Task Allocation and Collaborative Localisation in Multi-Robot Systems

Nick Sullivan

School of Mechanical Engineering

The University of Adelaide

This thesis is submitted for the degree of

Doctor of Philosophy

July 2019

Abstract

To utilise multiple robots, it is fundamental to know what they should do, called task allocation, and to know where the robots are, called localisation. The order that tasks are completed in is often important, and makes task allocation difficult to solve (40 tasks have 10^{47} different ways of completing them). Algorithms in literature range from fast methods that provide reasonable allocations, to slower methods that can provide optimal allocations. These algorithms work well for systems with identical robots, but do not utilise robot differences for superior allocations when robots are non-identical. They also can not be applied to robots that can use different tools, where they must consider which tools to use for each task.

Robot localisation is performed using sensors which are often assumed to always be available. This is not the case in GPS-denied environments such as tunnels, or on long-range missions where replacement sensors are not readily available. A promising method to overcome this is collaborative localisation, where robots observe one another to improve their location estimates. There has been little research on what robot properties make collaborative localisation most effective, or how to tune systems to make it as accurate as possible.

Most task allocation algorithms do not consider localisation as part of the allocation process. If task allocation algorithms limited inter-robot distance, collaborative localisation can be performed during task completion. Such an algorithm could equally

be used to ensure robots are within communication distance, and to quickly detect when a robot fails. While some algorithms for this exist in literature, they provide a weak guarantee of inter-robot distance, which is undesirable when applied to real robots.

The aim of this thesis is to improve upon task allocation algorithms by increasing task allocation speed and efficiency, and supporting robot tool changes. Collaborative localisation parameters are analysed, and a task allocation algorithm that enables collaborative localisation on real robots is developed.

This thesis includes a compendium of journal articles written by the author. The four articles forming the main body of the thesis discuss the multi-robot task allocation and localisation research during the author's candidature. Two appendices are included, representing conference articles written by the author that directly relate to the thesis.

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint award of this degree. I acknowledge that copyright of published works contained within this thesis resides with the copyright holder(s) of those works. I give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship, and from the Commonwealth of Australia (represented by the Defence Science and Technology Group) through a Defence Science Partnerships agreement.

Nick Sullivan, 21 Mar 2019

Acknowledgements

While I have spent a considerable number of hours on this thesis, it could not have been completed without the support of many people in my life.

An honourable mention to Ezzi Vision, for sending spam emails that continued to thwart the spam filter despite my best efforts. Your persistence was unwanted but admirable.

Thanks to my supervisors Steven and Ben, for knowing when to let me burrow into my work, and when to steer me back on track.

Thanks to DST Group for helping make my research relevant to real-world problems (and for some pocket money).

Thanks to my co-workers for the laughs and Friday afternoon drinks.

Thanks to my loved ones for their encouragement and support.

Nick

List of Publications Arising From This Thesis

Journals

1. **Sullivan, N**, Grainger, S, Cazzolato, B 2018, 'Sequential single-item auction improvements for heterogeneous multi-robot routing', *Robotics and Autonomous Systems*, vol 115, pp 130-142. Accepted 25 Feb 2019.

2. **Sullivan, N**, Grainger, S, Cazzolato, B 2018, 'Algorithms for multi-robot routing with adaptive heterogeneity', *Journal of Heuristics*, under review.

3. **Sullivan, N**, Grainger, S, Cazzolato, B 2018, 'Analysis of cooperative localisation performance under varying sensor qualities and communication rates', *Robotics and Autonomous Systems*, vol 110, pgs 73-84. Accepted 29 September 2018.

4. **Sullivan, N**, Grainger, S, Cazzolato, B 2018, 'Formation-based multi-robot routing with distance constraints', *IEEE Transactions on Automation Science and Engineering*, under review.

Conferences

1. **Sullivan, N**, Grainger, S, Cazzolato, B 2017, 'Robust heterogeneous multi-robot routing for low-intelligence agents', *Australasian Conference on Robotics and Automation* (ACRA 2017), Australia.

2. **Sullivan, N**, Grainger, S, Cazzolato, B 2018, 'A dual genetic algorithm for multi-robot routing with network connectivity and fuel efficiency', *International Conference on Robotics and Computer Vision* (ICARCV 2018), Singapore.

3. **Sullivan N**, Pearce, G, Grainger, S, Cazzolato, B 2018, 'An outdoor multi-vehicle platform for cooperative localisation research', *Australasian Conference on Robotics and Automation* (ACRA 2018), New Zealand.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

C-SLAM Collaborative simultaneous localisation and mapping

CL Collaborative/cooperative localisation

EKF Extended Kalman filter

GA Genetic algorithm

H-mTSP Heterogeneous multiple traveling salesman problem

ILP Integer linear program

MRS Multi-Robot System

MRTA Multi-robot task allocation

mTSP Multiple travelling salesman problem

NSGA-II Non-dominated sorting genetic algorithm II

SLAM Simultaneous localisation and mapping

TSP Travelling salesman problem

VRP Vehicle routing problem

Chapter 1

Background

1.1 Multi-Robot Systems

Many industries, such as manufacturing, are considered automated because they use robots to perform the same tasks repeatedