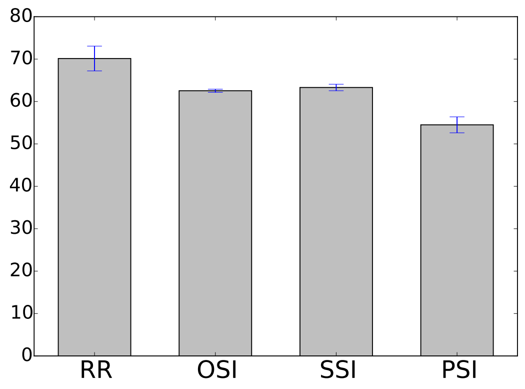
(a) MR-IT-DA, clust.



(b) MR-CT-DA, clust.



(c) MR-IT-DA, distrib.



(d) MR-CT-DA, distrib.

FIGURE 7.5: Average distance (metres) travelled in physical experiments for the multi-robot (MR) variations of the scenario shown in Figure 7.3.

As discussed above, one of the long term goals of this work is to develop task allocation mechanisms, or methods of choosing mechanisms, that perform well in different environments. Underlying this is the assumption that some mechanisms lead to better performance outcomes in some environments than others, and that there may not be a single mechanism that is best suited for all environments. I suggest two research hypotheses to evaluate this assumption and use the results of experiments discussed here to provide evidence for them.

The first hypothesis is that *within* a single $\langle sc, te, s\rangle$ tuple (where sc = starting configuration, te = task environment, and s = scenario), the four mechanisms examined here produce statistically significantly different results, according to performance metrics. It is important to show that performance differences between mechanisms exist in the first place before examining the effects of varying environments. To evaluate this first hypothesis, I apply *analysis of variance (ANOVA)* to determine if there are significant differences between the different mechanisms. I ran ANOVA on the four samples—one for each mechanism in each $\langle sc, te, s\rangle$ tuple. If the null hypothesis were true and the differences among the four samples were due to chance, then the likelihood of producing the F-ratio would be less than p%. The F-ratios of samples from both physical and simulation experiments are shown in Table 7.1. These F-ratios (p-value $= 0.01$) indicate

(a) Deliberation time

Physical						Simulation					
	$F(3,8)$	p		$F(3,8)$	p		$F(3,56)$	p		$F(3,56)$	p
MR-CT-DA-			SR-CT-DA-			MR-CT-DA-			SR-CT-DA-		
cl-s1	83.96	0.010	cl-s1	71.77	0.010	cl-s1	28709.89	0.010	cl-s1	241.51	0.010
di-s1	158.13	0.010	di-s1	43.87	0.010	di-s1	54561.93	0.010	di-s1	213.79	0.010
cl-s2	3901.58	0.010	cl-s2	1766.23	0.010	cl-s2	18630.69	0.010	cl-s2	30977.14	0.010
di-s2	3080.90	0.010	di-s2	3708.91	0.010	di-s2	22404.35	0.010	di-s2	21734.58	0.010
MR-IT-DA-			SR-IT-DA-			MR-IT-DA-			SR-IT-DA-		
cl-s1	93.79	0.010	cl-s1	5038.94	0.010	cl-s1	24591.32	0.010	cl-s1	174.05	0.010
di-s1	1150.34	0.010	di-s1	45.53	0.010	di-s1	44307.68	0.010	di-s1	2089.02	0.010
cl-s2	5124.26	0.010	cl-s2	37639.65	0.010	cl-s2	15842.79	0.010	cl-s2	23317.28	0.010
di-s2	5364.80	0.010	di-s2	146.10	0.010	di-s2	44591.27	0.010	di-s2	27112.49	0.010

(b) Execution phase time

	$F(3,8)$	p		$F(3,8)$	p		$F(3,56)$	p		$F(3,56)$	p
MR-CT-DA-			SR-CT-DA-			MR-CT-DA-			SR-CT-DA-		
cl-s1	1.39	**0.950**	cl-s1	19.70	0.010	cl-s1	30.43	0.010	cl-s1	60.39	0.010
di-s1	9.58	0.010	di-s1	3.27	**0.950**	di-s1	5.94	0.010	di-s1	22.79	0.010
cl-s2	5.49	0.050	cl-s2	19.72	0.010	cl-s2	24.02	0.010	cl-s2	19.82	0.010
di-s2	3.19	**0.950**	di-s2	24.63	0.010	di-s2	9.88	0.010	di-s2	39.51	0.010
MR-IT-DA-			SR-IT-DA-			MR-IT-DA-			SR-IT-DA-		
cl-s1	2.82	**0.950**	cl-s1	18.58	0.010	cl-s1	36.39	0.010	cl-s1	33.09	0.010
di-s1	1.54	**0.950**	di-s1	11.17	0.010	di-s1	9.53	0.010	di-s1	14.14	0.010
cl-s2	3.92	**0.950**	cl-s2	9.93	0.010	cl-s2	6.62	0.010	cl-s2	28.28	0.010
di-s2	0.77	**0.950**	di-s2	79.93	0.010	di-s2	5.10	0.010	di-s2	15.93	0.010

(c) Team distance

	$F(3,8)$	p		$F(3,8)$	p		$F(3,56)$	p		$F(3,56)$	p
MR-CT-DA-			SR-CT-DA-			MR-CT-DA-			SR-CT-DA-		
cl-s1	7.76	0.010	cl-s1	30.83	0.010	cl-s1	35.88	0.010	cl-s1	312.84	0.010
di-s1	13.04	0.010	di-s1	784.63	0.010	di-s1	4817.66	0.010	di-s1	75593.00	0.010
cl-s2	12.90	0.010	cl-s2	7.70	0.010	cl-s2	33.75	0.010	cl-s2	60.12	0.010
di-s2	9.39	0.010	di-s2	996.79	0.010	di-s2	132.54	0.010	di-s2	1395.83	0.010
MR-IT-DA-			SR-IT-DA-			MR-IT-DA-			SR-IT-DA-		
cl-s1	10.38	0.010	cl-s1	6.01	0.050	cl-s1	390.48	0.010	cl-s1	436.66	0.010
di-s1	68.46	0.010	di-s1	173.25	0.010	di-s1	3121.61	0.010	di-s1	98521.39	0.010
cl-s2	13.30	0.010	cl-s2	29.16	0.010	cl-s2	122.99	0.010	cl-s2	231.39	0.010
di-s2	10.21	0.010	di-s2	2823.98	0.010	di-s2	527.39	0.010	di-s2	3676.25	0.010

(d) Idle time

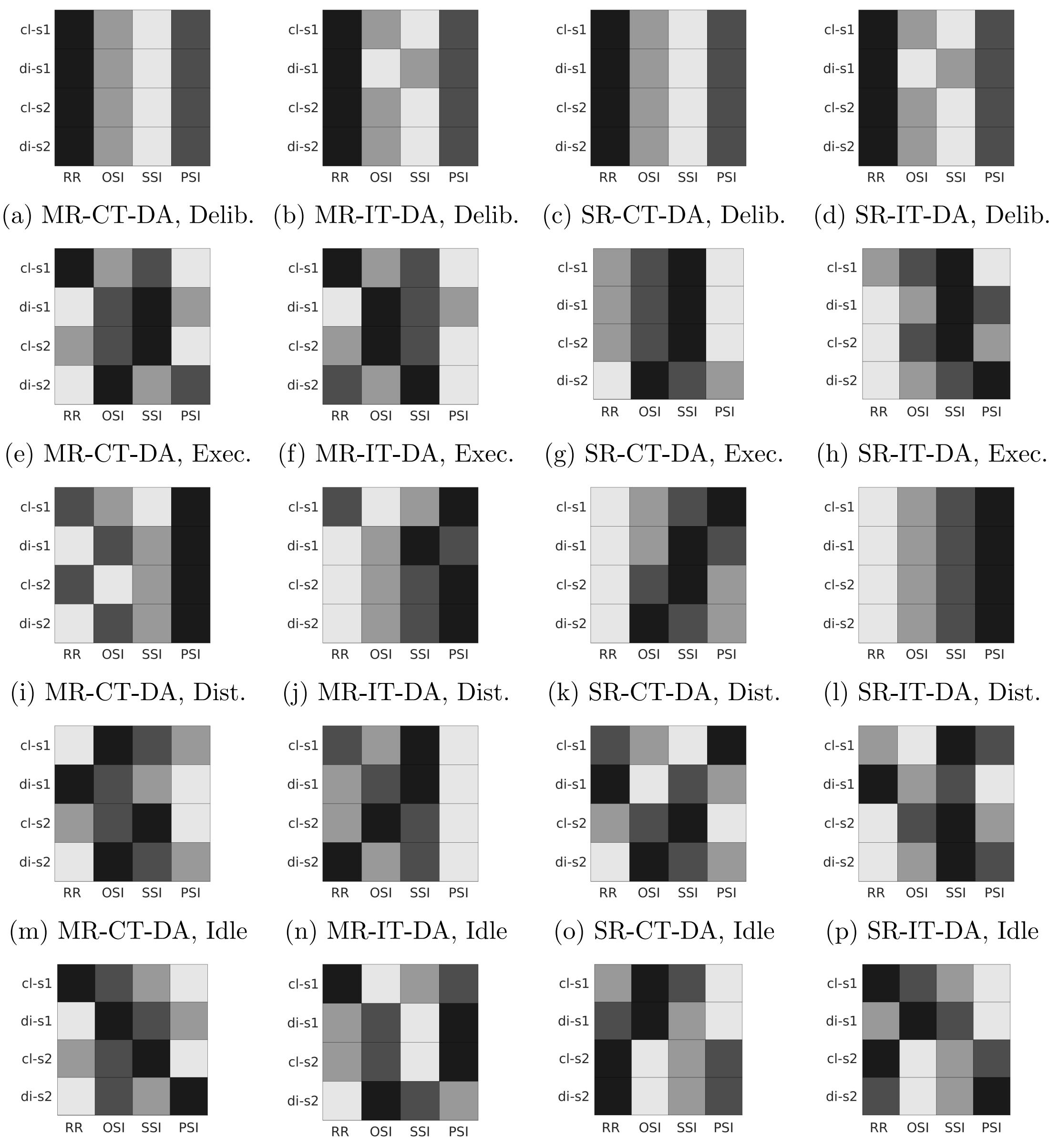
	$F(3,8)$	p		$F(3,8)$	p		$F(3,56)$	p		$F(3,56)$	p
MR-CT-DA-			SR-CT-DA-			MR-CT-DA-			SR-CT-DA-		
cl-s1	0.72	**0.950**	cl-s1	8.44	0.010	cl-s1	40.63	0.010	cl-s1	40.08	0.010
di-s1	2.17	**0.950**	di-s1	4.33	0.050	di-s1	36.85	0.010	di-s1	34.25	0.010
cl-s2	8.28	0.010	cl-s2	6.23	0.050	cl-s2	112.31	0.010	cl-s2	7.00	0.010
di-s2	4.30	0.050	di-s2	29.89	0.010	di-s2	70.23	0.010	di-s2	29.02	0.010
MR-IT-DA-			SR-IT-DA-			MR-IT-DA-			SR-IT-DA-		
cl-s1	111.22	0.010	cl-s1	2.19	**0.950**	cl-s1	117.09	0.010	cl-s1	14.87	0.010
di-s1	7.90	0.010	di-s1	6.69	0.050	di-s1	40.33	0.010	di-s1	52.47	0.010
cl-s2	20.62	0.010	cl-s2	4.12	0.050	cl-s2	99.37	0.010	cl-s2	12.82	0.010
di-s2	16.53	0.010	di-s2	90.31	0.010	di-s2	16.58	0.010	di-s2	8.40	0.010

(e) Waiting time

	$F(3,8)$	p		$F(3,8)$	p		$F(3,56)$	p		$F(3,56)$	p
MR-CT-DA-			SR-CT-DA-			MR-CT-DA-			SR-CT-DA-		
cl-s1	26.38	0.010	cl-s1	100.07	0.010	cl-s1	10.02	0.010	cl-s1	1260.61	0.010
di-s1	1.28	**0.950**	di-s1	9.19	0.010	di-s1	30.23	0.010	di-s1	100.39	0.010
cl-s2	0.15	**0.950**	cl-s2	6.01	0.050	cl-s2	16.90	0.010	cl-s2	38.08	0.010
di-s2	8.92	0.010	di-s2	22.55	0.010	di-s2	20.93	0.010	di-s2	6.94	0.010
MR-IT-DA-			SR-IT-DA-			MR-IT-DA-			SR-IT-DA-		
cl-s1	4.21	0.050	cl-s1	0.25	**0.950**	cl-s1	28.44	0.010	cl-s1	0.63	**0.950**
di-s1	0.26	**0.950**	di-s1	0.42	**0.950**	di-s1	23.39	0.010	di-s1	0.64	**0.950**
cl-s2	0.30	**0.950**	cl-s2	2.49	**0.950**	cl-s2	14.05	0.010	cl-s2	0.00	**0.950**
di-s2	1.00	**0.950**	di-s2	1.69	**0.950**	di-s2	8.03	0.010	di-s2	1.32	**0.950**

Table 7.1: F-ratios for 5 different metrics

a significant performance difference between the populations (mechanisms). For example, in the case of deliberation time (Table 7.1(a)), very large F-ratio values are the result of comparing RR, a simple mechanism that runs very quickly, with the others.



(a) MR-CT-DA, Delib. (b) MR-IT-DA, Delib. (c) SR-CT-DA, Delib. (d) SR-IT-DA, Delib.

(e) MR-CT-DA, Exec. (f) MR-IT-DA, Exec. (g) SR-CT-DA, Exec. (h) SR-IT-DA, Exec.

(i) MR-CT-DA, Dist. (j) MR-IT-DA, Dist. (k) SR-CT-DA, Dist. (l) SR-IT-DA, Dist.

(m) MR-CT-DA, Idle (n) MR-IT-DA, Idle (o) SR-CT-DA, Idle (p) SR-IT-DA, Idle

(q) MR-CT-DA, Wait. (r) MR-IT-DA, Wait. (s) SR-CT-DA, Wait. (t) SR-IT-DA, Wait.

FIGURE 7.6: Heat maps for the physical experiment data on each task environment. Each heatmap shows the two different scenarios and two different experimental conditions. For a given scenario/experimental condition pair (row) the colour of the squares indicates the rank order of the mechanism (column). The darkest square indicates the lowest value of the metric (best mechanism), the lightest square indicates the highest value (worst mechanism). a–d show deliberation time, e–h show execution time, i–l show distance, m–p show idle time, and q–t show waiting time.

In contrast, F-ratios for distance travelled (Table 7.1(c)) are lower but still above the critical value for the significance level and degrees of freedom tested. This supports the first hypothesis.

The second hypothesis is that *across* multiple $\langle sc, te, s\rangle$ tuples, there is no definitive ranking amongst the metrics for each mechanism. Figure 7.6 shows performance rankings obtained from physical experiments in the form of heat maps. Each row of heat maps in the figure corresponds to one of the five metrics discussed above. Within each heat map, the four columns correspond to the four task allocation mechanisms (RR, OSI, SSI, PSI, from left to right). The rows of each heat map are labelled with a variation of a particular scenario. For example, *cl-s1* indicates *clustered, scenario 1.* The shading of a cell indicates its rank: darker shades indicate lower values for that metric. While the ANOVA results mentioned in support of the first hypothesis don't directly measure the degree to which any pair of mechanisms differed in performance, they do provide evidence that the rankings shown in the heat maps are based on statistically significant differences. The heat maps for *deliberation time* (Figure 7.6a–7.6d) reveal some consistency when comparing environments and experimental conditions (rows within a single heat map, and across heat maps in the same row of the figure). RR is always the quickest to run, followed by PSI, while OSI and SSI trade ranks depending on the experimental condition. Apart from deliberation time, this type of performance ranking does not hold in a consistent way for the other metrics when comparing across environments and experimental conditions. This supports the second hypothesis.

7.6 Summary

The work presented in this chapter tests two hypotheses: (1) within a single parameterised environment, a given task allocation mechanism can be proven to consistently outperform others for certain metrics; and (2) across a varied set of parameterised environments, no single task allocation mechanism will consistently outperform others for any metrics. We conducted experiments with physical robots, as well as simulated robots in an environment that parallels our physical setup. Empirical results presented here support both of these hypotheses.

Chapter 8

Mechanism Selection

8.1 Introduction

Previous chapters have examined how the performance of task allocation mechanisms can vary when applied across different missions within the same task environment (Chapter 5) and across different task environments (Chapters 6 and 7). This chapter returns to the SR-IT-SA from environment Chapter 5 and presents a method for selecting a task allocation mechanism that is suited to the mission in which it is employed. The method selects an allocation mechanism from a portfolio of available mechanisms by examining spatial features of the environment.

Many market-based mechanisms have been suggested for the task allocation problem, as reviewed in Chapter 2. These mechanisms vary considerably in the trade-offs that they make between computation time and space, and the quality of solutions that they deliver, measured by metrics such as the total distance covered by the team while completing a set of tasks. In addition, the performance of mechanisms seems to be greatly affected by the environments in which they are deployed. In some environments, a simple, greedy mechanism which might not be expected to perform well in the general case may, in fact, perform competitively with more sophisticated mechanisms, with the advantage of scaling better. The experimental results in Chapters 5–7 and published in [106, 108, 110] have shown evidence that this is the case in both simulated and physical experiments.

In particular, Chapters 5–6 have shown that while the sequential single-item auction (SSI) [58] performs better than the parallel single-item auction (PSI) [58] when the allocation is carried out with robots clustered together geographically, this advantage diminishes as robots are distributed over space and tasks are distributed over space and time. Based on this observation, this chapter proposes a portfolio-based approach to the MRTA problem. Given a set of task allocation mechanisms and a set of environmental features that can be measured, a portfolio-based approach to mechanism selection should be able to classify a previously unseen mission environment in order to choose a task allocation mechanism that performs well in it.

"Mission environment" here refers to the spatial arrangements and distributions of robots and tasks. It seems appropriate to apply the tools and techniques of *cluster* *analysis* to these environments. Environmental obstacles like walls need to be considered, so it seems natural to model environments as graphs, where nodes may be robots and/or task locations, and edges are paths computed by a path planner (e.g., A* [45]) between these nodes, around obstacles. The graphs constructed in this way resemble something like road networks, which suggests an approach to characterising different environments.

The distribution of sites over road-network-like graphs is a well-studied research area in *Geographic Information Systems (GIS).* One particular class of problem from GIS that is useful to apply here is *location-allocation* or *facility location*, which seeks to determine the ideal locations for "facilities" and allocates "demand points" to them in a way that minimises some measure of overall cost or maximises some overall utility. Examples of facilities and demand points might be warehouses and customers, or police stations and potential crime scenes, respectively. In the family of facility location problems [97], the p-median problem (described below) seems most suitable here, with p representing the number of facilities one wishes to locate.

In the case of multi-robot routing missions, robot team members can be thought of as facilities and task locations as demand points. If such a facility location problem can be solved for a multi-robot routing mission, where the number of facilities is equal to the number of team members, it may be possible to find an ideal set of team starting locations for a simple, greedy mechanism like the parallel single-item auction (PSI). The hypothesis investigated by experiments presented in this chapter is that if actual robot start locations are close to ideal facility locations, then the parallel single-item auction will lead to competitive performance. Conversely, if actual robot start locations are far away from ideal facility locations, then a more sophisticated mechanism like the sequential single-item auction is a better choice. Furthermore, it will be possible to select the best mechanism for specific sets of start and facility locations based only on knowledge about those locations. This chapter provides an empirical test of this hypothesis and finds that, at least for some sets of locations, *machine learning* can be used to identify the best mechanism to use. Experiments show that this approach can produce significant improvements in team performance.

Section 8.2 describes related work, Section 8.3 explains the mechanism selection method proposed here works, Section 8.4 describes experiments that were run to evaluate the method's performance, Section 8.5 presents the results, Section 8.6 discusses the results and the effectiveness of the method, and Section 8.7 summarises the chapter.

Some of the experimental results presented in this chapter were published in Schneider et al. (2017) [110].

8.2 Related Work

Location theory sits at the intersection of (GIS) and economics. The *p-median* problem is one class of *location-allocation* or *facility location* problem that seeks to find optimal locations among existing sets of points that either maximize some measure of distribution utility or minimize some measure of cost [73]. Reese [97] gives the following definition of the p-median problem on a graph:

> Given a graph or a network $G = (V, E)$, find $V_p \subseteq V$ such that $|V_p| = p$ and that the sum of the shortest distances from the vertices in $\{V \setminus Vp\}$ to their nearest vertex in V_p is minimized.

Hakimi developed such problems on a graph to locate optimal switching centres for communication networks or police stations in a highway system [44]. Kariv & Hakimi showed that finding solutions to p-median problems is NP-hard on a general graph [54], but heuristics have been developed to make this more efficient [21, 120].

Clustering or bundling of tasks has been investigated in the design of task allocation mechanisms. Sandholm (1998) extended Smith's Contract Net Protocol with *C-contracts* (cluster contracts), which award bundles of tasks, rather than single tasks, to agents [102]. Dias & Stentz proposed a mechanism that clusters geographically close tasks into a forest of minimum spanning trees, which may then be auctioned and potentially swapped [23]. Heap proposed sequential-single-cluster (SSC) auctions, an extension to SSI that uses an agglomerative clustering algorithm to create task bundles, which are then auctioned as in SSI [46]. Liu & Shell [76] develop a hybrid distributed-centralised approach to MRTA. The task set is first partitioned into subsets that are then solved in parallel using a centralised assignment algorithm [68].

The problem of algorithm selection and defining criteria for selecting an algorithm were proposed at least as early as Rice (1976) [98]. Computational or algorithm portfolios that use domain knowledge to define features of problem instances in order to select an appropriate algorithm have been investigated by Huberman et al. [49], Gomes & Bart [41], and Leyton-Brown et al. [75]. Portfolio-based SAT solvers like SATzilla [135] and Hydra [134] have had success in selecting appropriate heuristics to solve NP-hard problems.

8.3 Portfolio-based Mechanism Selection

The mechanism selection method proposed here uses a corpus derived from the results of experimental runs carried out over a range of missions (that is, over a range of spatial arrangements of task and robot start locations) on a particular map to train a classifier to select a mechanism from a portfolio. Once trained, the classifier should select a mechanism that will optimise (typically minimise) some performance metric when employed for a previously unseen mission. The classifier does not make forward predictions about performance (e.g., the distance the team will travel) during a mission when a given mechanism is employed, but rather attempts to minimise the regret of choosing an alternative mechanism that does not perform well. It is possible to make an "offline" prediction of the best-performing mechanism for a given mission by running a series of simulations, one per mechanism, and choosing the best among them. However, such a

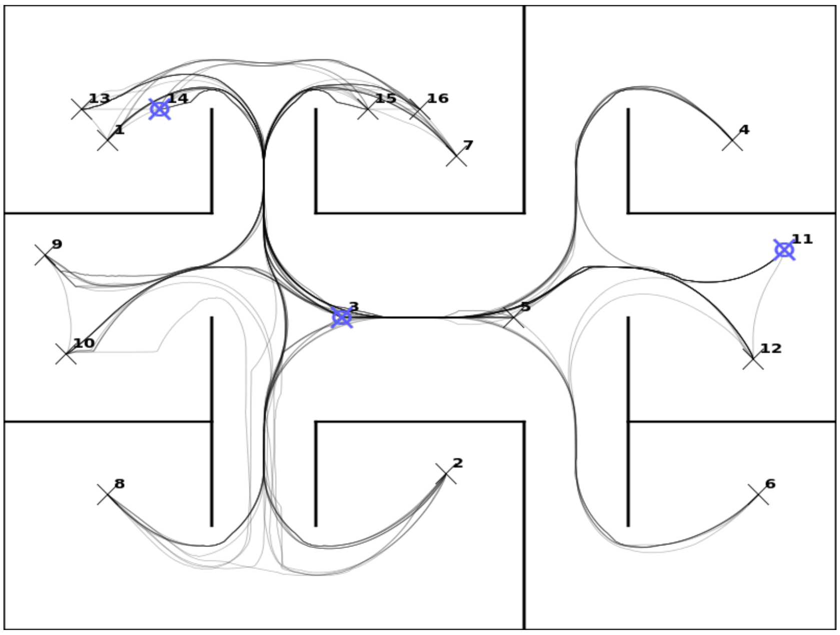


Figure 8.1: A complete task graph for the scenario shown in Fig. 8.4a–8.4d. Task locations are shown as × marks. Locations of medians are indicated small coloured ovals.

forward prediction method would not be practical in a setting that demands a timely prediction, especially since a sufficiently high-fidelity simulator must run in real-time, as discussed in Section 4.7 (p. 47). The portfolio used for experiments presented in this chapter comprises two mechanisms, the parallel single-item (PSI) auction and the sequential single-item (SSI) auction, and thus the trained classifier is a binary discriminator. Future work will incorporate the ordered single-item (OSI) and round-robin (RR) mechanisms into the portfolio.

The method works as follows. On a map (shown in Figures 8.1–8.2), a large number of *training missions* are generated in which tasks and robot starting locations are randomly chosen from a uniform distribution over the map (detailed in Section 8.4.1). For each training mission, an experimental run is conducted with both PSI and SSI auction mechanisms. This generates a pair of results with the same starting conditions but different performance outcomes. From these results, a training instance for each mission is created by recording properties of that mission as training features (Table 8.1) and the winning mechanism, for some performance metric to optimise, as a label. Finally, after balancing the training set and selecting features, a (binary) classifier is trained to predict a winning mechanism. We can now, in previously unseen missions (i.e., task and robot starting locations), query a classifier at runtime to select a mechanism that it predicts will perform best in that environment.

8.3.1 Training Features

The features used to train a classifier are based on the locations of medians of a *task graph* constructed between task locations (Figure 8.1) and the distances between medians and

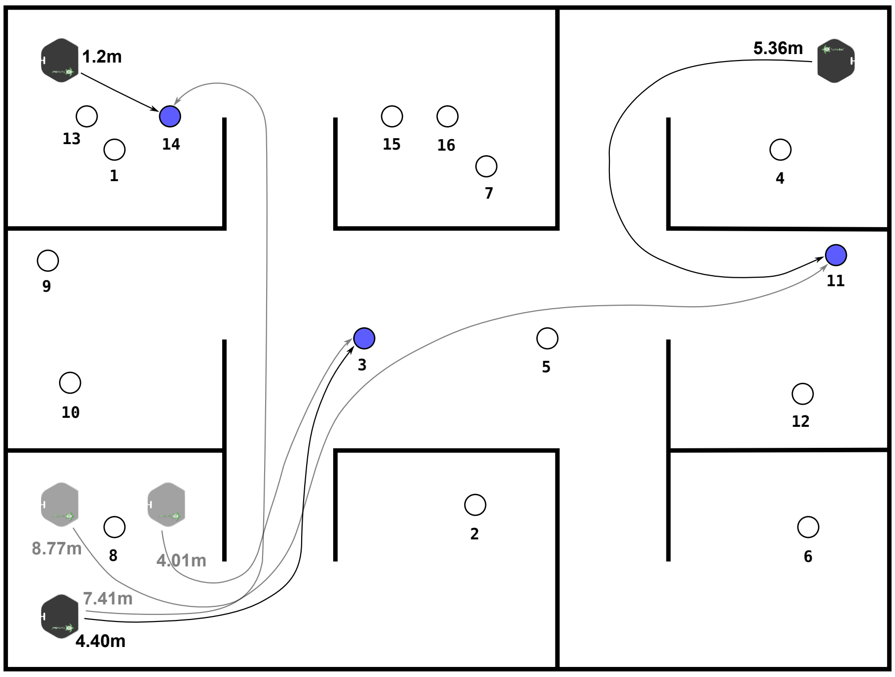


FIGURE 8.2: The path distance of each robot to its assigned median for *clustered* (faint) and *distributed* (dark) starting locations with the same set of task locations.

robot starting locations (Figure 8.2), where distances are measured from paths planned between two locations. Robots and tasks are thought of as "facilities" and "demand nodes", respectively, as in a facility location problem, and the number of medians p of a task graph is equal to the size of the team $n = |R|$.

Three steps are taken before the features can be computed:

1. Constructing a task graph

A path planner[1] is invoked to find a path between each pair of task locations. The result is a complete graph whose nodes represent task locations and edges are paths planned between them (Figure 8.1). The path planner is invoked $O(m^2)$ times, where $m = |T|$ is the number of tasks in a scenario, at a cost that must be considered when evaluating the run-time performance of this method. The cost of constructing a task graph in practice is discussed in Section 8.6.

2. Finding medians

The task graph is represented as a weighted adjacency matrix as input for a median solver. An implementation of the Teitz-Bart method [120] by Xiao [133] locates $p = n$ medians of the task graph coincident with task locations.

[1]The ROS global planner (Chapter 3), the same A* planner used by mechanisms to compute bid costs and by robots to navigate to task locations.

3. Assigning medians to robots

Assigning medians to robots is a task allocation problem in itself. Two methods of assignment are considered:

- *Greedy median assignment* uses a method similar to the parallel single-item (PSI) auction (Section 4.4), and assigns each median to the robot whose starting location it is closest to by path distance.
- SSI-*median assignment* uses a method similar to the sequential single-item auction (SSI) (Section 4.4).

Examples of SSI-median assignment are shown in Figures 8.2 and 8.4.

The following training features are then computed (summarised in Table 8.1):

- *Total distance to assigned medians* measures the sum of all robots' path distances from their starting locations to their assigned median locations.
- *Total distance to all medians* measures the sum of all robots' path distances from their starting locations to all median locations regardless of median assignments.
- *Maximum distance to assigned median* measures the maximum path distance of any one robot from its starting location to its (SSI-)assigned median location.
- *Maximum distance to any median* measures the maximum path distance of any robot's starting location to any median location.
- *Minimum distance to assigned median* measures the minimum path distance of any robot from its starting locations to its (SSI-)assigned median location.
- *Minimum distance to any median* measures the minimum path distance of any robot from its starting location to any median location.
- *Assigned median distance spread* measures the difference between the *Maximum distance to assigned median* and the *Minimum distance to assigned median.*
- *Total median distance spread* measures the path difference between the *maximum distance to any median* and the *minimum distance to any median.*
- *Greedy median count spread* measures the difference between the maximum number of medians assigned to any one robot via *greedy assignment* and the minimum number of the same.
- *Team diameter* measures the longest path distance between any two robots.

These features are recorded for each training instance.

Feature	Description
total dist. to assigned medians	Sum of all robots' distances to their (SSI-) assigned medians
total dist. to all medians	Sum of all robots' distances to all medians
max. distance to assigned median	Max. distance of any robot to its (SSI-)assigned median
max. distance to any median	Max. distance of any robot to any median
min. distance to assigned median	Min. distance of any robot to its (SSI-)assigned median
min. distance to any median	Min. distance of any robot to any median
assigned median distance spread	*max. distance to assigned median* − *min. distance to assigned median*
total median distance spread	*max. distance to any median* − *min. distance to any median*
greedy median count spread	max. number of medians greedily (PSI-)assigned to any one robot − min. number of the same
team diameter	Longest distance between any two team members

Table 8.1: Training features based on task, median, and robot starting locations

8.4 Experiments

I conducted experiments to compare the portfolio-based method of mechanism selection presented in this chapter (hereafter referred to as **SEL**) to the task allocation mechanisms studied in previous chapters, with the aim of investigating whether mechanism selection can improve performance, according to some objective, of a robot team executing its mission compared to employing the same mechanism alone across a range of missions.

Experiments were conducted in three configurations, described below. Each configuration consisted of two stages: a training stage and an evaluation stage that compared performance on the same set of starting conditions (the locations of tasks and robots). The experiments were set in an environment in which tasks were single-robot, independent, and statically allocated (SR-IT-SA). The size of the robot team was fixed at $n = 3$ robots and the number of tasks in each scenario was fixed at $m = 16$. All experiments were conducted with MRTeAm in Stage simulator on the University of Liverpool's Chadwick computing cluster (as described in Chapter 3).

Mechanisms Compared

Performance was compared between three task allocation methods: the parallel single-item auction (PSI), the sequential single-item auction (SSI) and the portfolio-based method (SEL). The SEL method was trained so that it could select either PSI or SSI to carry out task allocation at run time based on the features of the starting conditions measured at the outset of an experimental trial.

The PSI auction was chosen because the allocations it produces have been shown (in Chapters 5–7 and previous work [106, 108, 110]) to lead to performance that is, by some metrics, competitive with the best-performing mechanism overall that was studied, the SSI auction. Additionally, the computation and communication costs of a PSI allocation are lower than for SSI which allows it to scale better with the number of tasks.

Performance Metrics and Objectives

Performance among the three task allocation methods was compared using the metrics defined in Section 4.6 and used in experiments in previous chapters. *Team distance* measures the sum of the lengths of paths travelled by all team members over the course of an experiment. *Maximum robot distance* measures the longest distance travelled by any one robot in the course of an experiment. It gives an indication of how balanced the load of a mission is across the team for a given allocation. *Deliberation time* measures the time taken by an mechanism to allocate tasks to robots. In the case of SEL, deliberation time includes the time taken to build a task graph, compute spatial features, and select a mechanism from its portfolio. *Execution phase time* measures how long it takes for robots to complete their tasks once an allocation has been made. *Run time* measures the overall time taken by an experiment, that is, it is the sum of deliberation time and execution phase time.

Maximum robot distance and execution phase time chosen used as performance objectives for the SEL method.

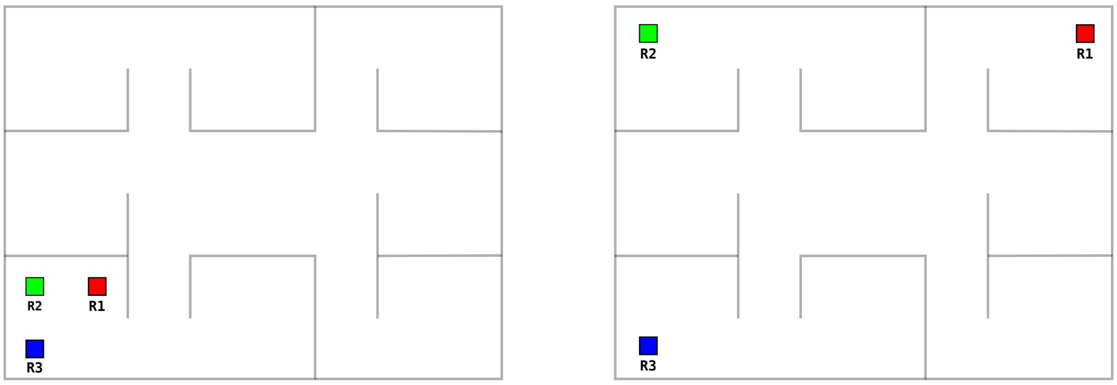
Experimental Configurations

Three different experimental configurations were investigated:

1. In the *maximum distance, random start* configuration, the SEL method was trained to minimise the maximum robot distance objective and evaluated against the performance of PSI and SSI in missions where task and robot starting locations were randomly chosen.

2. In the *execution time, random start* configuration the SEL method was trained to minimise the execution phase time objective and evaluated against the performance of PSI and SSI in missions where task and robot starting locations were randomly chosen.

3. In the *execution time, fixed start* configuration the SEL method was trained to minimise the execution phase time objective and evaluated against the performance of PSI and SSI in missions where task locations were randomly chosen but robot starting locations were fixed in the *clustered* and *distributed* configuration used shown in Figure 8.3 (also used for experiments in Chapter 7).



(a) Clustered (b) Distributed

FIGURE 8.3: Robot starting locations for the *execution time, fixed start* experimental configuration.

8.4.1 Training

The SEL method trains on the results of running a large number of missions in which task and robot starting locations are randomised (Section 8.3). In the experiments reported here, training missions, were set on the smARTLab UGV map shown in Figures 8.1–8.3 (also used for experiments in Chapter 7). Locations for both task and robot starting locations were chosen uniformly randomly with a small buffer distance away from the walls (and robot starting locations from each other).

1000 training missions were generated. Both of the mechanisms in the SEL method's portfolio, PSI and SSI, were run on each (all in simulation) to produce a training instance for each of the performance objectives tested (maximum robot distance and execution phase time), for a total of 1000 $\times$ {PSI, SSI} $=$ 2000 runs, producing 1000 labelled training examples.

8.4.2 Mechanism Selection Evaluation

Test missions with randomised task locations were used to compare performance of the three task allocation methods (PSI, SSI, and SEL) in each of the three experimental configurations:

1. For the *maximum distance, random start* experimental configuration, 300 test missions were generated with random robot starting locations. The three task allocation methods to be compared were run on each test mission for a total of 300 $\times$ {PSI|SSI|SEL} $=$ 900 experimental runs.

2. For the *execution time, random start* configuration, 300 test missions were generated with random robot starting locations. The three task allocation methods were run on each test mission for a total of
300 $\times$ {PSI, SSI|SEL} $=$ 900 experimental runs.

Metric	PSI Wins	%	SSI Wins	%
maximum robot distance	85	9.28	**831**	**90.72**
execution phase time	118	12.88	**798**	**87.12**

TABLE 8.2: "Winners" of objectives in the class-unbalanced training set

3. For the *execution time, fixed start* configuration, 150 test missions were generated for each of the *clustered* and *distributed* starting configurations. The three task allocation methods were run on each test mission for a total of
$150 \times \{\textit{clustered,distributed}\} \times \{\text{PSI}|\text{SSI}|\text{SEL}\} = 900$ experimental runs.

In total, 2700 experimental runs were conducted on test missions across the three experimental configurations.

8.5 Results

Results from three stages of the experiments are presented. First, some properties of the test missions that served as training data for classifiers are discussed. Second, classifier accuracy is shown on held-out portions of the training data. Finally, the ultimate results of the three task allocation mechanisms applied to previously unseen missions are compared.

8.5.1 Training Data

Training Missions

Of the 1000 training missions generated, only 916 {PSI|SSI} pairs of runs were successfully completed due to run failures. A run failure was typically due either to a software crash or a failure of the ROS navigation stack to avoid an obstacle (like a wall), resulting a robot becoming "stuck" to the obstacle. In such cases of navigation failure, the run was terminated after 15 minutes of time had elapsed. In general, SSI dominated performance on the training missions, producing allocations that led to both lower maximum robot distances (831/916 = 91%) and lower execution phase times (798/916 = 87%) than PSI (Table 8.2). These results demonstrate the effectiveness of SSI at producing efficient allocations—at least compared to those produced by PSI—even for performance objectives other than ones it was designed to optimise (*team distance*, or MINIMAX).

Table 8.3 gives the mean performance values observed for the two performance objectives that resulted from allocations produced by PSI, SSI and compares them with an ideal mechanism selector (an "oracle") able to predict with perfect accuracy which of PSI or SSI allocations would yield better performance on the same training mission. An ideal selector would be able to perform slightly better than SSI on both objectives (although it would be difficult to perform significantly better for the range of task and robot starting locations observed in the training missions).

	PSI		SSI		Ideal SEL	
Metric	μ	σ	μ	σ	μ	σ
maximum robot distance	20.97	5.89	13.08	1.696	**12.96**	**1.704**
execution phase time	327.26	88.13	232.756	49.01	**226.68**	**40.69**

Table 8.3: PSI and SSI performance compared to an "ideal" mechanism selector on the training missions.

Training Instances

Training instances were created by recording properties of each training mission as features (Section 8.3.1) and the winning mechanism as a label, for each of the {*maximum robot distance*|*execution phase time*} performance objectives.

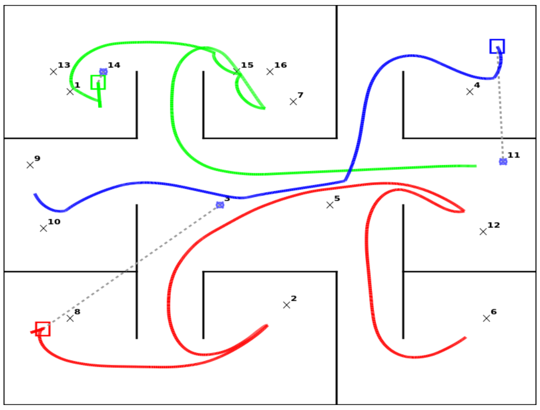
Figure 8.4 visualizes some of the features recorded from two hypothetical training missions that share the task locations but have different robot starting locations. The figure shows assignments of robots to medians as dotted lines. Figures 8.5a and 8.5b visualise training instances in for *execution phase time* objective. In each figure, red and blue samples indicate instances in which SSI and PSI, respectively, had a lower execution phase time.

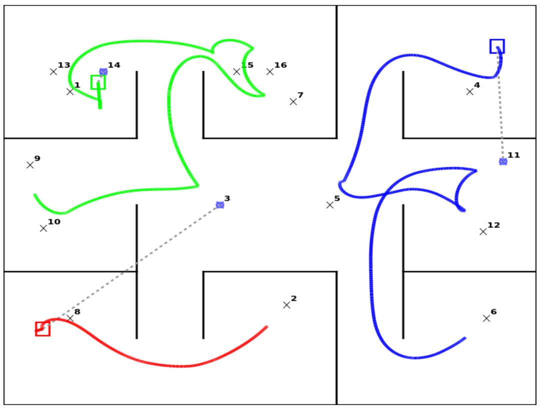
8.5.2 Classifier Performance

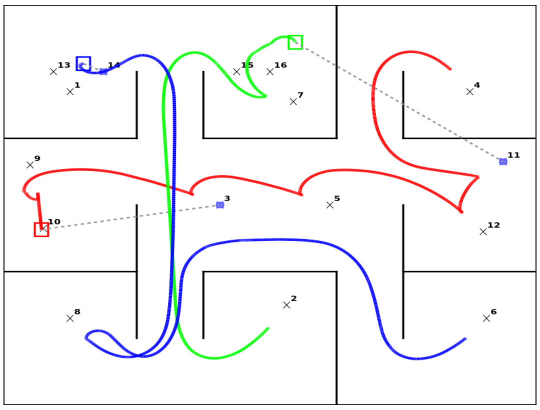
Initially, the training sets had a severe class imbalance. In the training set for the *maximum robot distance* objective, for example, SSI was the winning mechanism in 831 cases compared to PSI's count of 85. The training sets were balanced using a random undersampling method [74], although other methods are possible. Figures 8.5a and 8.5b show instances of a training set before and after balancing. The scikit-learn [94] library was used to select features and train classifiers. Several types of classifier were evaluated, including decision trees,[2] k-nearest neighbours,[3] random forests,[4] and support vector machines.[5] Table 8.4 shows the average accuracy of some of these classifiers on held-out data over 10-fold cross validation. Parameters of the classifiers were tuned using an exhaustive grid search.[6]

As a result of these experiments, the random forest classifier was selected for the remainder of the work presented here. The features selected for the random forest classifier trained to optimise the *execution phase time* objective were: {*maximum distance to assigned median*, *assigned median distance spread*, *team diameter*, and *greedy median count spread*}. Features selected for the random forest classifier trained to optimise the

[2] `http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`
[3] `http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`
[4] `http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`
[5] `http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`
[6] `http://scikit-learn.org/stable/modules/grid_search.html`


(a) SSI, max. robot dist. loser


(b) PSI, max. robot dist. winner


(c) SSI, max. robot dist. winner

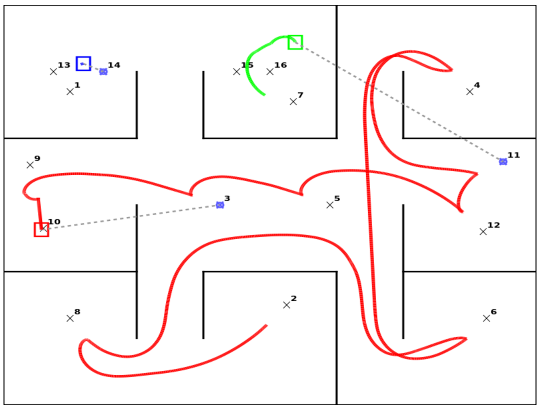
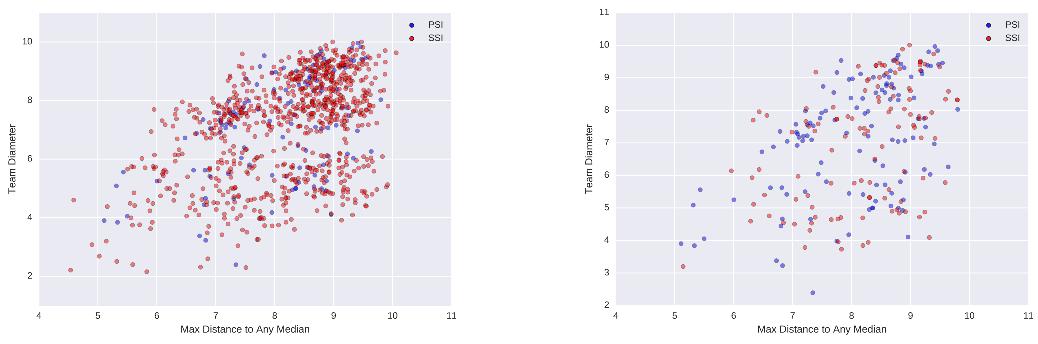

(d) PSI, max. robot dist. loser

FIGURE 8.4: Trajectories and median assignments for two sets of *random start, random task location* environments. Note the different robot start locations between the top and bottom rows. Robot start locations are shown as large open coloured squares, task locations are shown as × marks, medians are shown as small closed coloured squares, and assignments of medians to robots are shown as dotted lines. (a) and (b) show a mission for which the PSI allocation led to a smaller maximum robot distance. (c) and (d) show a mission for which an SSI allocation led to a smaller maximum robot distance. In (a), the green robot's start location and its assigned median are close together and the line between them is barely visible.

Classifier Type	Objective	Accuracy	Std. Dev
Random Forest	Execution Phase Time	75.22%	0.91%
SVM	Execution Phase Time	74.55%	1.00%
Random Forest	Max. Robot Distance	80.88%	1.26%
SVM	Max. Robot Distance	76.80%	1.20%

TABLE 8.4: Accuracy of several classifiers trained for different performance objectives.

maximum robot distance objective were: {*total distance to all medians*, *maximum distance to any median*, *team diameter*, and *greedy median count spread*}. Other technical details of the random forest classifier are discussed in Appendix C (p. 141).



(a) Class-unbalanced

(b) Class-balanced

FIGURE 8.5: Training features *maximum distance to any median* by *team diameter* in a class-unbalanced (left) and class-balanced (right) training set. Red and blue samples indicate where SSI and PSI performed best, respectively.

	Max. Robot Distance	Deliberation Time	Run Time
SSI	**13.18** ± 0.18	32.92 ± 0.11	**270.24** ± 5.34
PSI	21.01 ± 0.59	**3.24** ± 0.03	333.21 ± 8.49
SEL	19.19 ± 0.51	35.6 ± 1.38	340.71 ± 7.63

TABLE 8.5: Results for the *maximum distance, random start* experimental configuration. Units are metres. Values are means with 95% confidence intervals.

	Run time	Deliberation time	Execution time
SSI	**270.64** ± 5.82	32.95 ± 0.11	**235.74** ± 5.83
PSI	326.36 ± 8.87	**3.23** ± 0.03	321.19 ± 8.87
SEL	347.89 ± 8.67	33.57 ± 1.06	312.37 ± 8.28

TABLE 8.6: Results for the *execution time, random start* experimental configuration. Time is in seconds. The values given are means with 95% confidence intervals.

8.5.3 Mechanism Selection Results

Having trained a classifier for each of the two performance objectives, we then used them in experiments to see if the SEL method, which uses initial locations of tasks and robots to pick an allocation mechanism using these classifiers, could outperform either SSI or PSI alone (i.e., as the only task allocation available in the system). Any runs that failed to complete were repeated so that results from the full set of 900 runs for each experimental configuration were collected. The results of these experiments are given in Tables 8.5–8.7 and Figures 8.6–8.10.

Results from the three experimental configurations are discussed in turn.

		Run time	Deliberation time	Execution time
Clustered	SSI	**353.89** ± 11.37	33.02 ± 0.21	**318.92** ± 11.37
	PSI	555.4 ± 11.22	**3.38** ± 0.04	550.08 ± 11.22
	SEL	391.73 ± 14.34	58.2 ± 1.31	331.58 ± 14.84
Distributed	SSI	273.47 ± 12.88	32.82 ± 0.18	238.71 ± 12.84
	PSI	**197.92** ± 5.08	**3.52** ± 0.02	**192.46** ± 5.07
	SEL	222.55 ± 5.55	27.23 ± 0.86	193.38 ± 5.25
Combined	SSI	309.81 ± 10.07	32.91 ± 0.14	274.95 ± 10.04
	PSI	313.96 ± 24.46	**3.47** ± 0.02	308.54 ± 24.47
	SEL	**294.43** ± 12.79	40.39 ± 2.11	**252.1** ± 11.23

TABLE 8.7: Results for the *execution time, fixed start* experimental configuration, with starting locations shown separately and combined. Time is in seconds. The values given are means with 95% confidence intervals.

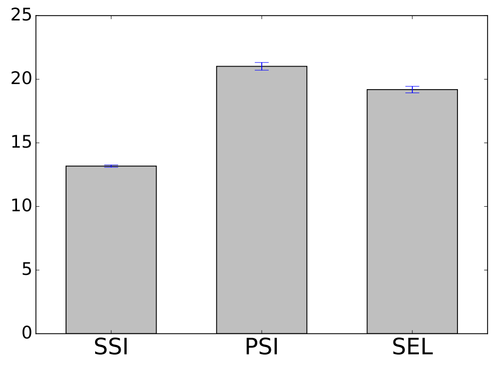


FIGURE 8.6: Maximum robot distance for the *maximum distance, random start* experimental configuration. Units are seconds.

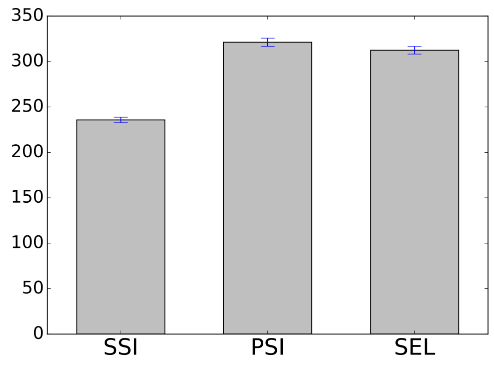
Maximum distance, random start

Results for the *maximum distance, random start* experimental configuration are shown in Table 8.5 and Figure 8.6. The SEL method did not outperform SSI in terms of maximum robot distance. SSI allocations led to significantly lower maximum robot distances on average (13.18 metres) than both SEL (19.19 metres) and PSI (21.01 metres). The average maximum robot distance produced by SEL allocations was slightly lower than that of PSI, but significantly so ($t = 4.59, p = 5.195$).

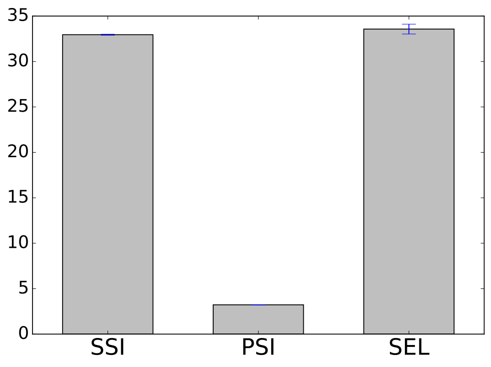
In addition, SEL's deliberation time was significantly higher than that of SSI or PSI since it constructs a task graph as part of its selection process (Section 8.3.1), and this led to higher run times than the other two mechanisms.

Execution time, random start

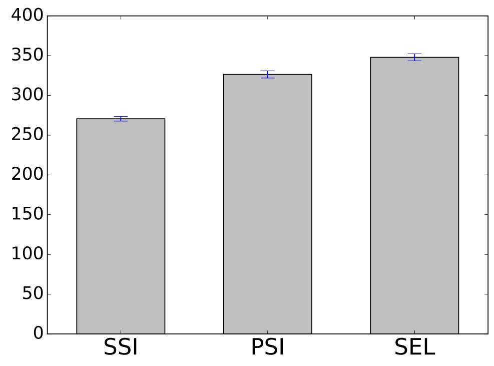
Results for the *execution time, random start* experimental configuration are shown in Table 8.6 and Figure 8.7. Similar to the results found in the *maximum distance, random start* experimental configuration, the SEL method did not outperform SSI in terms



(a) Execution Phase Time



(b) Deliberation Time



(c) Run Time

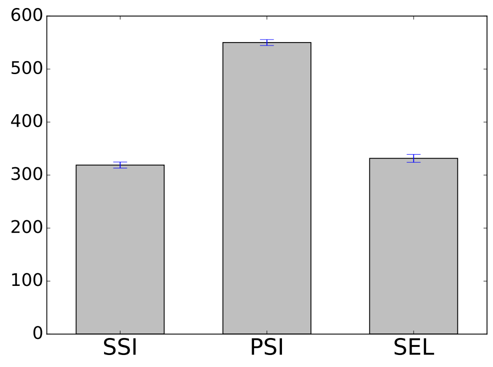
FIGURE 8.7: Execution phase time (8.7a), deliberation time (8.7b), and run time (8.7c) for the *execution time, random start* experiments. Units are seconds.

of average execution phase time when robot start locations were randomised. SSI allocations led to significantly lower execution phase times on average (235.74 seconds) than both SEL (312.37 seconds) and PSI (321.19 seconds). The average execution phase time produced by SEL allocations was slightly, but significantly lower than that of PSI ($t = 1.43$, $p = 0.15$). In terms of deliberation time, SEL's task graph was again costly enough to compute that it led to the highest run times of any mechanism tested (Figure 8.7c).

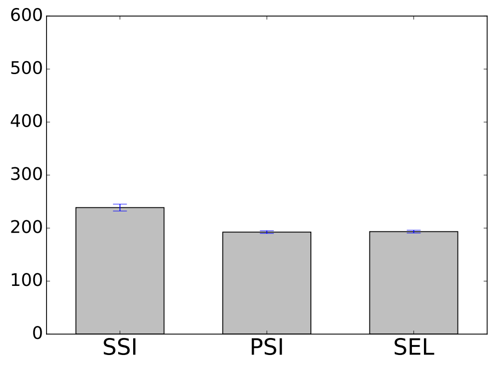
Execution time, fixed start

Results for the *execution time, fixed start* experimental configuration are shown in Table 8.7 and Figures 8.8–8.10. With *clustered* starting locations, SEL allocations led to an average execution phase time (331.58 seconds) only slightly higher than the best-performing mechanism, SSI (318.92 seconds) and far lower than PSI (550.08 seconds). With *distributed* starting locations, SEL allocations led to an average execution phase time (193.38 seconds) that was not significantly higher than the best-performing mechanism, PSI (192.46 seconds) and was significantly lower than SSI (238.71 seconds).

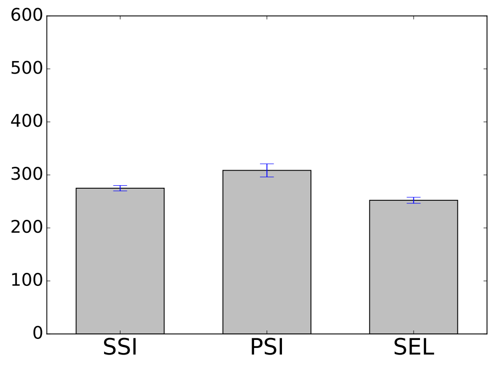
When results for missions with *clustered* and *distributed* starting locations are combined, SEL allocations led to average execution phase times (252.1 seconds) that were



(a) Clustered



(b) Distributed



(c) Combined

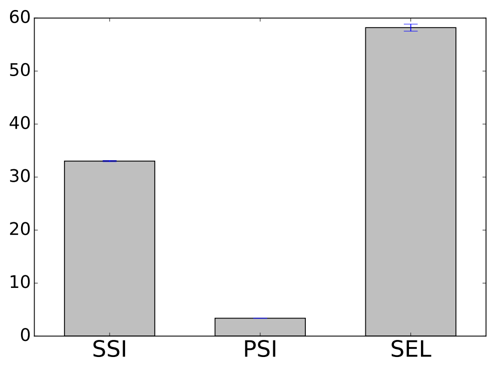
FIGURE 8.8: Execution phase time for clustered starts (8.8a), distributed starts (8.8b), and the combination of clustered and distributed starts (8.8c) for the *execution time, fixed start* experimental configuration. Units are seconds.

significantly lower than either PSI (308.54 seconds) or SSI (274.95 seconds) ($t = 2.99$, $p = 0.0029$). In addition, while the cost of computing SEL's task graph made its average deliberation time (40.39 seconds) the highest of the three methods, its low execution time led to an average run time (294.43 seconds) that was significantly lower than either PSI (313.96 seconds) or SSI (309.81 seconds).

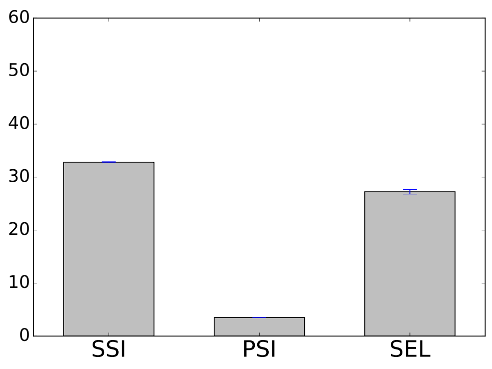
8.6 Discussion

Over the three experimental configurations, performance of the SEL method was mixed. The SSI auction mechanism showed its effectiveness at producing efficient allocations [58, 70] across a range of missions and performance objectives. The SEL method did not outperform PSI or SSI in the *maximum robot distance* or *execution, random start* experimental configurations, but did show significant improvement over the other mechanisms in the *execution, fixed start* configuration when the results for the *clustered* and *distributed* starting locations were combined.

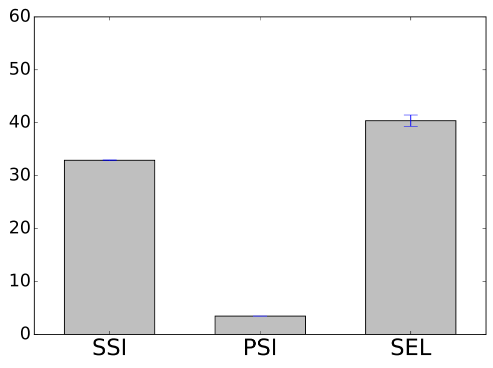
The robot starting locations of the *execution, fixed start* configuration were manually chosen based on the results of previous experiments (Chapters 5 and 6), where it was observed that the performance advantages of the SSI auction over other mechanisms



(a) Clustered



(b) Distributed



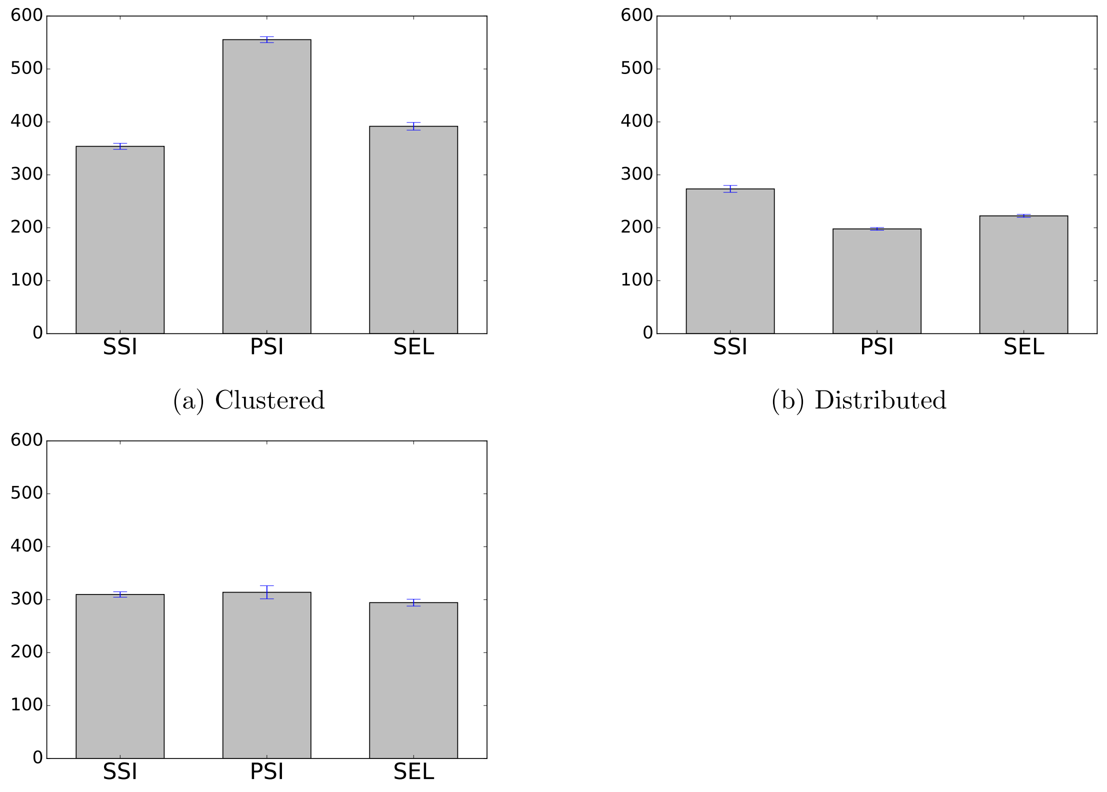
(c) Combined

FIGURE 8.9: Deliberation time for clustered starts (8.9a), distributed starts (8.9b), and the combination of clustered and distributed starts (8.9c) for the *execution time, fixed start* experimental configuration. Units are seconds.

like PSI diminished, or even disappeared, when robot starting locations were spread out ("distributed") to the corners of the map. In such cases, it might be preferable to employ a simpler mechanism such as PSI, which has its own advantage of lower computational and communication costs. Experimental results for the SEL method presented here show that, at least under the *fixed start* conditions, a method that selects a mechanism from a portfolio by examining spatial features of an environment before conducting an allocation can increase performance over using a single mechanism alone. It should be noted that the SEL method was trained on missions in which robot starting locations were randomised, so the results of the *execution, fixed start* experimental configuration do not test performance on training data.

The SEL method proposed here has several drawbacks. First, it did not lead to significantly improved performance in general in the missions tested here when robot starting locations were chosen randomly. This suggests that the spatial features used to train its classifiers, based on the medians of a graph constructed between task locations, may not capture

Second, the SEL method requires a large amount of historical data to train its classifiers, data which may not be available for missions that involve exploring unknown environments. Finally, the cost of classifying an environment incorrectly and selecting



(a) Clustered

(b) Distributed

(c) Combined

FIGURE 8.10: Run time for clustered starts (8.10a), distributed starts (8.10b), and the combination of clustered and distributed starts (8.10c) for the *execution time, fixed start* experimental configuration. Units are seconds.

an ineffective mechanism may be high. The cost of constructing a task graph scales quadratically with the number of tasks in a mission ($O(m^2)$), and this can be seen in the results for deliberation time across all three experimental configurations.

8.7 Summary

This chapter has presented a method for selecting a task allocation mechanism from among a portfolio of available mechanisms by examining spatial features of a mission before an allocation is conducted. Experimental results show that this method can lead to significantly increased performance for a performance objective under certain starting conditions of a mission.

Chapter 9

Conclusions and Future Work

This thesis has investigated the use of market-based mechanisms for multi-robot task allocation with the aim of discovering how the performance of mechanisms varies when they are employed in different kinds of environments. Performance is evaluated with a number of metrics that measure resource usage of a robot team as it executes a mission as well as the costs of conducting allocations themselves, with the intention of understanding the trade-offs that are made between the cost and solution quality of an allocation. The investigation has been primarily experimental, conducted on physical robots when possible and in high fidelity simulations otherwise. The principal results of the investigation show that while the theoretical performance guarantees of mechanisms are sometimes supported empirically, factors such as spatial arrangements of tasks and inter-robot interference can complicate predictions of performance.

A long term goal is to determine the suitability of mechanisms to particular environments. The experimental investigation presented in Chapters 5–7 shows that, given a number of task allocation mechanisms to choose from, a single mechanism may not perform best across all task environments. This thesis proposes a task allocation method that examines features of an environment to select a mechanism, from a number of options, that will perform best in that environment (Chapter 8). Experimental results show that, under certain circumstances, this method can lead to significantly greater performance than employing a single mechanism alone.

9.1 Summary of contributions

1. **An experimental software framework** (Chapter 3)
MRTeAm[1] is an implementation of a multi-robot system designed to conduct experimental investigations of multi-robot mission performance. It is used to conduct experiments on both physical and simulated robots with minimal modification to the behaviour between the two. MRTeAm can perform routing missions with tasks that require multiple robots, tasks that have precedence-ordering constraints, and tasks that arrive over time. As part of MRTeAm, I have also developed the ROS

[1] `http://github.com/nitsuga/mrta`

Master Bridge, an extension of the Robot Operating System (ROS) [96] communication framework that enables multi-robot communication. MRTeAm has been used to conduct the experiments presented in Chapters 6–8.

2. **Rich Performance metrics** (Chapter 4)
   Metrics often used to measure the performance of a multi-robot routing mission are the distance travelled by the robot team over the course of a mission and the time taken to reach task locations. Sometimes overlooked is the cost of computing an allocation itself, an important factor when considering the scalability of missions and teams. Inter-robot interference due to the need to avoid collisions can hamper robots' execution of a mission and confound predictions of performance based on allocations alone. The time robots spend idle while team mates execute tasks is also a measure of inefficiency of the team. This thesis defines a set of metrics that measure these factors and tell a much richer story about the performance of a mission than commonly used metrics can.

3. **An empirical investigation of task environments** (Chapter 5–8)
   This thesis has presented a set of multi-robot routing experiments conducted in a series of increasingly complex task environments, with the aim of discovering how theoretical guarantees of mission performance, based on allocations alone, are borne out in practice. Experiments with a team of autonomous robots reveal how confounding factors, like inter-robot interference and the need to re-plan during task execution, can complicate performance expectations. The experiments also reveal how spatial and temporal arrangements of tasks can diminish the performance advantages of some task allocation mechanisms compared to others, and suggest that, under certain conditions, it may be possible to employ simple allocation mechanisms with low computational and communication costs to achieve competitive performance with more sophisticated allocation algorithms.

4. **A method of mechanism selection** (Chapter 8)
   I have developed a data-driven method of selecting a task allocation mechanism from among several options based on reading the arrangements of tasks and robots at the beginning of a mission. The method was inspired by the results of the empirical investigation mentioned above which suggest that, under certain conditions, it may be possible to employ a low cost task allocation mechanism that achieves performance that is competitive with, or better than, a more expensive alternative. Experiments show that selecting a task allocation mechanism using this method can significantly improve the performance of a robot team executing its mission compared to using any single mechanism in the portfolio.

The greater part of effort was spent developing the software infrastructure and work flow that made experimental investigation possible. I spent about a year developing MRTeAm to work with both simulated and physical robots and about half a year adapting its simulations to work on a high performance compute cluster (HPCC). The ability to run large numbers of simulations in parallel was vital for producing the large numbers of training data required by the mechanism selection method discussed in Chapter 8.

9.2 Discussion

The experiments presented in this thesis employed the same four task allocation mechanisms (Section 4.4) in missions set in a series of increasingly complex environments. The aim of the experiments was to understand factors that arise during execution of a mission and that affect performance in ways that might not be apparent at the time an allocation is made. The four mechanisms make different trade-offs between the quality of the solutions they provide and the costs of computing them. The round-robin (RR) mechanism is trivial to compute but produces arbitrarily poor allocations. The parallel single-item (PSI) auction is a simple, "greedy" algorithm that is inexpensive to compute, but can also produce arbitrarily poor allocations [58]. The ordered single-item (OSI) auction can take some inter-task synergies into account, but is strongly affected by the order in which tasks are auctioned [46]. The sequential-single item auction (SSI) has been proven to produce allocations that are close to optimal for some performance objectives [58], but carries the highest cost to compute its allocations.

Chapter 5 compared the performance of these mechanisms in a simple environment in which tasks were statically allocated and could be executed independently by single robots (SR-IT-SA). Experimental results show that SSI was indeed effective at producing efficient allocations that generally led to the best performance in practice, by most metrics. But an interesting result was observed when the starting locations of robots were spread out over the map ("distributed"), and the differences among the four mechanisms was diminished in terms of the time taken to execute the mission. The results of experiments in Chapter 6 showed that when the appearance of tasks was distributed over time (SR-IT-DA), the performance differences among three of the mechanisms diminished even further. In particular, the performance of PSI allocations during execution was indistinguishable from that of SSI while being much less expensive to compute. The experiments in Chapter 7, designed to determine if relative performance rankings of mechanisms would hold when the mechanisms were employed across different task environments (SR/MR-IT/CT-DA), showed that factors like multi-robot coordination and honouring precedence-ordering of tasks during mission execution made predictions of performance outcomes based on the expected efficiency of allocations alone difficult or impossible to confirm.

The method of mechanism selection presented in Chapter 8 was developed based on these results, especially those presented in Chapters 5 and 6. It was motivated by the idea that it might be possible to identify environments in which an inexpensive mechanism (like PSI) produces allocations that are competitive with a more sophisticated mechanism (like SSI) *a priori*, before actually conducting an allocation. The results of the method proposed in Chapter 8 are a proof of concept of this methodology and show that, in certain settings, mechanism selection can improve performance as compared to employing any one single mechanism alone.

9.3 Future work

The work presented in this thesis falls into two broad categories. The first is a comparative evaluation of task allocation mechanisms in practice as they are employed in different environments (Chapters 5–7). The second, mechanism selection (Chapter 8), attempts to exploit the results of the evaluation to choose a mechanism suited to the environment at hand from a number of options. The following sections suggest directions for future work in each category.

9.3.1 Evaluation

One natural direction of investigation is to increase the sizes of both the team and the missions they carry out and examine how scaling each affects both the costs of conducting allocations and the performance of executing missions in practice. There are practical limits to the number of robots that can be fielded in physical experiments, but larger scale experiments can be carried out in simulation. It would also be interesting to see if observations made on one map carry over (or not) to different maps, including randomly generated ones.

Other mechanisms should also be investigated. In particular, a benchmark of (predicted) optimal performance would provide a basis of comparison for other mechanisms that make trade-offs between solution quality and cost. A combinatorial auction mechanism can generate provably optimal allocations for different performance objectives by changing its winner determination function, although it may have problems scaling up with mission or team sizes. Besides a "benchmark" mechanism, the various extensions to SSI (lookaheads, bundle-bids [140], regret-clearing [139], or sequential single-cluster SSC auctions [46]), which each make different trade-offs to improve solution quality, would be interesting to investigate.

Collection and delivery tasks have clear applications for public and commercial transportation scheduling and warehouse automation, but have received little attention researchers of market-based MRTA (with some exceptions [47]). As autonomous vehicles become commonplace, this will be an exciting and perhaps important class of problem to investigate.

9.3.2 Mechanism Selection

A natural next step is to train and evaluate the performance of the mechanism selection method (SEL) in the multi-robot, constrained (MR-CT-DA) task environments investigated in Chapter 8, where it was observed that predictions about mission performance were difficult to make based on the efficiency of allocations alone.

The complete task graph used by SEL is expensive to compute, and run-time performance can be improved by constructing a more sparse, but still connected graph. The features used to train the SEL method were devised manually and may not have represented the properties of robot starting locations in relation to medians well enough to characterise task environments. A more principled (or automated) approach of developing training features might improve its classification performance. Features based on metrics of the task graph other than its medians may also help discriminate between environments better.

It would also be interesting to expand the portfolio of mechanisms that the SEL method can choose from to include the other two mechanisms investigated in this thesis, RR and OSI, as well as other methods like a combinatorial auction mechanism and extensions to the SSI mechanism discussed in Section 2.3.1.

9.4 Summary

The work presented in this thesis has focused on the relationship between auction-based task allocation mechanisms and properties of task environments, with the goal of developing a method of selecting, from a portfolio of options, a mechanism that is appropriate for a given task environment. The first part of this work was an empirical performance evaluation of a range of mechanisms employed in a series of environments of increasing complexity. The second part of this work used results from the evaluation to develop and train a data-driven method of mechanism selection using properties of environments that can be measured at the start of a mission. The results show that, under certain conditions, this method of mechanism selection can lead to significant performance improvements compared to using a single mechanism alone.

Part I

Appendices

Appendix A

Additional Results

A.1 SR-IT-SA

A.1.1 Experiment and results

An experiment similar to those discussed in Chapter 5 was conducted using the scenario and 2 starting configurations shown in Figures A.1–A.2. The experiment was conducted with MRTeAm using the Stage simulator (Section 3.6).

The four task allocation mechanisms discussed in Section 4.4 were employed in each of 2 missions with 15 trials for each combination, for a total of 120 experimental trials:

$$\begin{aligned} 2\,\mathit{missions} &= (2\ \mathit{starting\ configurations} \times 1\ \mathit{scenario}) \\ &\qquad \times 4\ \mathit{allocation\ mechanisms} \times 15\ \mathit{trials} \end{aligned}$$

The results are presented in the figures below.

Table A.1 shows mean values of metrics over 15 runs of each experimental configuration. The metrics discussed in Chapter 5 are presented followed by plots of robots' trajectories and timelines. Note that the implementation of SSI (Algorithm 3) used in the experiment presented here does attempt to optimise the MiniSum objective of an allocation using a task insertion heuristic during bid computation, as discussed in Section 4.4.

One point made in Section 5.4 can be corroborated by the results presented here. In the case of clustered start locations, OSI leads to significantly lower run times and team distances than SSI. This seems to be due to inter-robot interference seen in the higher number of near collisions and associated delay time in SSI allocations as compared to OSI (note the delay time of Robot 2 in Figure A.7). Another factor is scalability. Bid computation for both OSI and ssi is more expensive in MRTeAm than HRTeam due to the task insertion heuristic it uses. The RR and PSI mechanisms do not calculate task insertion, and this difference can be seen in comparisons of deliberation time.

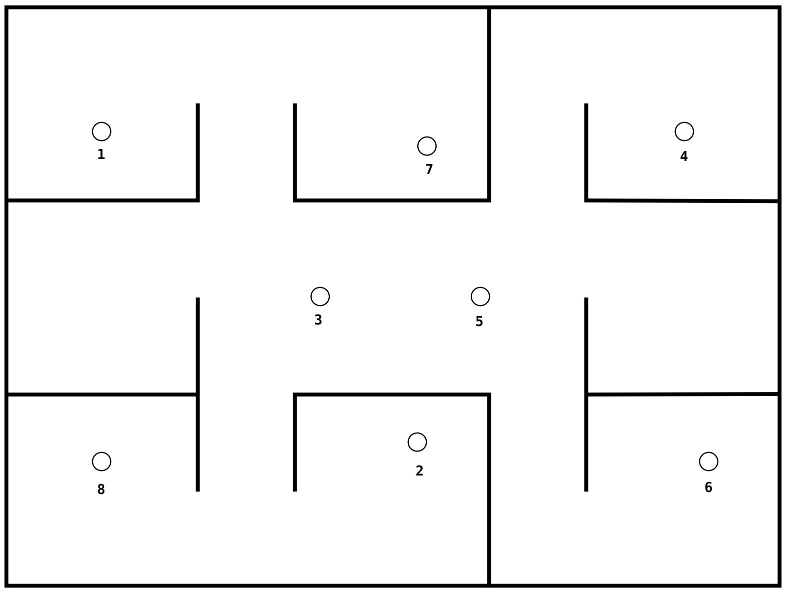
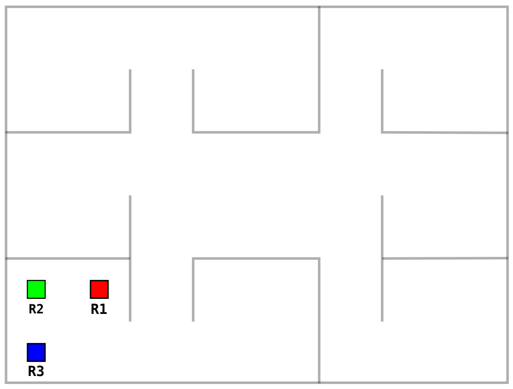
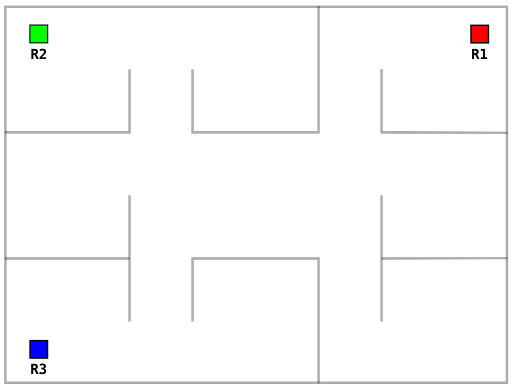


FIGURE A.1: An SR-IT-SA scenario



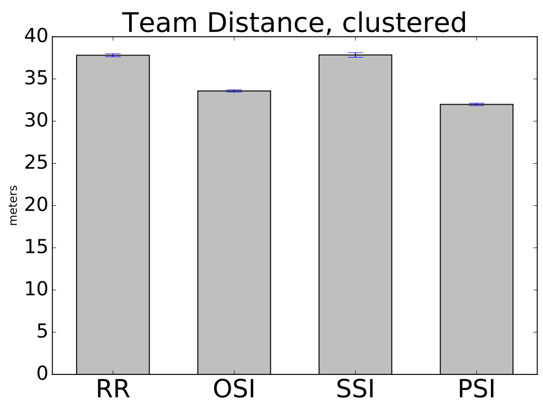
(a) Clustered

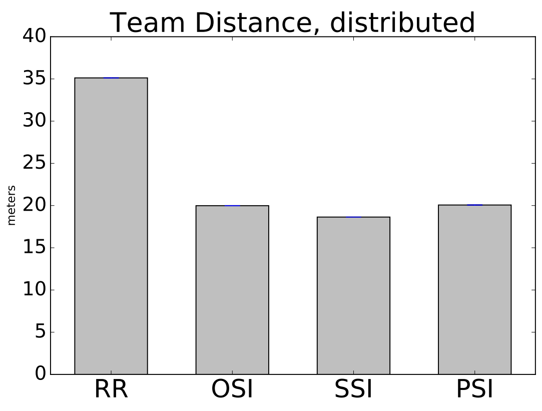


(b) Distributed

FIGURE A.2: Starting locations of robots, clustered (left) and distributed (right)

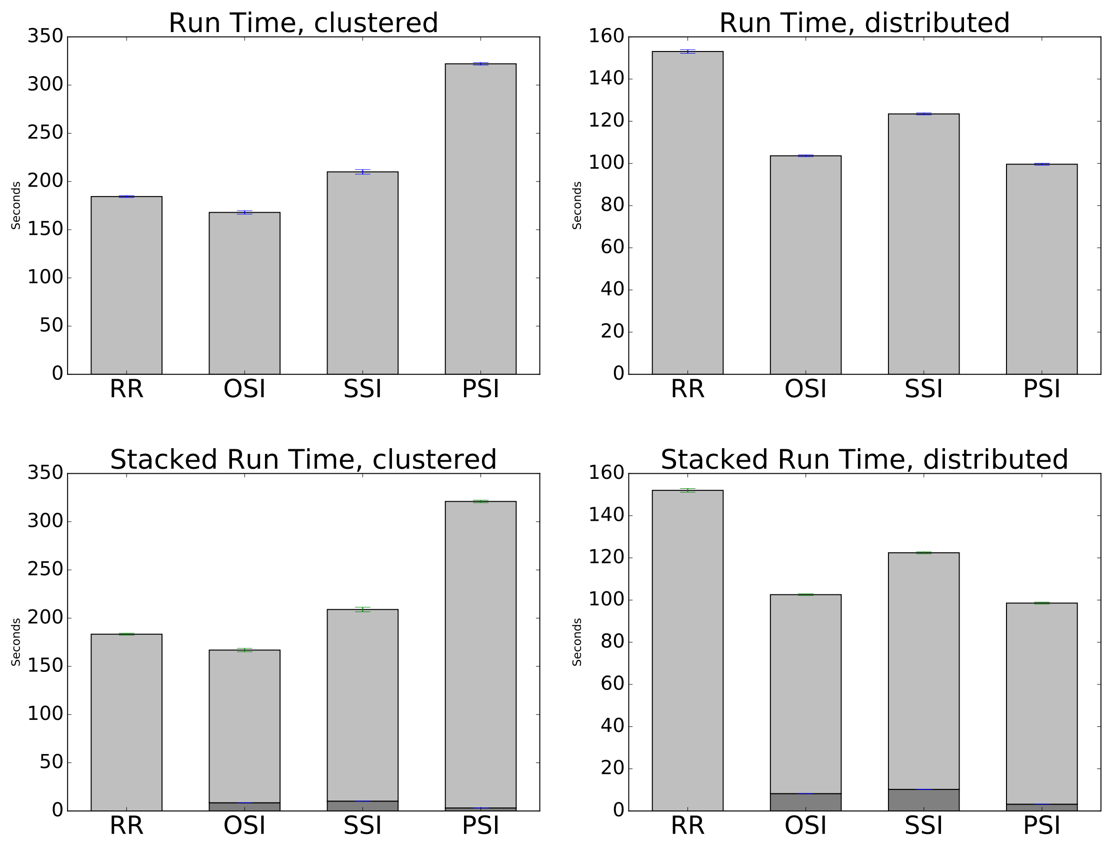
A.1.2 SR-IT-SA Results

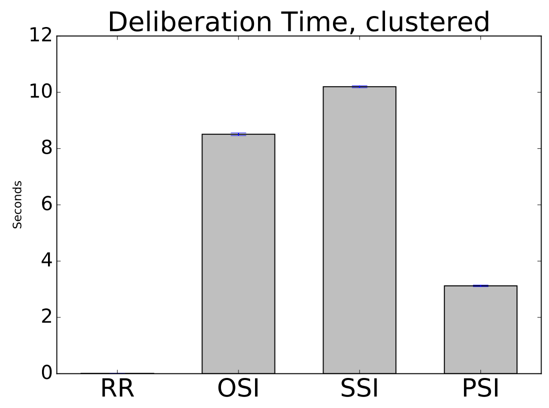


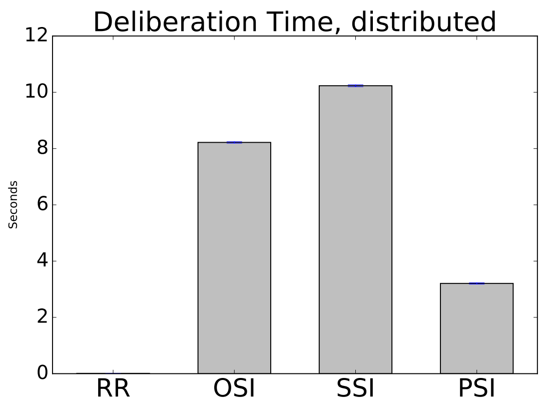


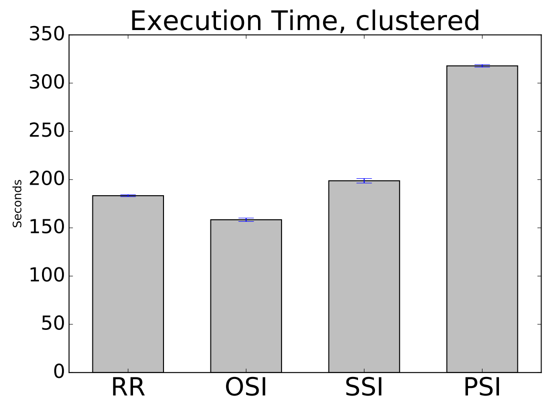
		Distance	Run time	Delib. time	Delay time
Clustered	RR	37.8 ± 0.38	184.39 ± 2.14	**0.0113 ± 0.0011**	17.47 ± 3.72
	OSI	33.58 ± 0.28	**167.92 ± 3.81**	8.51 ± 0.1	19.67 ± 3.29
	SSI	37.84 ± 0.61	210.0 ± 5.18	10.2 ± 0.08	44.69 ± 9.49
	PSI	**31.99 ± 0.32**	322.06 ± 2.77	3.12 ± 0.06	**8.57 ± 1.65**
Distributed	RR	35.12 ± 0.08	153.06 ± 1.85	**0.0104 ± 0.0018**	35.15 ± 1.62
	OSI	19.99 ± 0.06	103.61 ± 0.88	8.22 ± 0.06	**0.0 ± 0.0**
	SSI	**18.64 ± 0.05**	123.47 ± 1.01	10.23 ± 0.07	**0.0 ± 0.0**
	PSI	20.07 ± 0.1	**99.61 ± 0.99**	3.21 ± 0.05	**0.0 ± 0.0**

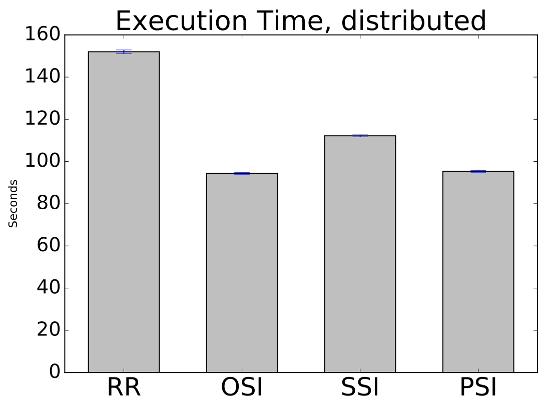
Table A.1: Metrics for the SR-IT-SA scenario shown in Figure A.1
Metrics for the SR-IT-SA scenario shown in Figure A.1. The values given are means with 95% confidence intervals.

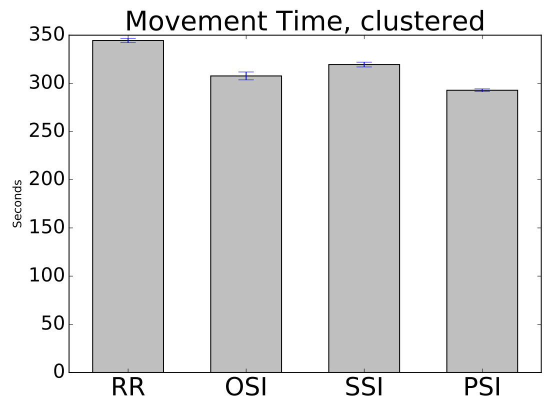


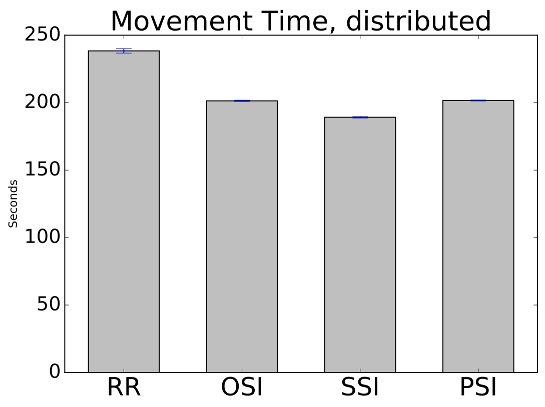

Deliberation Time, clustered
Seconds
0
2
4
6
8
10
12
RR
OSI
SSI
PSI

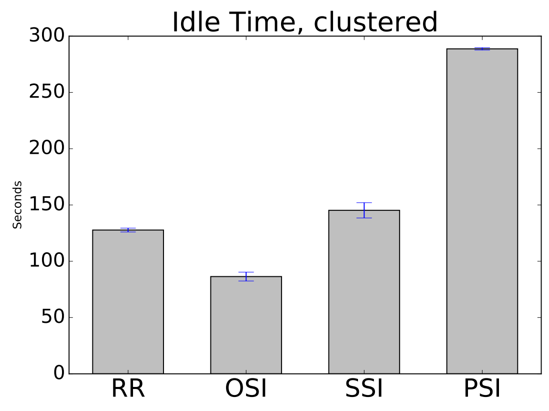

Deliberation Time, distributed
Seconds
0
2
4
6
8
10
12
RR
OSI
SSI
PSI

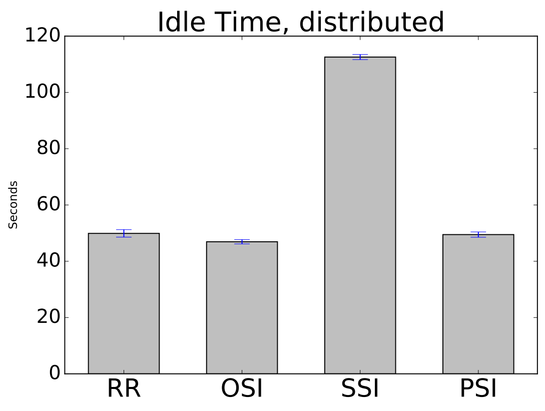

Execution Time, clustered
Seconds
0
50
100
150
200
250
300
350
RR
OSI
SSI
PSI

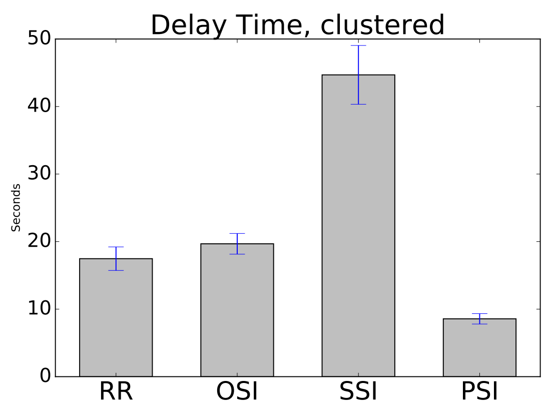

Execution Time, distributed
Seconds
0
20
40
60
80
100
120
140
160
RR
OSI
SSI
PSI

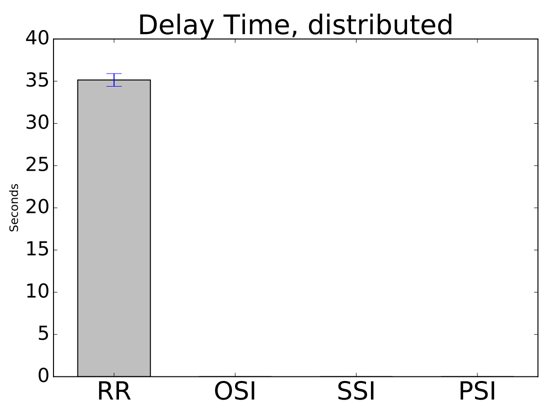

Movement Time, clustered
Seconds
0
50
100
150
200
250
300
350
RR
OSI
SSI
PSI

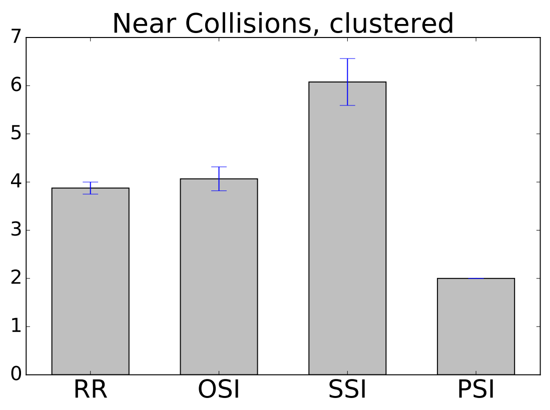

Movement Time, distributed
Seconds
0
50
100
150
200
250
RR
OSI
SSI
PSI

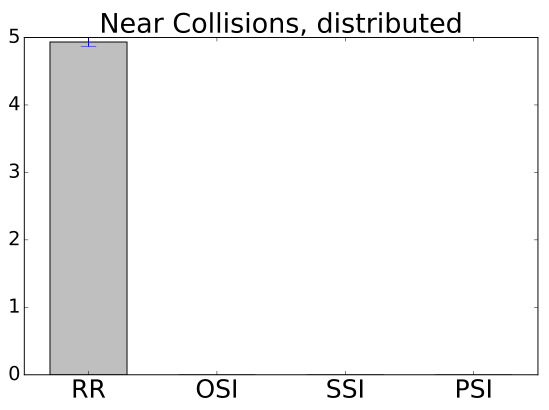

Idle Time, clustered
Seconds
0
50
100
150
200
250
300
RR
OSI
SSI
PSI

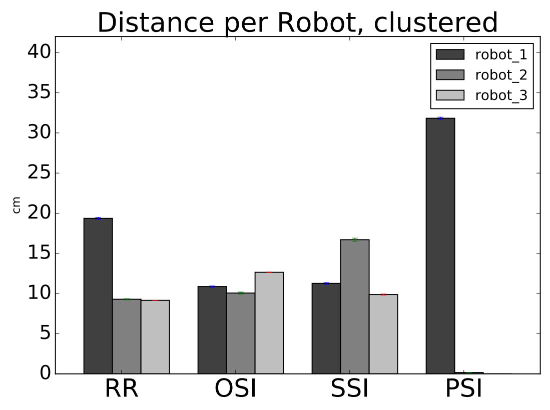

Idle Time, distributed
Seconds
0
20
40
60
80
100
120
RR
OSI
SSI
PSI

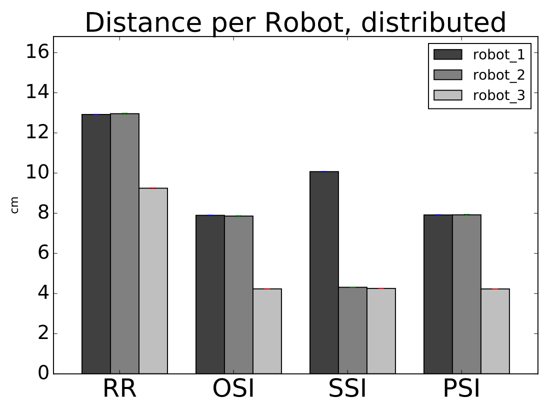

Delay Time, clustered
Seconds
RR
OSI
SSI
PSI

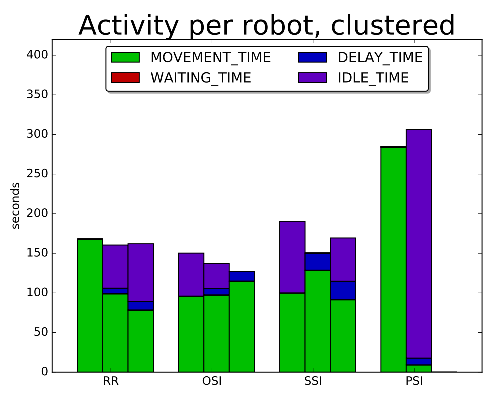

Delay Time, distributed
Seconds
RR
OSI
SSI
PSI

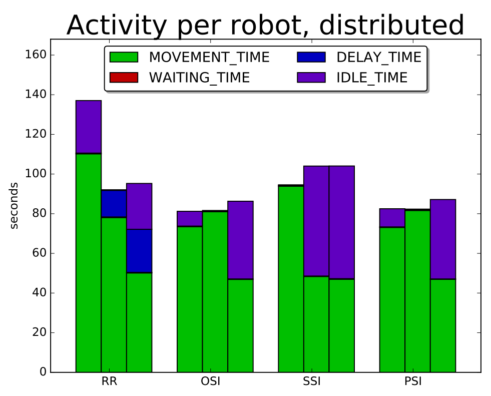

Near Collisions, clustered
RR
OSI
SSI
PSI

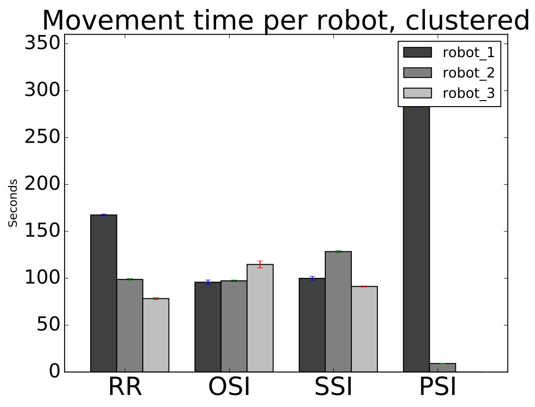

Near Collisions, distributed
RR
OSI
SSI
PSI

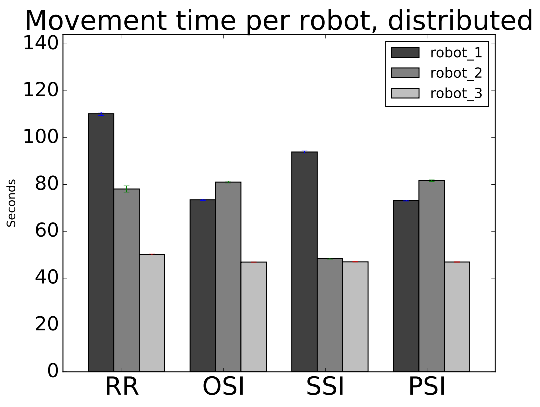

Distance per Robot, clustered
cm
robot_1
robot_2
robot_3
RR
OSI
SSI
PSI

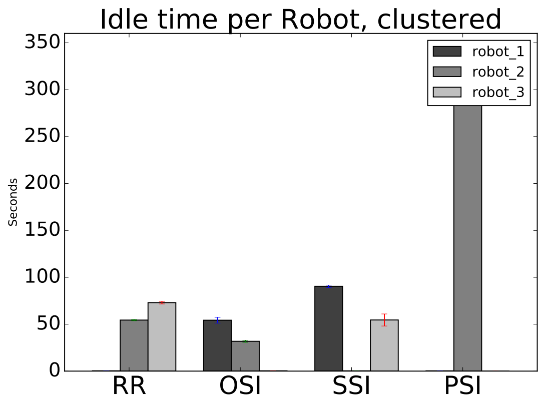

Distance per Robot, distributed
cm
robot_1
robot_2
robot_3
RR
OSI
SSI
PSI

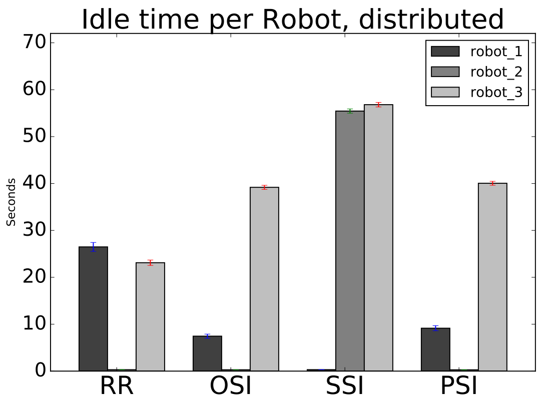

Activity per robot, clustered
MOVEMENT_TIME
DELAY_TIME
WAITING_TIME
IDLE_TIME
seconds
RR
OSI
SSI
PSI

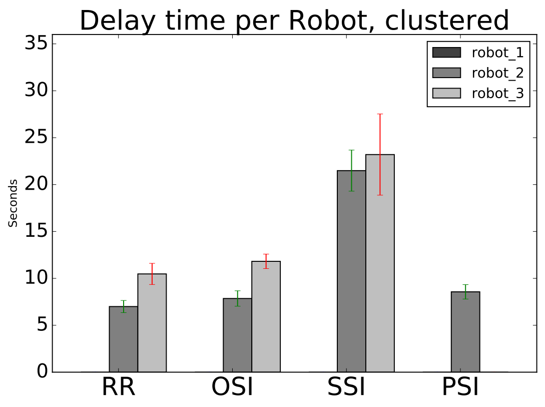

Activity per robot, distributed
MOVEMENT_TIME
DELAY_TIME
WAITING_TIME
IDLE_TIME
seconds
RR
OSI
SSI
PSI

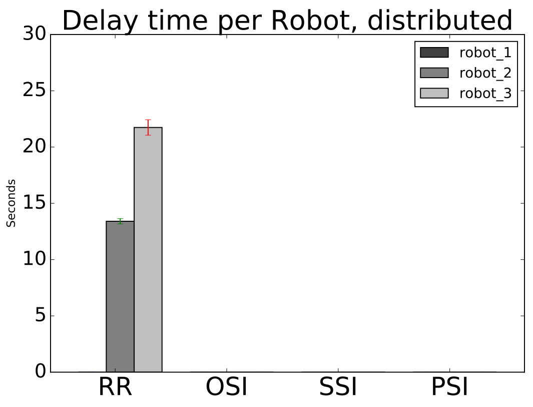

Movement time per robot, clustered
Seconds
robot_1
robot_2
robot_3
RR
OSI
SSI
PSI


Movement time per robot, distributed
Seconds
robot_1
robot_2
robot_3
RR
OSI
SSI
PSI


Idle time per Robot, clustered
Seconds
robot_1
robot_2
robot_3
RR
OSI
SSI
PSI


Idle time per Robot, distributed
Seconds
robot_1
robot_2
robot_3
RR
OSI
SSI
PSI


Delay time per Robot, clustered
Seconds
robot_1
robot_2
robot_3
RR
OSI
SSI
PSI


Delay time per Robot, distributed
Seconds
robot_1
robot_2
robot_3
RR
OSI
SSI
PSI

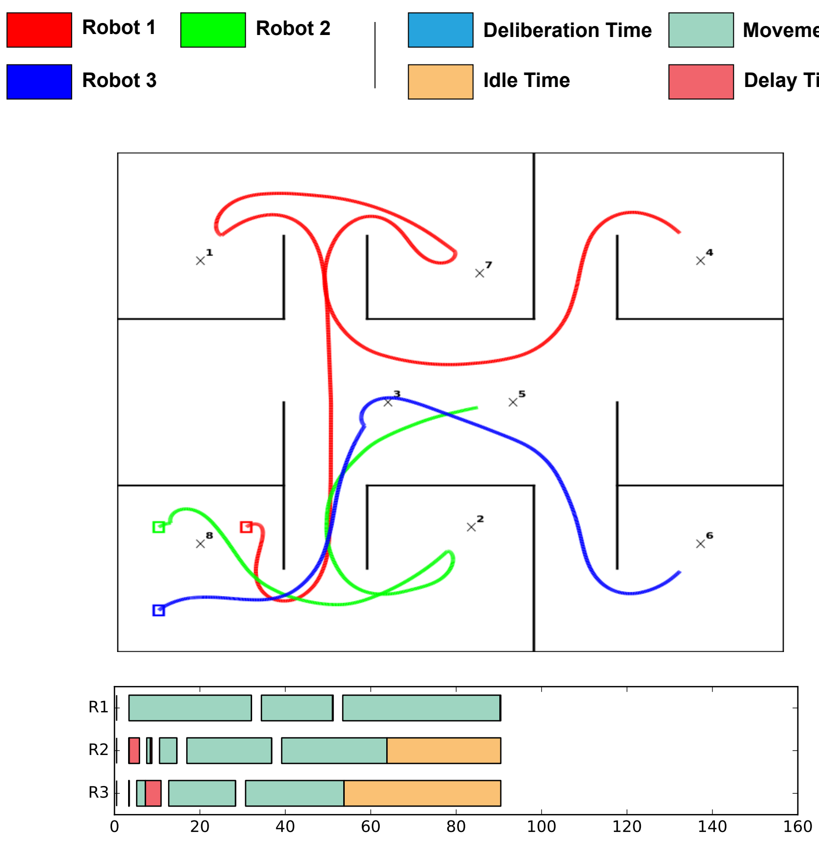


FIGURE A.3: Robot trajectories for RR, clustered starting locations

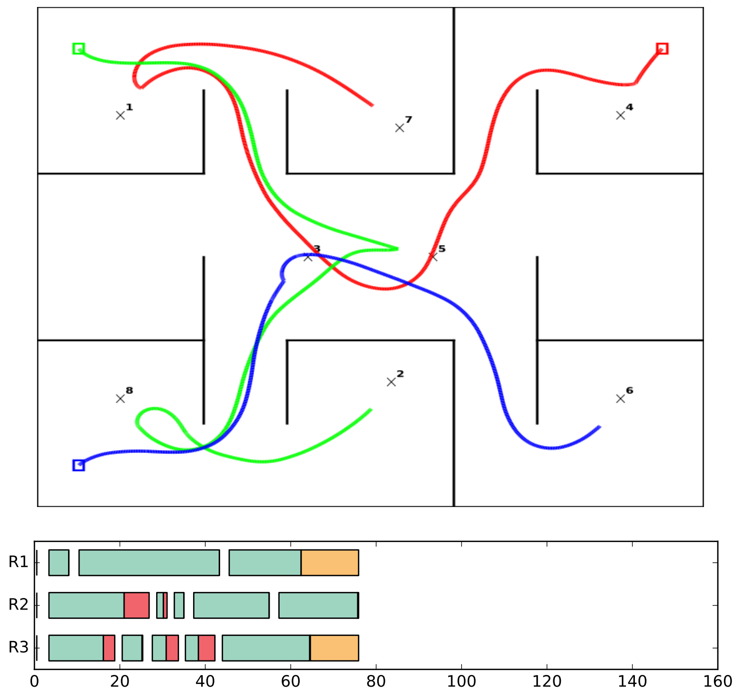


FIGURE A.4: Robot trajectories for RR, distributed starting locations

🟥 Robot 1 🟩 Robot 2 🟦 Robot 3

🟦 Deliberation Time 🟩 Movement Time 🟧 Idle Time 🟥 Delay Time

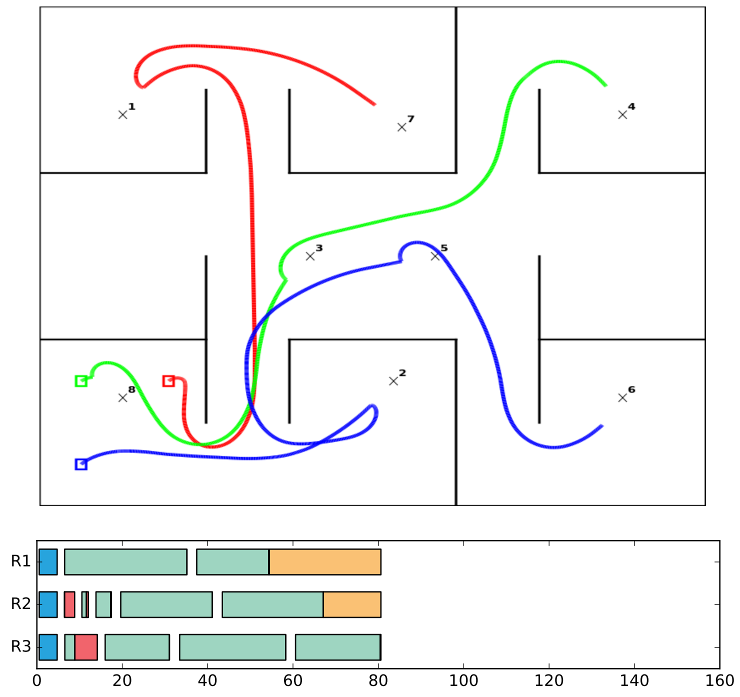


FIGURE A.5: Robot trajectories for OSI, clustered starting locations

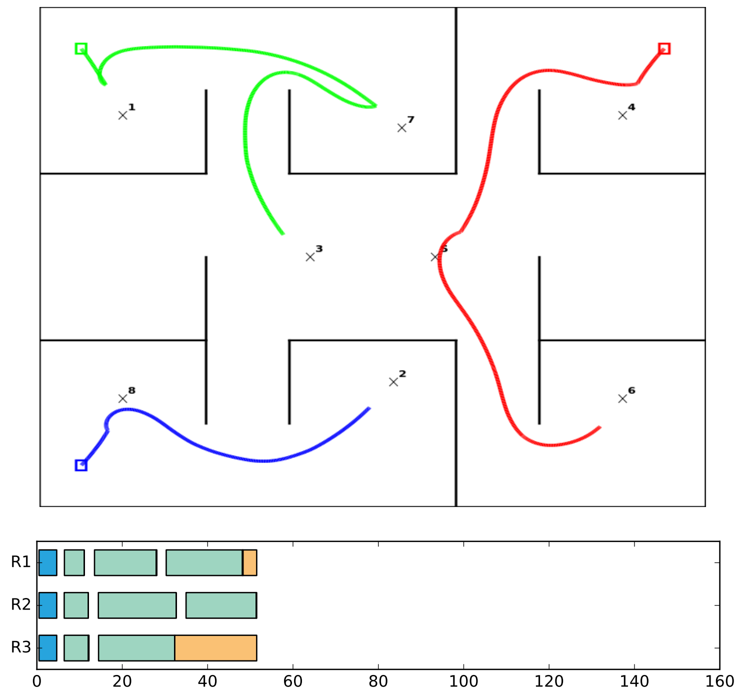
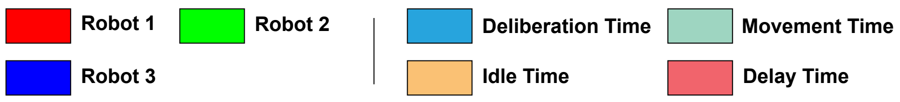


FIGURE A.6: Robot trajectories for OSI, distributed starting locations



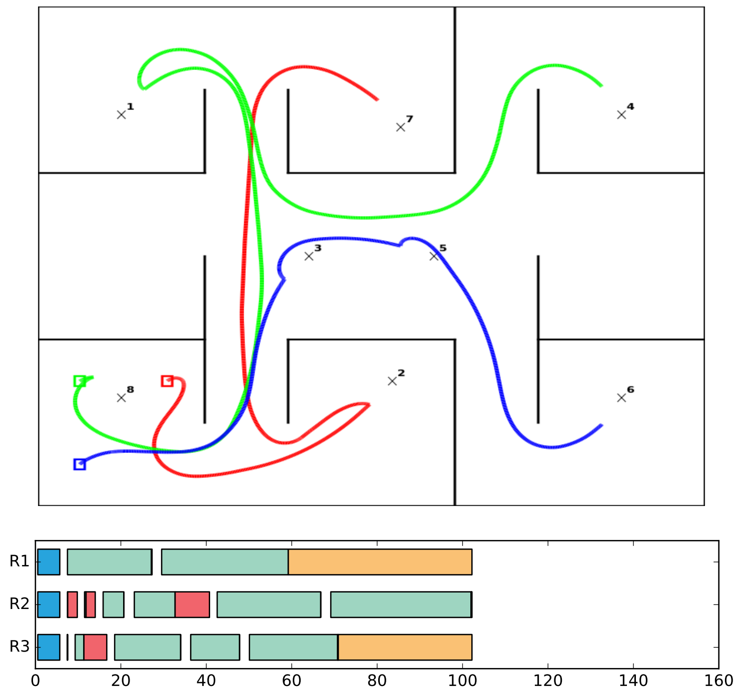


FIGURE A.7: Robot trajectories for SSI, clustered starting locations

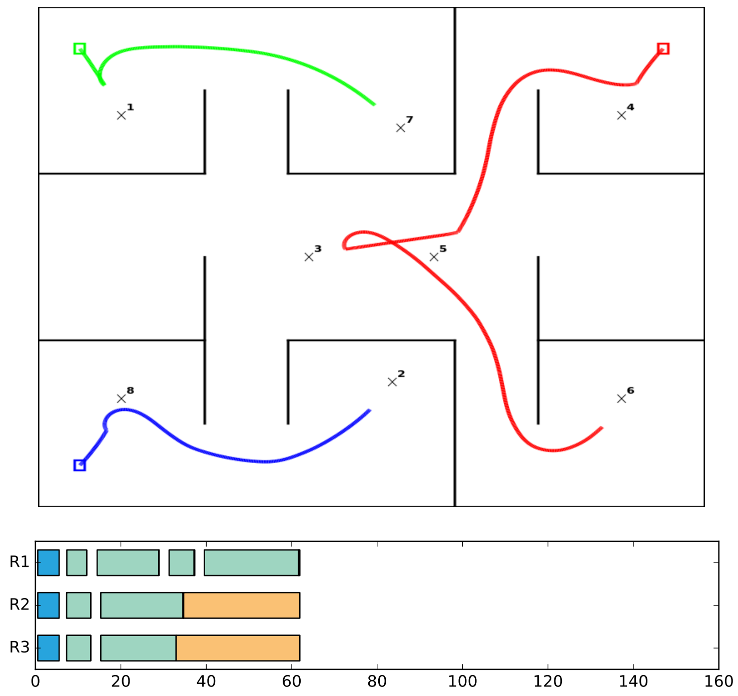


FIGURE A.8: Robot trajectories for SSI, distributed starting locations

🟥 Robot 1 🟩 Robot 2 🟦 Robot 3 | 🟦 Deliberation Time 🟩 Movement Time 🟧 Idle Time 🟥 Delay Time

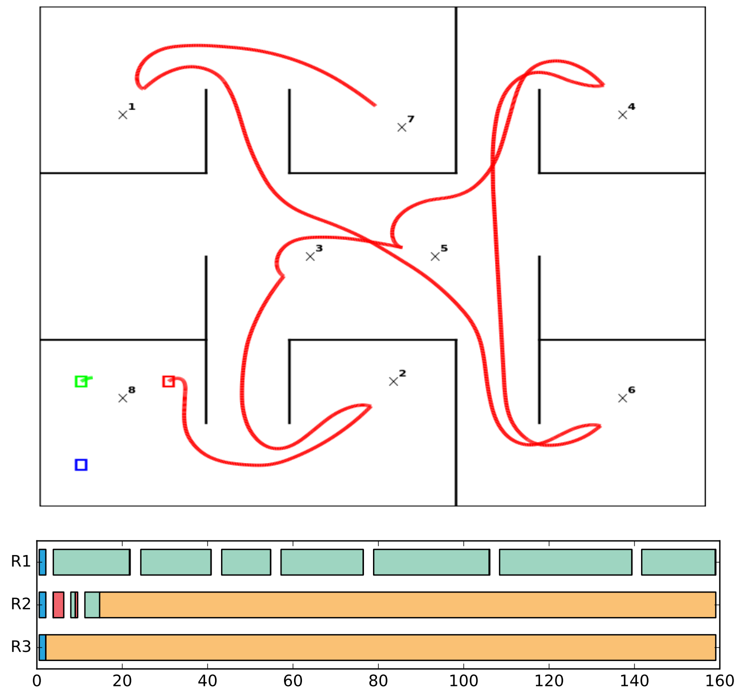


FIGURE A.9: Robot trajectories for PSI, clustered starting locations

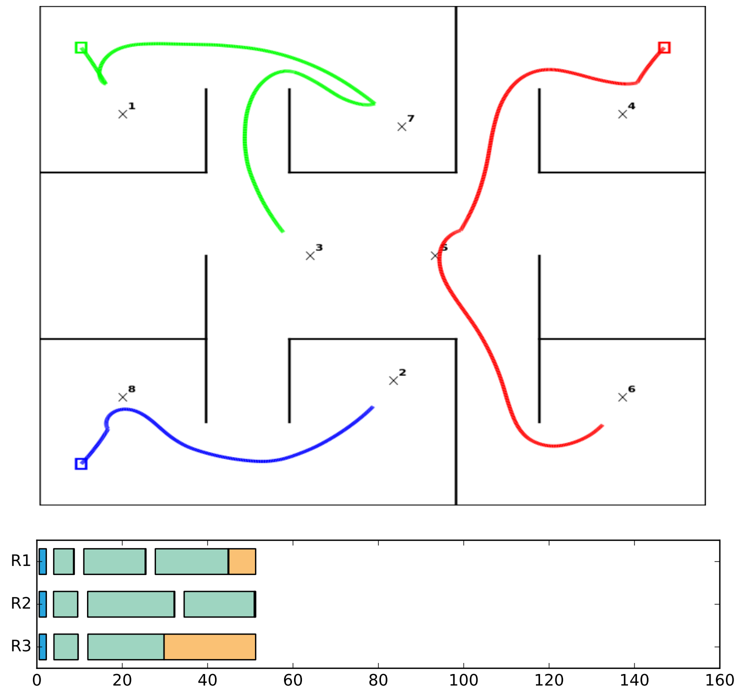


FIGURE A.10: Robot trajectories for PSI, distributed starting locations

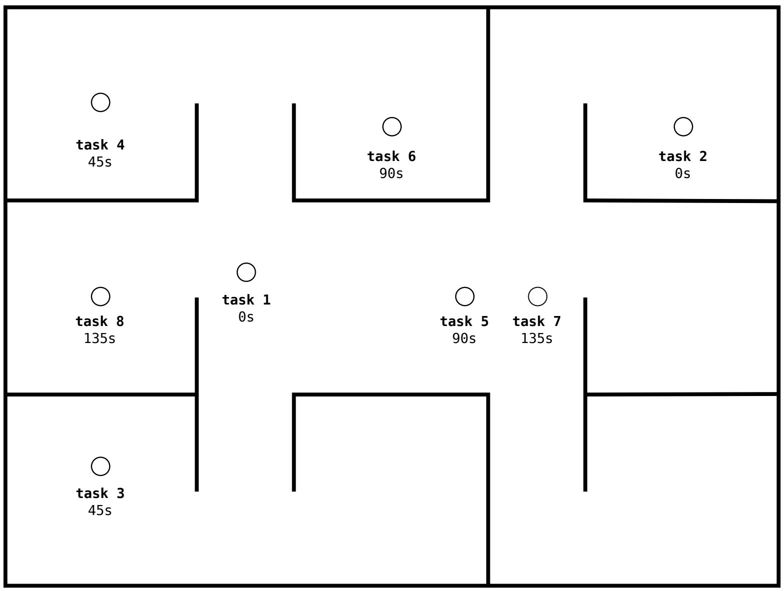
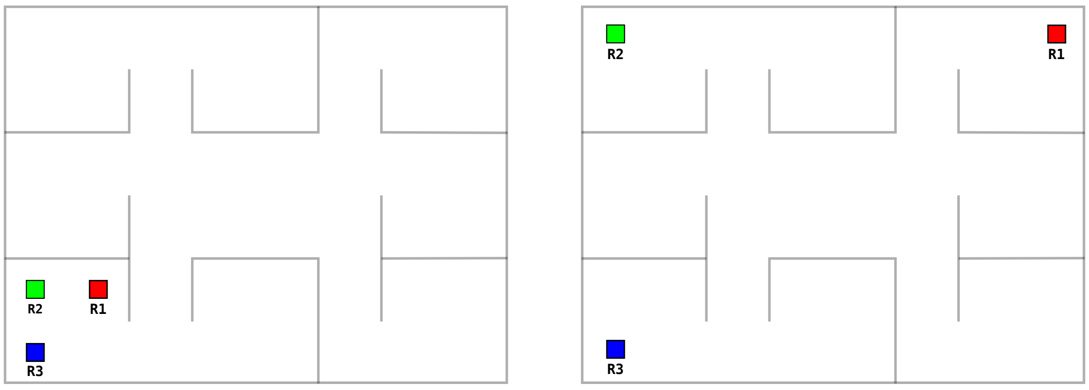


FIGURE A.11: An SR-IT-DA scenario



(a) Clustered (b) Distributed

FIGURE A.12: Starting locations of robots, clustered (left) and distributed (right)

A.2 SR-IT-DA

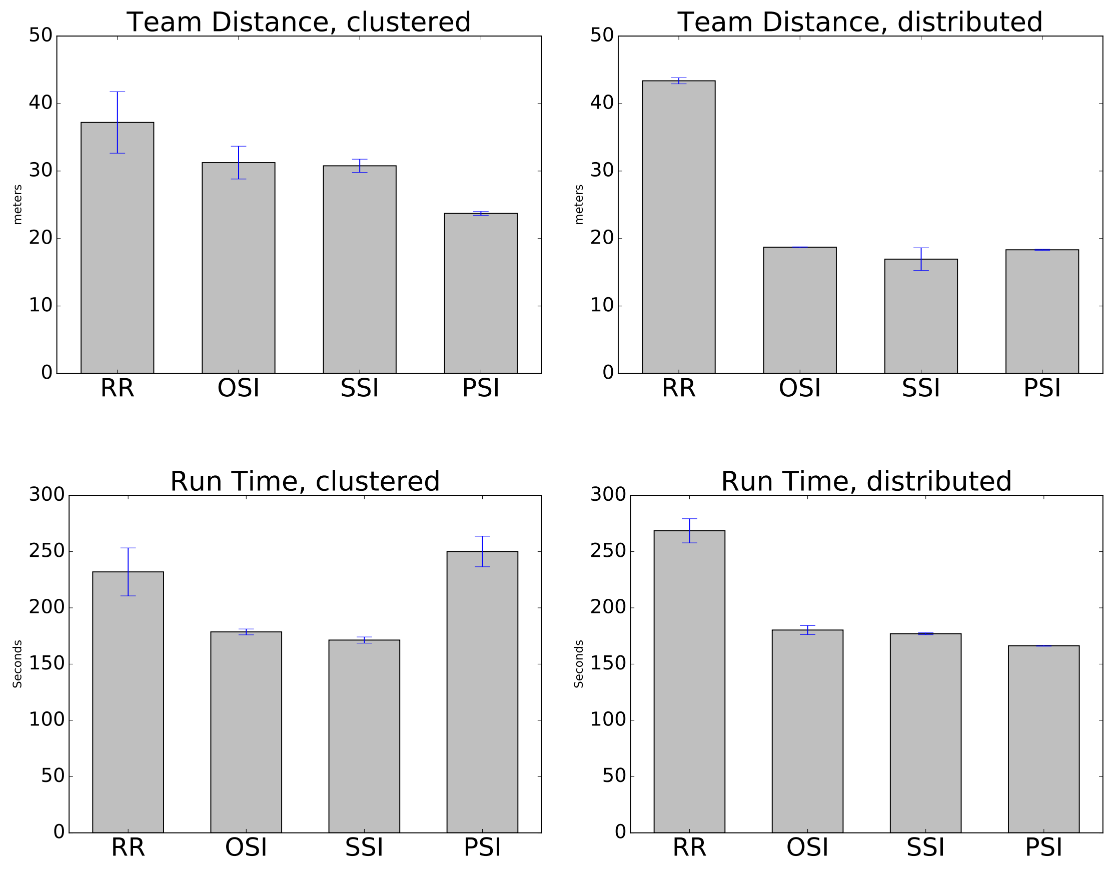
A.2.1 Experiment and results

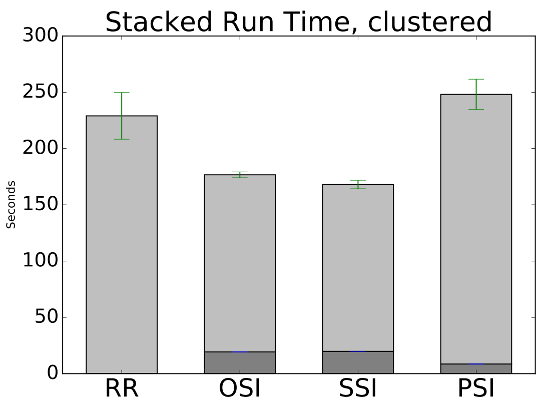
An experiment similar to those discussed in Chapter 6 was conducted using the scenario and start locations shown in Figures A.11–A.12. The scenario is dynamic (SR-IT-DA) and is not compared with a static counterpart (SR-IT-SA). The experiment was conducted with MRTeAm on **physical robots** and the results are presented in the figures below. The results show mean values of metrics over 3 runs of each experimental configuration. Although the number of runs was small, the results are similar those those in Chapter 6, showing that in a dynamic scenario (SR-IT-DA), the performance advantages of SSI over PSI are diminished, particularly when robots started from the *distributed* starting locations.

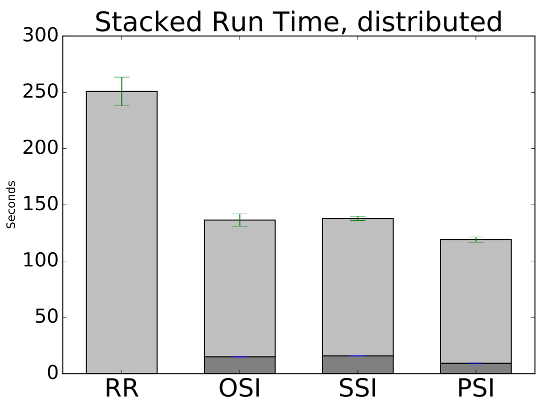
		Distance	Run time	Delib. time	Delay time
Clustered	RR	37.2 ± 14.52	231.97 ± 67.83	**0.0383 ± 0.0034**	58.53 ± 44.58
	OSI	31.25 ± 10.46	178.62 ± 11.19	19.31 ± 0.11	24.3 ± 14.88
	SSI	30.78 ± 4.21	**171.36 ± 11.89**	19.86 ± 0.14	22.8 ± 0.25
	PSI	**23.72 ± 1.21**	250.09 ± 58.34	8.58 ± 0.38	**0.0 ± 0.0**
Distributed	RR	43.37 ± 1.95	268.52 ± 46.18	**0.0378 ± 0.0019**	91.2 ± 30.43
	OSI	18.71 ± 0.25	180.28 ± 17.25	15.01 ± 0.9	**0.0 ± 0.0**
	SSI	**16.95 ± 7.24**	176.98 ± 3.72	15.81 ± 0.38	**0.0 ± 0.0**
	PSI	18.33 ± 0.34	**166.28 ± 1.8**	9.25 ± 0.62	**0.0 ± 0.0**

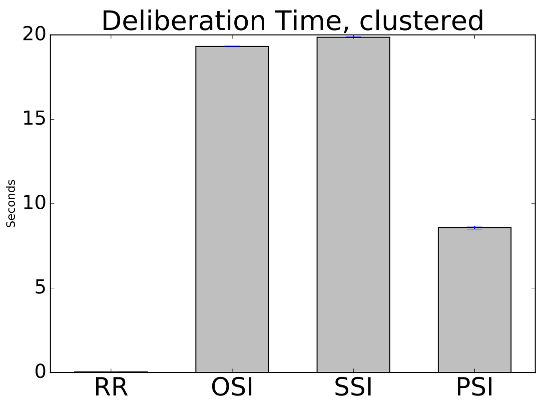
Table A.2: Metrics for the SR-IT-DA scenario shown in Figure A.11
Metrics for the SR-IT-DA scenario shown in Figure A.11. The values given are means with 95% confidence intervals.

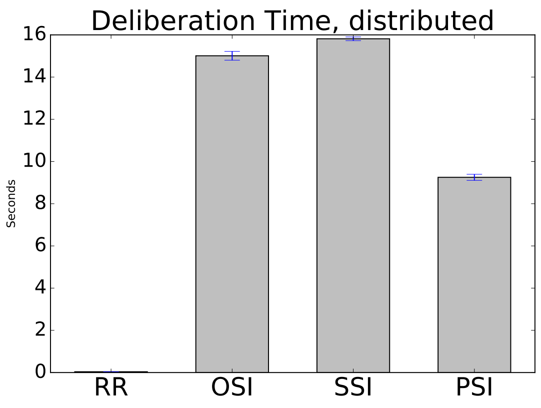
A.2.2 SR-IT-DA Results

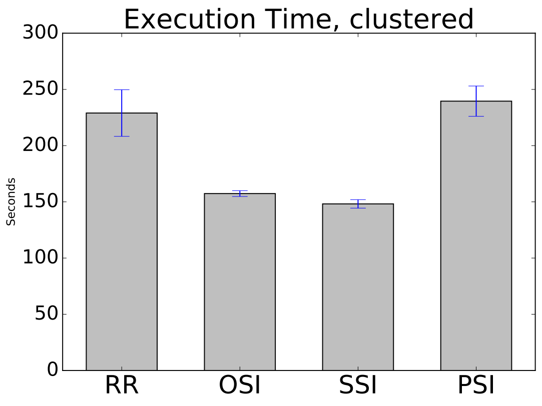


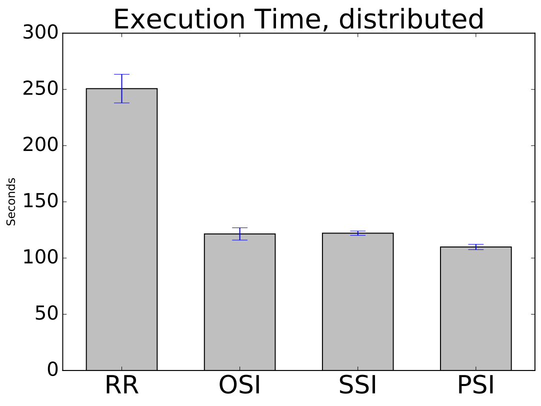

Stacked Run Time, clustered
Seconds
RR
OSI
SSI
PSI

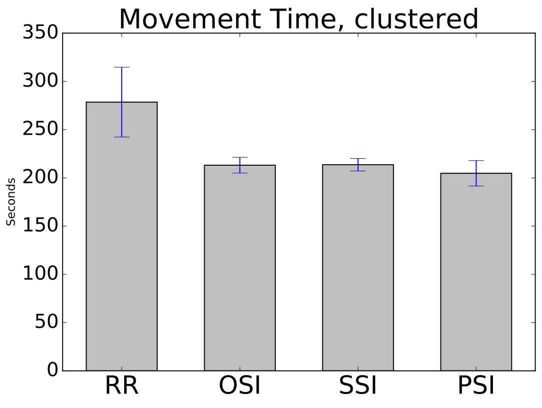

Stacked Run Time, distributed
Seconds
RR
OSI
SSI
PSI

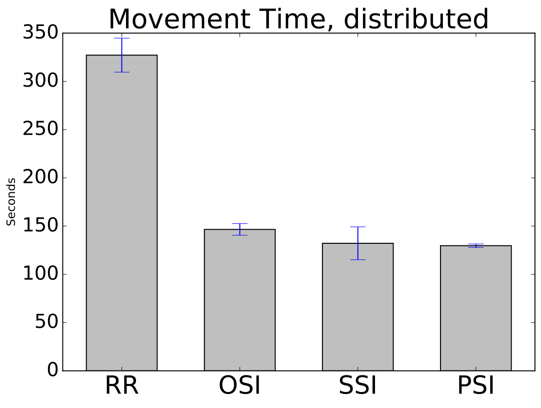

Deliberation Time, clustered
Seconds
RR
OSI
SSI
PSI

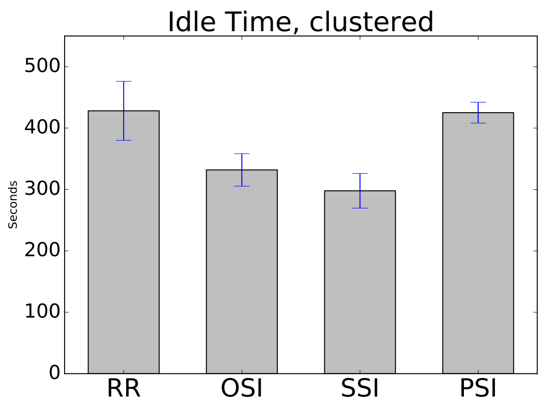

Deliberation Time, distributed
Seconds
RR
OSI
SSI
PSI

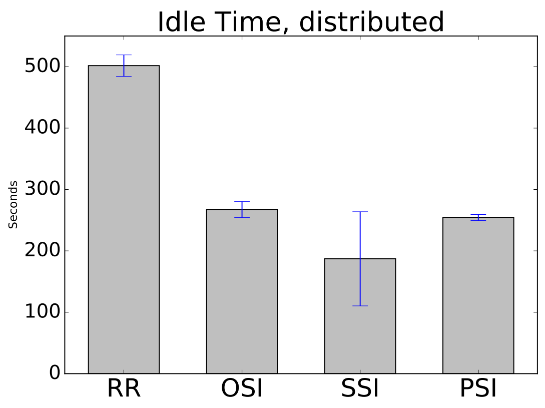

Execution Time, clustered
Seconds
RR
OSI
SSI
PSI

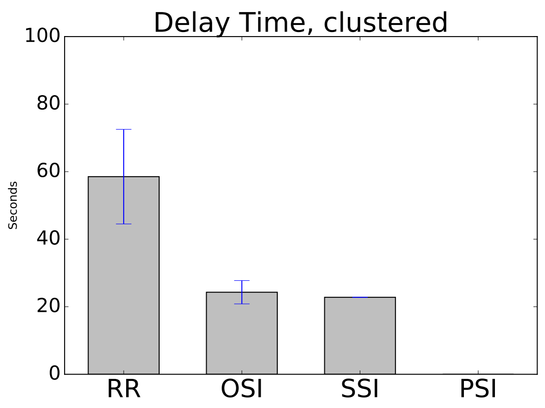

Execution Time, distributed
Seconds
RR
OSI
SSI
PSI

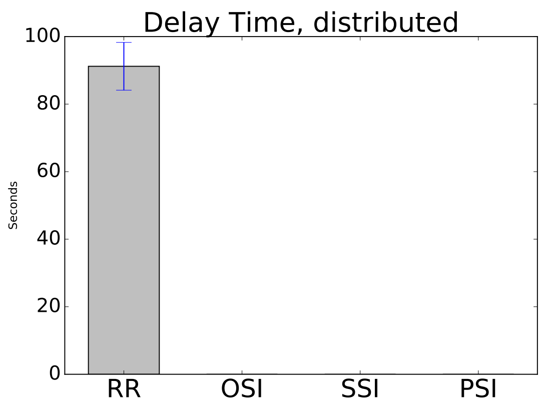

Movement Time, clustered
Seconds
RR
OSI
SSI
PSI

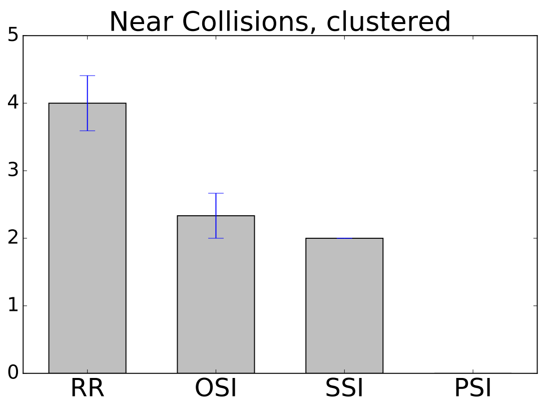

Movement Time, distributed
Seconds
RR
OSI
SSI
PSI

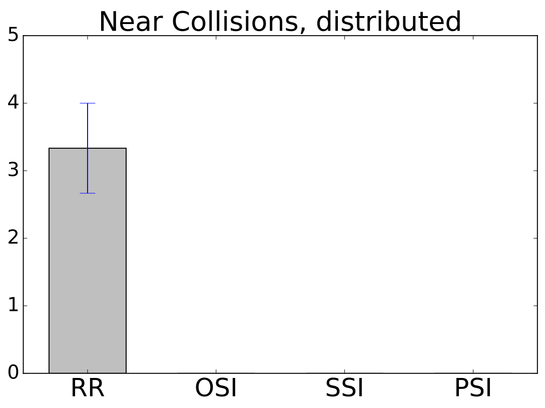

Idle Time, clustered
Seconds
0
100
200
300
400
500
RR
OSI
SSI
PSI

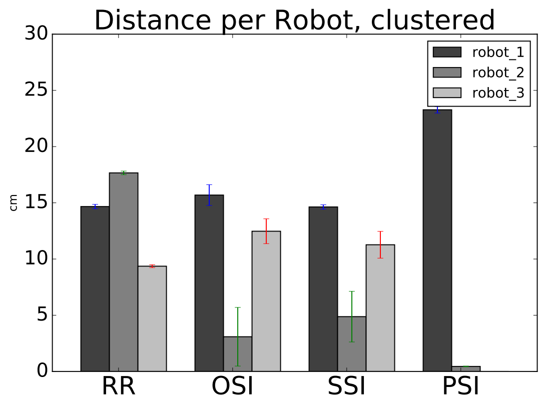

Idle Time, distributed
Seconds
0
100
200
300
400
500
RR
OSI
SSI
PSI

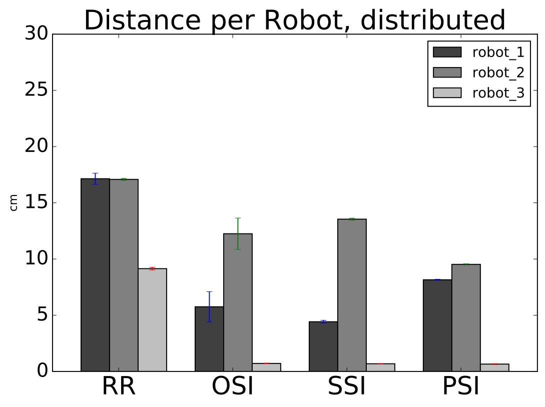

Delay Time, clustered
Seconds
0
20
40
60
80
100
RR
OSI
SSI
PSI

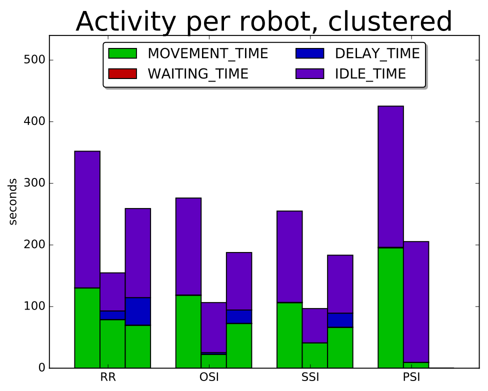

Delay Time, distributed
Seconds
0
20
40
60
80
100
RR
OSI
SSI
PSI

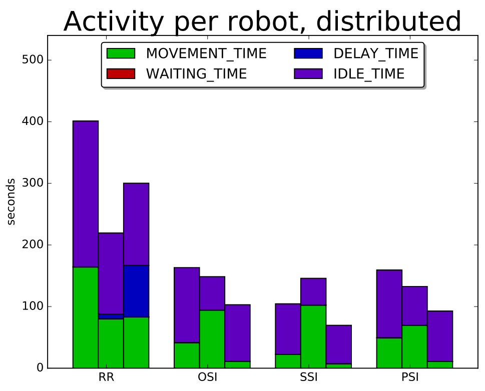

Near Collisions, clustered
0
1
2
3
4
5
RR
OSI
SSI
PSI

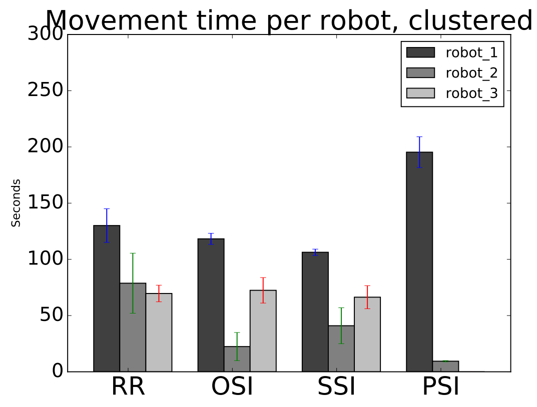

Near Collisions, distributed
0
1
2
3
4
5
RR
OSI
SSI
PSI

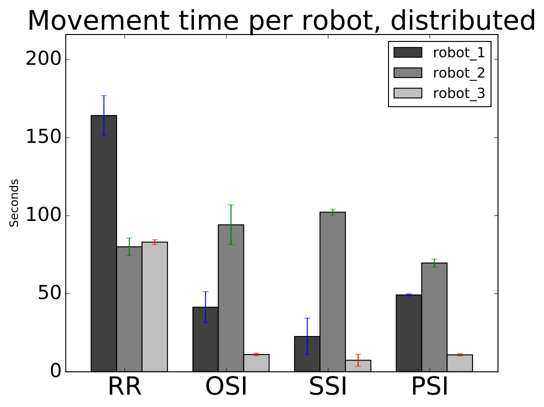

Distance per Robot, clustered
robot_1
robot_2
robot_3
cm
0
5
10
15
20
25
30
RR
OSI
SSI
PSI

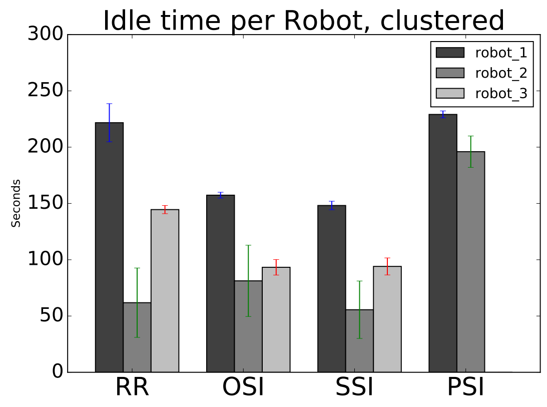

Distance per Robot, distributed
robot_1
robot_2
robot_3
cm
0
5
10
15
20
25
30
RR
OSI
SSI
PSI

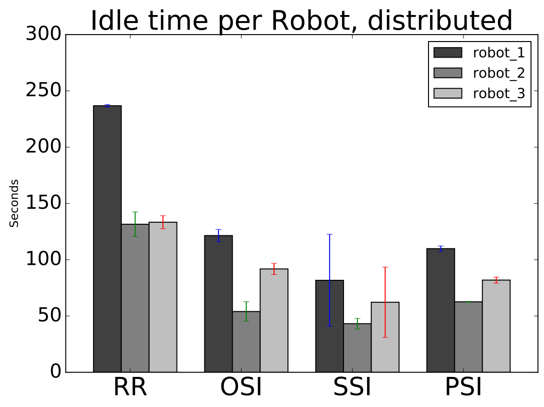

Activity per robot, clustered
MOVEMENT_TIME
WAITING_TIME
DELAY_TIME
IDLE_TIME
seconds
0
100
200
300
400
500
RR
OSI
SSI
PSI

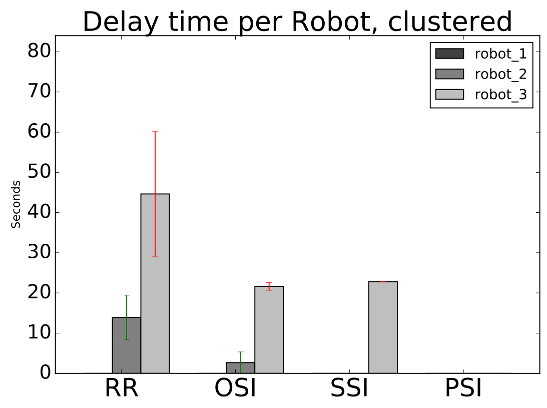

Activity per robot, distributed
MOVEMENT_TIME
WAITING_TIME
DELAY_TIME
IDLE_TIME
seconds
0
100
200
300
400
500
RR
OSI
SSI
PSI

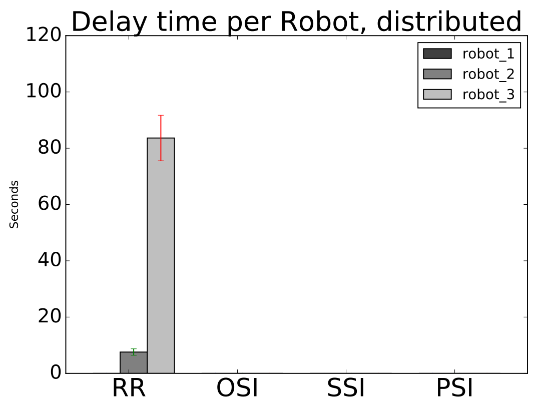

Movement time per robot, clustered
robot_1
robot_2
robot_3
Seconds
0
50
100
150
200
250
300
RR
OSI
SSI
PSI

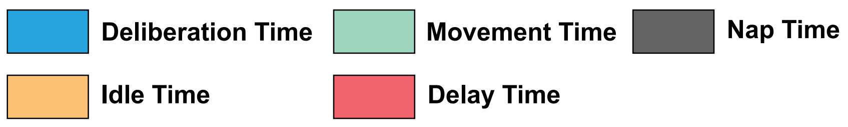

Movement time per robot, distributed
robot_1
robot_2
robot_3
Seconds
0
50
100
150
200
RR
OSI
SSI
PSI


Idle time per Robot, clustered
robot_1
robot_2
robot_3
Seconds
0
50
100
150
200
250
300
RR
OSI
SSI
PSI

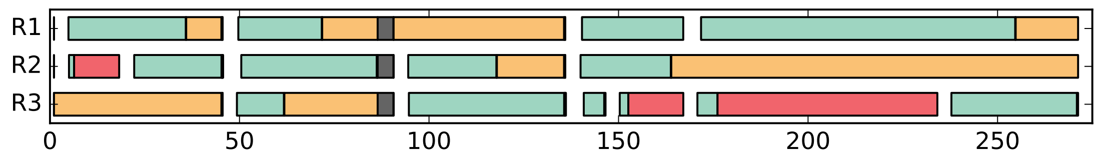

Idle time per Robot, distributed
robot_1
robot_2
robot_3
Seconds
0
50
100
150
200
250
300
RR
OSI
SSI
PSI


Delay time per Robot, clustered
robot_1
robot_2
robot_3
Seconds
0
10
20
30
40
50
60
70
80
RR
OSI
SSI
PSI

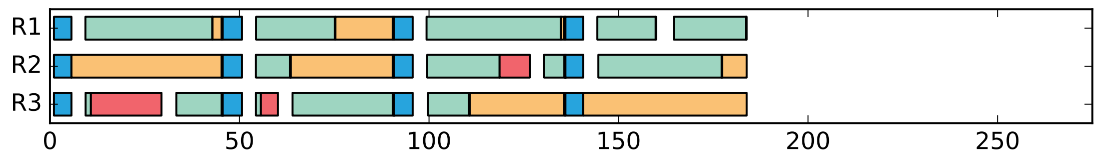

Delay time per Robot, distributed
robot_1
robot_2
robot_3
Seconds
0
20
40
60
80
100
120
RR
OSI
SSI
PSI



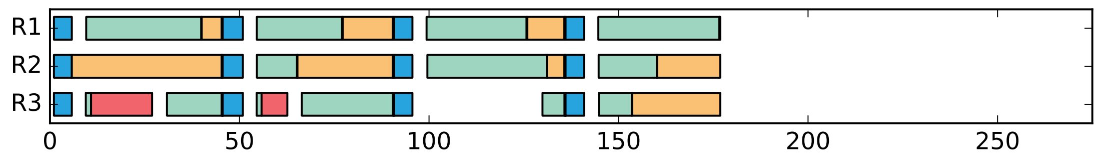
(a) RR



(b) OSI



(c) SSI



(d) PSI

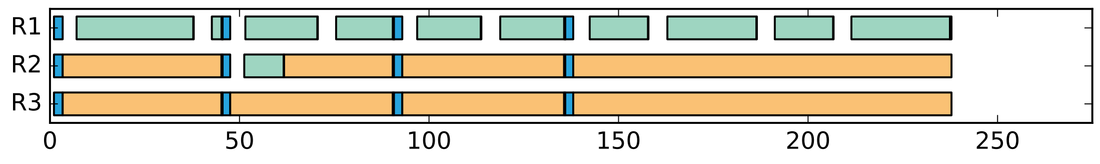


FIGURE A.13: Timelines showing robot activity in a single trial of a SR-IT-DA scenario for **clustered** start locations.

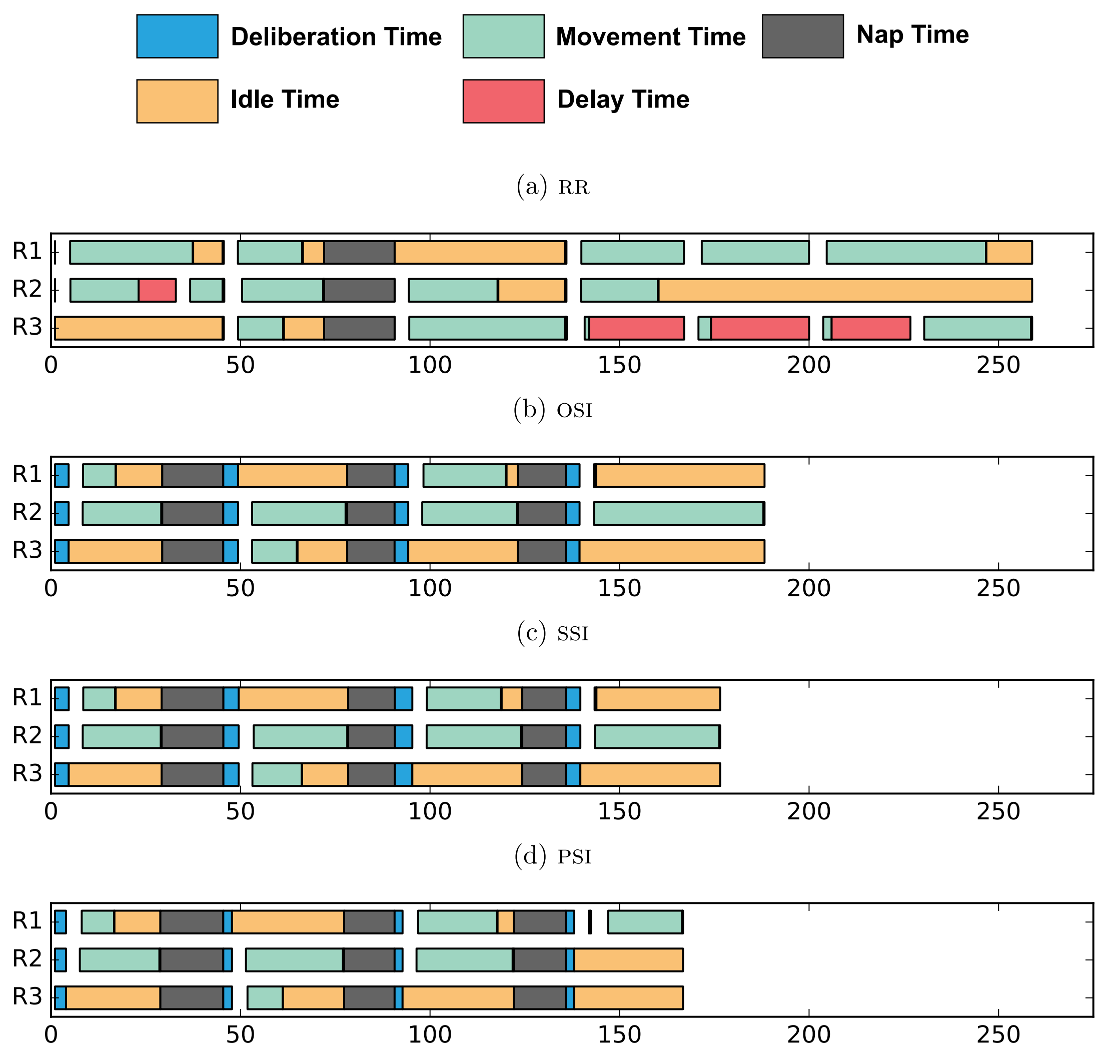


FIGURE A.14: Timelines showing robot activity in a single trial of a SR-IT-DA scenario for **distributed** start locations.

Appendix B

System Architecture

B.1 Components (state machines)

B.1.1 Robot Controller

The robot controller's state machine is shown in Figure B.2.

B.1.2 Auctioneer

The auctioneer's state machine is shown in Figure B.3.

B.1.3 Task Representation

Listing 1 Task Description format

```
- task_id: <string>
  type: SENSOR_SWEEP
  location:
    x: <float>
    y: <float>
  arrival_time: <float>
  num_robots: <int>
  duration: <float>
  depends: [<list of strings>]
```

B.2 Message definitions

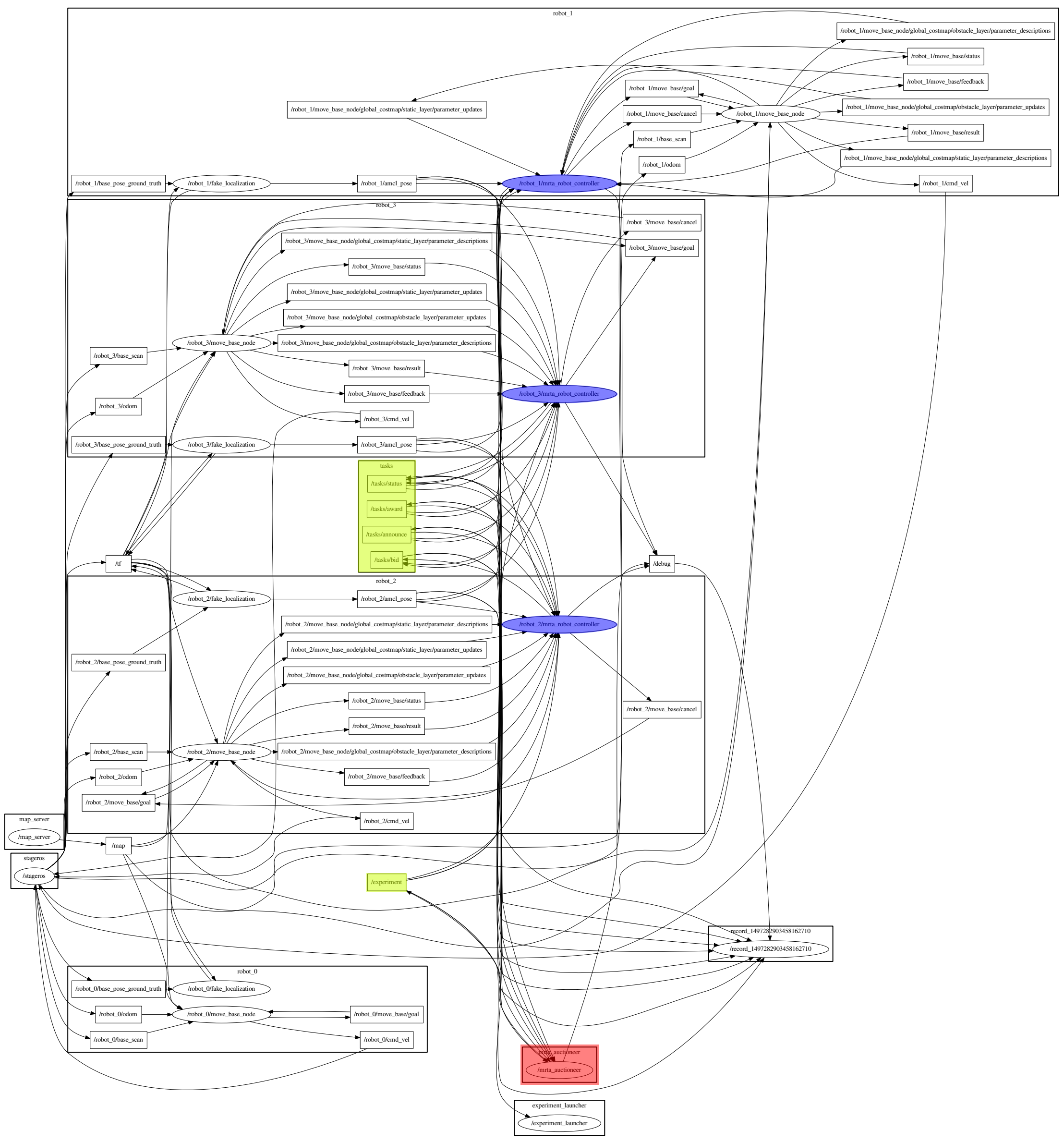

robot_1
/robot_1/move_base_node/global_costmap/obstacle_layer/parameter_descriptions
/robot_1/move_base/status
/robot_1/move_base/feedback
/robot_1/move_base/goal
/robot_1/move_base_node/global_costmap/static_layer/parameter_updates
/robot_1/move_base/cancel
/robot_1/move_base_node
/robot_1/move_base_node/global_costmap/obstacle_layer/parameter_updates
/robot_1/base_scan
/robot_1/move_base/result
/robot_1/odom
/robot_1/move_base_node/global_costmap/static_layer/parameter_descriptions
/robot_1/base_pose_ground_truth
/robot_1/fake_localization
/robot_1/amcl_pose
/robot_1/mrta_robot_controller
/robot_1/cmd_vel
robot_3
/robot_3/move_base/cancel
/robot_3/move_base_node/global_costmap/static_layer/parameter_descriptions
/robot_3/move_base/goal
/robot_3/move_base/status
/robot_3/move_base_node/global_costmap/static_layer/parameter_updates
/robot_3/move_base_node/global_costmap/obstacle_layer/parameter_updates
/robot_3/move_base_node
/robot_3/move_base_node/global_costmap/obstacle_layer/parameter_descriptions
/robot_3/base_scan
/robot_3/move_base/result
/robot_3/odom
/robot_3/move_base/feedback
/robot_3/mrta_robot_controller
/robot_3/cmd_vel
/robot_3/base_pose_ground_truth
/robot_3/fake_localization
/robot_3/amcl_pose
tasks
/tasks/status
/tasks/award
/tasks/announce
/tasks/bid
/tf
/debug
robot_2
/robot_2/fake_localization
/robot_2/amcl_pose
/robot_2/move_base_node/global_costmap/static_layer/parameter_descriptions
/robot_2/mrta_robot_controller
/robot_2/base_pose_ground_truth
/robot_2/move_base_node/global_costmap/static_layer/parameter_updates
/robot_2/move_base_node/global_costmap/obstacle_layer/parameter_updates
/robot_2/move_base/status
/robot_2/move_base/cancel
/robot_2/move_base/result
/robot_2/base_scan
/robot_2/move_base_node
/robot_2/move_base_node/global_costmap/obstacle_layer/parameter_descriptions
/robot_2/odom
/robot_2/move_base/feedback
/robot_2/move_base/goal
/robot_2/cmd_vel
map_server
/map_server
/map
stageros
/stageros
/experiment
record_1497282903458162710
/record_1497282903458162710
robot_0
/robot_0/base_pose_ground_truth
/robot_0/fake_localization
/robot_0/odom
/robot_0/move_base_node
/robot_0/move_base/goal
/robot_0/base_scan
/robot_0/cmd_vel
mrta_auctioneer
/mrta_auctioneer
experiment_launcher
/experiment_launcher

FIGURE B.1: A ROS computation graph of nodes and communication links in a MRTeAm experiment. Robot controllers are shaded blue. The auctioneer is shaded red. Message topics used to communicate about tasks during deliberation and task execution are shaded in yellow.

Listing 2 Task message definition

```
string task_id        # Unique task identifier
string[] depends      # Ids of other tasks that must be completed first
string type           # Currently just 'SENSOR_SWEEP'
uint8 num_robots      # Number of robots needed to complete this task
float32 duration      # Time (in seconds) required to 'execute' the task
float32 arrival_time  # Time (in seconds) at which the task 'arrives'
                      # after start of experiment
```

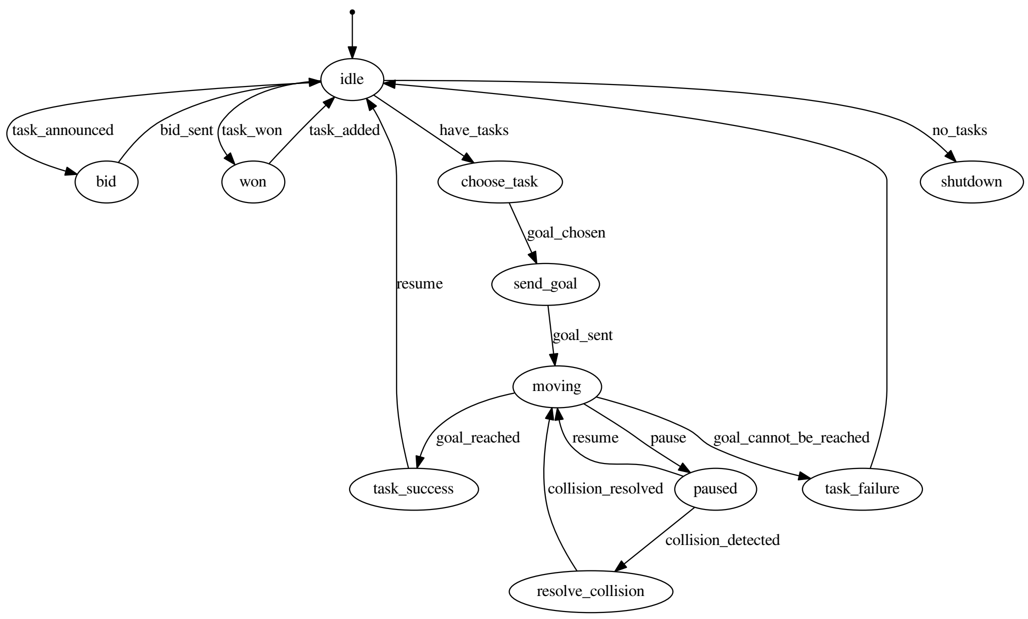


FIGURE B.2: State machine that controls a robot controller agent's behaviour.

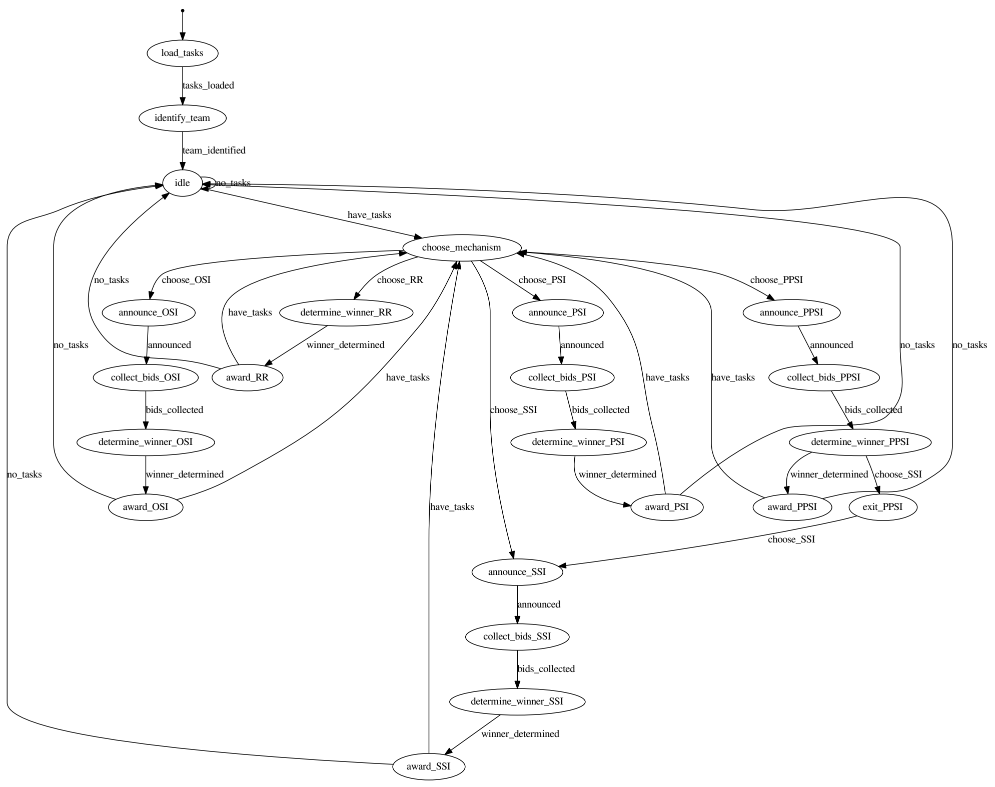


FIGURE B.3: State machine that controls the auctioneer agent's behaviour.

Listing 3 SensorSweepTask message definition

```
Task task                    # See Task.msg
geometry_msgs/Point location # Location in which to perform the sweep
```

Listing 4 AnnounceSensorSweep message definition

```
std_msgs/Header header  # Message header
string mechanism        # Robots need to know this in order to know how
                        # to bid
SensorSweepTask[] tasks # Tasks to announce. A list lets us announce
                        # bundles of tasks.
```

Listing 5 TaskBid message definition

```
std_msgs/Header header # Message header
string[] task_ids      # Unique task identifier(s)
string robot_id        # Unique id/name of the bidding robot
float64 bid            # Cost to complete the task (e.g., distance)
```

Listing 6 TaskAward message definition

```
std_msgs/Header header  # Message header
string robot_id         # Unique id/name of the robot being awarded
SensorSweepTask[] tasks # The tasks to be awarded
```

Listing 7 TaskStatus message definition

```
std_msgs/Header header       # Message header
string robot_id              # Unique robot identifier
string task_id               # Unique task identifier
uint8 status
uint8 MOVING             = 0 # Robot has begun to travel toward the task
uint8 PAUSE              = 1 # Paused (e.g., to avoid a collision)
uint8 RESUME             = 2 # Resuming from a PAUSE
uint8 ARRIVED            = 3 # Robot has arrived at the task site
uint8 BEGIN              = 4 # Robot has begun executing the task
uint8 SUCCESS            = 5 # Task successfully executed
uint8 FAILURE            = 6 # Task failed
uint8 ALL_TASKS_COMPLETE = 7 # All tasks have been completed
uint8 AGENDA_CLEARED     = 8 # This robot's agenda has been cleared
uint8 ABANDONED          = 9 # Gave up on moving toward this task
```

Appendix C

Classifier Details

This appendix provides some technical details about the classifier trained and used for the mechanism selection method discussed in Chapter 8.

The random forest classifier[1] is an ensemble classifier that comprises a set of decision trees. Each tree in the ensemble is constructed from a random sample drawn (with replacement) from a set of labelled training instances. Each internal node of a tree splits its child nodes by finding values of the training feature (from a *random sample* of training features) that group the sample of training instances into the most inequal class distributions (i.e., with the highest Gini coefficient [38]). When the random forest classifier is run, the class label selected by the greatest number of trees in the forest is returned.

Optimal parameters for the random forest classifier were found using an exhaustive grid search[2] and are given in table C.1.

[1]`http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`
[2]`http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`

Parameter	Description	Value
n_estimators	The number of trees in the forest	50
criterion	Function to measure quality of a split	Gini
max_features	Number of features to consider when looking for the best split	sqrt(*n_features*)
max_depth	Maximum depth of a tree	None
min_samples_split	Minimum number of samples required to split an internal node	3
min_samples_leaf	Minimum number of samples required to be at a leaf node	10
bootstrap	Whether bootstrap samples are used when building trees	False

Table C.1: Parameters of the random forest classifier evaluated in Chapter 8.

Bibliography

[1] Martin Andersson and Tuomas Sandholm, *Contract type sequencing for reallocative negotiation*, Proceedings 20th IEEE International Conference on Distributed Computing Systems, 2000, pp. 154–160.

[2] Kianoush Azarm and Günther Schmidth, *A decentralized approach for the conflict-free motion of multiple mobile robots*, Advanced Robotics **11** (1996), no. 4, 323–340.

[3] Tucker Balch, *The impact of diversity on performance in multi-robot foraging*, Proceedings of the third annual conference on Autonomous Agents, ACM, 1999, pp. 92–99.

[4] Tucker Balch and Ronald C. Arkin, *Behavior-based formation control for multirobot teams*, IEEE Transactions on robotics and automation **14** (1998), no. 6, 926–939.

[5] David P Barnes and J Gray, *Behaviour synthesis for co-operant mobile robot control*, Control 1991. Control'91., International Conference on, IET, 1991, pp. 1135–1140.

[6] Maxim A Batalin and Gaurav S Sukhatme, *Spreading out: A local approach to multi-robot coverage*, Distributed autonomous robotic systems **5** (2002), 373–382.

[7] Tolga Bektas, *The multiple traveling salesman problem: an overview of formulations and solution procedures*, Omega **34** (2006), no. 3, 209–219.

[8] Marc Berhault, He Huang, Pinar Keskinocak, Sven Koenig, Wedad Elmaghraby, Paul Griffin, and Anton Kleywegt, *Robot exploration with combinatorial auctions*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), vol. 2, IEEE, 2003, pp. 1957–1962.

[9] Sylvia C Botelho and Rachid Alami, *M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement*, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), vol. 2, IEEE, 1999, pp. 1234–1239.

[10] Craig Boutilier, Moisés Goldszmidt, and Bikash Sabata, *Sequential auctions for the allocation of resources with complementarities*, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), vol. 99, 1999, pp. 527–523.

[11] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo, *Swarm robotics: a review from the swarm engineering perspective*, Swarm Intelligence **7** (2013), no. 1, 1–41.

[12] Rodney A Brooks, *Artificial life and real robots*, 1st European Conference on Artificial Life, MIT Press, 1992, pp. 3–10.

[13] Barry L Brumitt and Anthony Stentz, *Grammps: A generalized mission planner for multiple mobile robots in unstructured environments*, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), vol. 2, IEEE, 1998, pp. 1564–1571.

[14] Grazyna Brzykcy, Jacek Martinek, Adam Meissner, and Piotr Skrzypczynski, *Multi-agent blackboard architecture for a mobile robot*, Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on, vol. 4, IEEE, 2001, pp. 2369–2374.

[15] Philippe Caloud, Wonyun Choi, J-C Latombe, Claude Le Pape, and Mark Yim, *Indoor automation with many mobile robots*, Intelligent Robots and Systems' 90.'Towards a New Frontier of Applications', Proceedings. (IROS 1990). IEEE International Workshop on, IEEE, 1990, pp. 67–72.

[16] Y Uny Cao, Alex S Fukunaga, and Andrew Kahng, *Cooperative mobile robotics: Antecedents and directions*, Autonomous robots **4** (1997), no. 1, 7–27.

[17] Jennifer Casper and Robin R Murphy, *Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center*, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **33** (2003), no. 3, 367–385.

[18] Fabrice Chantemargue and Béat Hirsbrunner, *A collective robotics application based on emergence and self-organization*, Proc. of Fifth International Conference for Young Computer Scientists (ICYCS99), 1999.

[19] Yann Chevaleyre, Paul E Dunne, Ulle Endriss, Jérôme Lang, Michel Lemaitre, Nicolas Maudet, Julian Padget, Steve Phelps, Juan A Rodriguez-Aguilar, and Paulo Sousa, *Issues in multiagent resource allocation*, Informatica **30** (2006), no. 1.

[20] Daniel D Corkill, *Blackboard systems*, AI expert **6** (1991), no. 9, 40–47.

[21] Paul J Densham and Gerard Rushton, *A more efficient heuristic for solving large p-median problems*, Papers in Regional Science **71** (1992), no. 3, 307–329.

[22] M Bernardine Dias, *Traderbots: A new paradigm for robust and efficient multirobot coordination in dynamic environments*, Ph.D. thesis, Carnegie Mellon University, 2004.

[23] M Bernardine Dias and Anthony Stentz, *Opportunistic optimization for market-based multirobot control*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002), vol. 3, 2002, pp. 2714–2720.

[24] Edsger W Dijkstra, *A note on two problems in connexion with graphs*, Numerische mathematik **1** (1959), no. 1, 269–271.

[25] Gregory Dudek, Michael RM Jenkin, Evangelos Milios, and David Wilkes, *A taxonomy for multi-agent robotics*, Autonomous Robots **3** (1996), no. 4, 375–397.

[26] Susan L Epstein, Anoop Aroor, Matthew Evanusa, Elizabeth I Sklar, and Simon Parsons, *Learning spatial models for navigation*, Proceedings of the 12th International Conference on Spatial Information Theory (New York, NY, USA), COSIT 2015, Springer-Verlag New York, Inc., 2015, pp. 403–425.

[27] Susan L Epstein, Anoop Aroor, Matthew Evanusa, Elizabeth I Sklar, and Simon Parsons, *Navigation with learned spatial affordances*, Proceedings of the 37th Annual Meeting of the Cognitive Science Society (Austin, Texas) (D. C. Noelle, R. Dale, A. S. Warlaumont, J. Yoshimi, T. Matlock, C. D. Jennings, and P. P. Maglio, eds.), Cognitive Science Society, 2015, July 22 - 25, 2015.

[28] Susan L Epstein, Anoop Aroor, Matthew Evanusa, Elizabeth I Sklar, and Simon Parsons, *Spatial abstraction for autonomous robot navigation*, Cognitive Processing **16** (2015), no. 1, 215–219.

[29] Alessandro Farinelli, Luca Iocchi, and Daniele Nardi, *Multirobot systems: a classification focused on coordination*, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **34** (2004), no. 5, 2015–2028.

[30] Paolo Fiorini and Zvi Shiller, *Motion planning in dynamic environments using velocity obstacles*, The International Journal of Robotics Research **17** (1998), no. 7, 760–772.

[31] Dieter Fox, *Kld-sampling: Adaptive particle filters*, Advances in Neural Information Processing Systems 14 (T. G. Dietterich, S. Becker, and Z. Ghahramani, eds.), MIT Press, 2002, pp. 713–720.

[32] Vanessa Frias-Martinez and Elizabeth I Sklar, *A framework for exploring role assignment in real-time, multiagent teams*, The second European Workshop on Multi-Agent Systems (EUMAS), 2004.

[33] Vanessa Frias-Martinez, Elizabeth I Sklar, and Simon Parsons, *Exploring auction mechanisms for role assignment in teams of autonomous robots*, Proceedings of the 8th RoboCup International Symposium, 2004.

[34] Brian P Gerkey and Maja J Matarić, *Murdoch: Publish/subscribe task allocation for heterogeneous agents*, Proceedings of the fourth international conference on Autonomous agents, ACM, 2000, pp. 203–204.

[35] Brian P Gerkey and Maja J Mataric, *Sold!: Auction methods for multirobot coordination*, Robotics and Automation, IEEE Transactions on **18** (2002), no. 5, 758–768.

[36] Brian P Gerkey and Maja J Matarić, *A formal analysis and taxonomy of task allocation in multi-robot systems*, The International Journal of Robotics Research **23** (2004), no. 9, 939–954.

[37] Brian P Gerkey, Richard T Vaughan, and Andrew Howard, *The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems*, Proceedings of the 11th International Conference on Advanced Robotics, 2003.

[38] Corrado Gini, *Variabilità e mutabilità*, Reprinted in Memorie di metodologica statistica (Ed. Pizetti E, Salvemini, T). Rome: Libreria Eredi Virgilio Veschi (1912).

[39] Dani Goldberg, Vincent Cicirello, M Bernardine Dias, Reid Simmons, Stephen Smith, Trey Smith, and Anthony Stentz, *A distributed layered architecture for mobile robot coordination: Application to space exploration*, In Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space, Citeseer, 2002.

[40] Matthew Gombolay, Ronald Wilcox, and Julie A Shah, *Fast scheduling of multi-robot teams with temporospatial constraints.*, Robotics: Science and Systems, 2013.

[41] Carla P Gomes and Bart Selman, *Algorithm portfolios*, Artificial Intelligence **126** (2001), no. 1-2, 43–62.

[42] Trond Grenager, Rob Powers, and Yoav Shoham, *Dispersion games: general definitions and some specific learning results*, Proceedings of the AAAI Conference on Artificial Intelligence, 2002, pp. 398–403.

[43] M. K. Habib, *Humanitarian Demining: Reality and the Challenge of Technology*, International Journal of Advanced Robotic Systems **4** (2007), no. 2.

[44] S Louis Hakimi, *Optimum locations of switching centers and the absolute centers and medians of a graph*, Operations research **12** (1964), no. 3, 450–459.

[45] Peter E Hart, Nils J Nilsson, and Bertram Raphael, *A formal basis for the heuristic determination of minimal cost paths*, IEEE Transactions on Systems Science and Cybernetics **4** (1968), no. 2.

[46] Bradford Heap, *Sequential single-cluster auctions for multi-robot task allocation*, Ph.D. thesis, The University of New South Wales, November 2013.

[47] Bradford Heap and Maurice Pagnucco, *Repeated sequential single-cluster auctions with dynamic tasks for multi-robot task allocation with pickup and delivery*, German Conference on Multiagent System Technologies, Springer, 2013, pp. 87–100.

[48] Daniel Hennes, Daniel Claes, Wim Meeussen, and Karl Tuyls, *Multi-robot collision avoidance with localization uncertainty*, Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems (AAMAS), International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 147–154.

[49] Bernardo A Huberman, Rajan M Lukose, and Tad Hogg, *An economics approach to hard computational problems*, Science **275** (1997), no. 5296, 51–54.

[50] Kao-Shing Hwang, Jin-Ling Lin, and Hui-Ling Huang, *Cooperative patrol planning of multi-robot systems by a competitive auction system*, ICCAS-SICE, 2009, IEEE, 2009, pp. 4359–4363.

[51] Markus Jager and Bernhard Nebel, *Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots*, roceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001), vol. 3, IEEE, 2001, pp. 1213–1219.

[52] Nick Jakobi, Phil Husbands, and Inman Harvey, *Noise and the reality gap: The use of simulation in evolutionary robotics*, Advances in Artificial Life, Springer, 1995, pp. 704–720.

[53] Nidhi Kalra, Robert Zlot, M Bernardine Dias, and Anthony Stentz, *Market-based multirobot coordination: A comprehensive survey and analysis*, Tech. report, DTIC Document, 2005.

[54] Oded Kariv and S Louis Hakimi, *An algorithmic approach to network location problems. ii: The p-medians*, SIAM Journal on Applied Mathematics **37** (1979), no. 3, 539–560.

[55] Hiroaki Kitano and Satoshi Tadokoro, *Robocup rescue: A grand challenge for multiagent and intelligent systems*, AI magazine **22** (2001), no. 1, 39.

[56] Hiroaki Kitano, Satoshi Tadokoro, Itsuki Noda, Hitoshi Matsubara, Tomoichi Takahashi, Atsuhi Shinjou, and Susumu Shimada, *Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research*, Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on, vol. 6, IEEE, 1999, pp. 739–743.

[57] S. Koenig, P. Keskinocak, and C. Tovey, *Progress on agent coordination with cooperative auctions*, Proceedings of the AAAI Conference on Artificial Intelligence, 2010.

[58] Sven Koenig, Craig A Tovey, Michail G Lagoudakis, David M Kempe, Pinar Keskinocak, Anton J Kleywegt, Adam Meyerson, and Sonal Jain, *The Power of Sequential Single-Item Auctions for Agent Coordination*, Proceedings of National Conference on Artificial Intelligence, 2006.

[59] Mary Koes, Illah Nourbakhsh, Katia Sycara, Mary Koes, Katia Sycara, Illah Nourbakhsh, Mary Koes, Illah Nourbakhsh, Katia Sycara, Sarvapali D Ramchurn, et al., *Heterogeneous multirobot coordination with spatial and temporal constraints*, Proceedings of the AAAI Conference on Artificial Intelligence, vol. 5, 2005, pp. 1292–1297.

[60] Kurt Konolige, Dieter Fox, Charlie Ortiz, Andrew Agno, Michael Eriksen, Benson Limketkai, Jonathan Ko, Benoit Morisset, Dirk Schulz, Benjamin Stewart, et al., *Centibots: Very large scale distributed robotic teams*, Experimental Robotics IX (2006), 131–140.

[61] Kurt Konolige, Charles Ortiz, Regis Vincent, Andrew Agno, Michael Eriksen, Benson Limketkai, Mark Lewis, Linda Briesemeister, Enrique Ruspini, Dieter Fox, et al., *Large scale robot teams*, Multi-Robot Systems: From Swarms to Intelligent Autonoma **2** (2003), 193–204.

[62] G Ayorkor Korsah, Anthony Stentz, and M Bernardine Dias, *A comprehensive taxonomy for multi-robot task allocation*, The International Journal of Robotics Research **32** (2013), no. 12, 1495–1512.

[63] John R Koza, *Evolution of subsumption using genetic programming*, Proceedings of the First European Conference on Artificial Life, 1992, pp. 110–119.

[64] Michael JB Krieger and Jean-Bernard Billeter, *The call of duty: Self-organised task allocation in a population of up to twelve mobile robots*, Robotics and Autonomous Systems **30** (2000), no. 1, 65–84.

[65] Michael JB Krieger, Jean-Bernard Billeter, and Laurent Keller, *Ant-like task allocation and recruitment in cooperative robots*, Nature **406** (2000), no. 6799, 992.

[66] Geert-Jan M Kruijff, Fiora Pirri, Mario Gianni, Panagiotis Papadakis, Matia Pizzoli, Arnab Sinha, Viatcheslav Tretyakov, Thorsten Linder, Emanuele Pianese, Salvatore Corrao, et al., *Rescue robots at earthquake-hit mirandola, italy: A field report*, Safety, security, and rescue robotics (SSRR), 2012 IEEE international symposium on, IEEE, 2012, pp. 1–8.

[67] C Ronald Kube and Eric Bonabeau, *Cooperative transport by ants and robots*, Robotics and autonomous systems **30** (2000), no. 1, 85–101.

[68] Harold W Kuhn, *The hungarian method for the assignment problem*, Naval research logistics quarterly **2** (1955), no. 1-2, 83–97.

[69] Michail G Lagoudakis, Marc Berhault, Sven Koenig, Pinar Keskinocak, and Anton J Kelywegt, *Simple auctions with performance guarantees for multi-robot task allocation*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004), 2004.

[70] Michail G Lagoudakis, Evangelos Markakis, David Kempe, Pinar Keskinocak, Sven Koenig, Anton Kleywegt, Craig Tovey, Adam Meyerson, and Sonal Jain, *Auction-based multi-robot routing*, Proceedings of Robotics: Science and Systems Conference, 2005.

[71] David Landén, Fredrik Heintz, and Patrick Doherty, *Complex task allocation in mixed-initiative delegation: a uav case study*, Principles and Practice of Multi-Agent Systems, Springer, 2012, pp. 288–303.

[72] Gilbert Laporte, *The vehicle routing problem: An overview of exact and approximate algorithms*, European journal of operational research **59** (1992), no. 3, 345–358.

[73] Gilbert Laporte, Stefan Nickel, and Francisco Saldanha da Gama, *Location science*, Springer, February 2015.

[74] Guillaume Lemaître, Fernando Nogueira, and Christos K Aridas, *Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning*, Journal of Machine Learning Research **18** (2017), no. 17, 1–5.

[75] Kevin Leyton-Brown, Eugene Nudelman, Galen Andrew, Jim McFadden, and Yoav Shoham, *A portfolio approach to algorithm selection*, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), vol. 1543, 2003, p. 2003.

[76] Lantao Liu and Dylan A. Shell, *Large-scale multi-robot task allocation via dynamic partitioning and distribution*, Autonomous Robots **33** (2012), no. 3, 291–307.

[77] Beatriz López, Silvia Suárez, and JL De La Rosa, *Task allocation in rescue operations using combinatorial auctions*, Artificial Intelligence Research and Development **100** (2003), 233–243.

[78] Lingzhi Luo, Nilanjan Chakraborty, and Katia Sycara, *Multi-robot assignment algorithm for tasks with set precedence constraints*, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2011, pp. 2526–2533.

[79] Maja J Mataríc, *A distributed model for mobile robot environment-learning and navigation*, Tech. report, MIT Artificial Intelligence Laboratory, 1990.

[80] Maja J Matarić and Dave Cliff, *Challenges in evolving controllers for physical robots*, Robotics and autonomous systems **19** (1996), no. 1, 67–83.

[81] Maja J Matarić, Gaurav S Sukhatme, and Esben H Østergaard, *Multi-robot task allocation in uncertain environments*, Autonomous Robots **14** (2003), no. 2-3, 255–263.

[82] Orazio Miglino, Henrik Hautop Lund, and Stefano Nolfi, *Evolving mobile robots in simulated and real environments*, Artificial life **2** (1995), no. 4, 417–434.

[83] Robin R Murphy, Jeffery Kravitz, Samuel L Stover, and Rahmat Shoureshi, *Mobile robots in mine rescue and recovery*, IEEE Robotics & Automation Magazine **16** (2009), no. 2.

[84] Keiji Nagatani, Seiga Kiribayashi, Yoshito Okada, Kazuki Otake, Kazuya Yoshida, Satoshi Tadokoro, Takeshi Nishimura, Tomoaki Yoshida, Eiji Koyanagi, Mineo Fukushima, et al., *Emergency response to the nuclear accident at the fukushima daiichi nuclear power plants using mobile rescue robots*, Journal of Field Robotics **30** (2013), no. 1, 44–63.

[85] Ranjit Nair, Takayuki Ito, Milind Tambe, and Stacy Marsella, *Task allocation in the robocup rescue simulation domain: A short note*, Robot Soccer World Cup, Springer, 2001, pp. 751–754.

[86] Maitreyi Nanjanath and Maria Gini, *Repeated auctions for robust task execution by a robot team*, Robotics and Autonomous Systems **58** (2010), no. 7, 900–909.

[87] Fabrice R Noreils, *Toward a robot architecture integrating cooperation between mobile robots: Application to indoor environment*, The International Journal of Robotics Research **12** (1993), no. 1, 79–98.

[88] Ernesto Nunes and Maria Gini, *Multi-robot auctions for allocation of tasks with temporal constraints*, Proceedings of the AAAI Conference on Artificial Intelligence, 2015.

[89] A Tuna Özgelen, *Real-time supervision for human-robot teams in complex task domains*, Ph.D. thesis, City University of New York, 2015.

[90] A Tuna Özgelen, Eric Schneider, Elizabeth I Sklar, Michael Costantino, Susan L Epstein, and Simon Parsons, *A first step toward testing multiagent coordination mechanisms on multi-robot teams*, Proceedings of the Workshop on Autonomous Robots and Multirobot Systems (ARMS) at Autonomous Agents and MultiAgent Systems (AAMAS), 2013.

[91] Lynne E Parker, *Alliance: An architecture for fault tolerant multirobot cooperation*, IEEE transactions on robotics and automation **14** (1998), no. 2, 220–240.

[92] Lynne E Parker, *Distributed intelligence: Overview of the field and its application in multi-robot systems*, Journal of Physical Agents **2** (2008), no. 1, 5–14.

[93] Simon Parsons, Juan A Rodriguez-Aguilar, and Mark Klein, *Auctions and bidding: A guide for computer scientists*, ACM Computing Surveys (CSUR) **43** (2011), no. 2, 10.

[94] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research **12** (2011), 2825–2830.

[95] Steve Phelps, Peter McBurney, and Simon Parsons, *Evolutionary mechanism design: a review*, Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems (AAMAS) **21** (2010), no. 2, 237–264.

[96] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng, *Ros: an open-source robot operating system*, ICRA Workshop on Open Source Software, 2009.

[97] Josh Reese, *Solution methods for the p-median problem: An annotated bibliography*, Networks **48** (2006), no. 3, 125–142.

[98] John R Rice, *The algorithm selection problem*, Advances in computers **15** (1976), 65–118.

[99] Michael Rubenstein, Christian Ahler, and Radhika Nagpal, *Kilobot: A low cost scalable robot system for collective behaviors*, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2012, pp. 3293–3298.

[100] Stuart Russell and Peter Norvig, *Artificial intelligence: A modern approach*, 3rd ed., Prentice Hall Press, Upper Saddle River, NJ, USA, 2009.

[101] Paul E Rybski, Sascha A Stoeter, Maria Gini, Dean F Hougen, and Nikolaos P Papanikolopoulos, *Performance of a distributed robotic system using shared communications channels*, IEEE transactions on Robotics and Automation **18** (2002), no. 5, 713–727.

[102] Tuomas Sandholm, *Contract types for satisficing task allocation: I theoretical results*, Proceedings of the AAAI Spring Symposium: Satisficing Models, 1998, pp. 68–75.

[103] Tuomas Sandholm, *Algorithm for optimal winner determination in combinatorial auctions*, Artificial Intelligence **135** (2002), 1–54.

[104] Sanem Sariel and Tucker Balch, *Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments*, Proceedings of the AAAI-05 Workshop on Integrating Planning into Scheduling, AAAI Palo Alto, CA, 2005, pp. 27–33.

[105] Andrey V Savkin, *The problem of coordination and consensus achievement in groups of autonomous mobile robots with limited communication*, Nonlinear Analysis: Theory, Methods & Applications **65** (2006), no. 5, 1094–1102.

[106] Eric Schneider, Ofear Balas, A Tuna Ozgelen, Elizabeth I Sklar, and Simon Parsons, *An empirical evaluation of auction-based task allocation in multi-robot teams*, Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 1443–1444.

[107] Eric Schneider, Elizabeth I. Sklar, M. Q. Azhar, Simon Parsons, and Karl Tuyls, *Towards a methodology for describing the relationship between simulation and reality*, Proceedings of the European Conference on Artificial Life (ECAL), MIT Press, 2015, pp. 562–569.

[108] Eric Schneider, Elizabeth I Sklar, and Simon Parsons, *Evaluating multi-robot teamwork in parameterised environments*, Towards Autonomous Robotic Systems: 17th Annual Conference, TAROS 2016 (Lyuba Alboul, Dana Damian, and Jonathan M. Aitken, eds.), Springer International Publishing, 2016, pp. 301–313.

[109] Eric Schneider, Elizabeth I Sklar, and Simon Parsons, *Mechanism selection for multi-robot task allocation*, Towards Autonomous Robotic Systems: 18th Annual Conference, TAROS 2017 (Yang Gao, Saber Fallah, Yaochu Jin, and Constantina Lekakou, eds.), Springer International Publishing, 2017, pp. 421–435.

[110] Eric Schneider, Elizabeth I Sklar, Simon Parsons, and A Tuna Özgelen, *Auction-based task allocation for multi-robot teams in dynamic environments*, Towards Autonomous Robotic Systems: 16th Annual Conference, TAROS 2015 (Clare Dixon and Karl Tuyls, eds.), Springer International Publishing, 2015, pp. 246–257.

[111] Adrian Schoenig and Maurice Pagnucco, *Evaluating sequential single-item auctions for dynamic task allocation*, AI 2010: Advances in Artificial Intelligence, Springer, 2011.

[112] Yoav Shoham, *A survey of auction types*, Lecture notes: Stanford University CS206Technical Foundations of Electronic Commerce (2000).

[113] Reid Simmons and David Apfelbaum, *A task description language for robot control*, Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on, vol. 3, IEEE, 1998, pp. 1931–1937.

[114] Reid Simmons, David Apfelbaum, Wolfram Burgard, Dieter Fox, Mark Moor, Sbastian Thrun, and Hakan Younes, *Coordination for multi-robot exploration and mapping*, Proceedings of the AAAI Conference on Artificial Intelligence, 2000.

[115] Elizabeth I Sklar, A Tuna Özgelen, J Pablo Muñoz, Joel Gonzalez, Mark Manashirov, Susan L Epstein, and Simon Parsons, *Designing the HRTeam Framework: Lessons Learned from a Rough-and-Ready Human/Multi-Robot Team*, Proceedings of the Workshop on Autonomous Robots and Multirobot Systems (ARMS) at Autonomous Agents and MultiAgent Systems (AAMAS) (Taipei, Taiwan), May 2011.

[116] Reid G Smith, *The contract net protocol: High-level communication and control in a distributed problem solver*, IEEE Transactions on computers **C-29** (1980), 12.

[117] Peng Song and Vijay Kumar, *A potential field based approach to multi-robot manipulation*, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), vol. 2, IEEE, 2002, pp. 1217–1222.

[118] Anthony Stentz and M Bernardine Dias, *A free market architecture for coordinating multiple robots*, Tech. report, DTIC Document, 1999.

[119] Peter Stone and Manuela Veloso, *Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork*, Artificial Intelligence **110** (1999), no. 2, 241–273.

[120] Michael B Teitz and Polly Bart, *Heuristic methods for estimating the generalized vertex median of a weighted graph*, Operations research **16** (1968), no. 5, 955–961.

[121] George Thomas and Andrew B Williams, *Sequential auctions for heterogeneous task allocation in multiagent routing domains*, IEEE International Conference on Systems, Man and Cybernetics, IEEE, 2009.

[122] C. Tovey, M. G. Lagoudakis, S. Jain, and S. Koenig, *Generation of Bidding Rules for Auction-Based Robot Coordination*, Proceedings of the 3rd International Multi-Robot Systems Workshop, March 2005.

[123] Nikolaos Tsiogkas, Georgios Papadimitriou, Zeyn Saigol, and David Lane, *Efficient multi-auv cooperation using semantic knowledge representation for underwater archaeology missions*, Oceans-St. John's, 2014, IEEE, 2014, pp. 1–6.

[124] Nikolaos Tsiogkas, Zeyn Saigol, and David Lane, *Distributed multi-auv cooperation methods for underwater archaeology*, OCEANS 2015-Genova, IEEE, 2015, pp. 1–5.

[125] Douglas Vail and Manuela Veloso, *Multi-robot dynamic role assignment and coordination through shared potential fields*, Multi-robot systems (2003), 87–98.

[126] Jur Van den Berg, Ming Lin, and Dinesh Manocha, *Reciprocal velocity obstacles for real-time multi-agent navigation*, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2008, pp. 1928–1935.

[127] Manuela Veloso and Peter Stone, *Individual and collaborative behaviors in a team of homogeneous robotic soccer agents*, Multi Agent Systems, 1998. Proceedings. International Conference on, IEEE, 1998, pp. 309–316.

[128] Steve Vinoski, *Advanced message queuing protocol*, IEEE Internet Computing **10** (2006), no. 6, 87–89.

[129] Glenn Wagner and Howie Choset, *M*: A complete multirobot path planning algorithm with performance bounds*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011), IEEE, 2011, pp. 3260–3267.

[130] Jing Wang, *On sign-board based inter-robot communication in distributed robotic systems*, Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on, IEEE, 1994, pp. 1045–1050.

[131] Gerhard Weiss, *Multiagent systems: a modern approach to distributed artificial intelligence*, MIT press, 1999.

[132] Michael P Wellman and Peter R Wurman, *Market-Aware Agents for a Multiagent World*, Robotics and Autonomous Systems **24** (1998), 115–125.

[133] Ningchuan Xiao, *Gis algorithms*, SAGE Publications, 2015.

[134] Lin Xu, Holger Hoos, and Kevin Leyton-Brown, *Hydra: Automatically configuring algorithms for portfolio-based selection.*, Proceedings of the AAAI Conference on Artificial Intelligence, vol. 10, 2010, pp. 210–216.

[135] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown, *Satzilla: portfolio-based algorithm selection for sat*, Journal of artificial intelligence research **32** (2008), 565–606.

[136] Zhi Yan, Luc Fabresse, Jannik Laval, and Noury Bouraqadi, *Team size optimization for multi-robot exploration*, International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Springer, 2014, pp. 438–449.

[137] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif, *A survey and analysis of multi-robot coordination*, International Journal of Advanced Robotic Systems **10** (2013), no. 12, 399.

[138] Wei Ye, Richard T Vaughan, Gaurav S Sukhatme, John Heidemann, Deborah Estrin, and Maja J Mataric, *Evaluating control strategies for wireless-networked robots using an integrated robot and network simulation*, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), vol. 3, IEEE, 2001, pp. 2941–2947.

[139] Sven Koenig Xiaoming Zheng, Craig Tovey, Richard Borie, Philip Kilby, Vangelis Markakis, and Pinar Keskinocak, *Agent coordination with regret clearing*, Proceedings of the AAAI Conference on Artificial Intelligence, AAAI Press, 2008, p. 101.

[140] Xiaoming Zheng, Sven Koenig, and Craig Tovey, *Improving sequential single-item auctions*, Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, IEEE, 2006, pp. 2238–2244.

[141] Tsvetan Zhivkov, Eric Schneider, and Elizabeth I Sklar, *Measuring the effects of communication quality on multi-robot team performance*, Towards Autonomous Robotic Systems: 18th Annual Conference, TAROS 2017 (Yang Gao, Saber Fallah, Yaochu Jin, and Constantina Lekakou, eds.), Springer International Publishing, 2017, pp. 408–420.

[142] Robert Zlot and Anthony Stentz, *Market-based multirobot coordination for complex tasks*, The International Journal of Robotics Research **25** (2006), no. 1, 73–101.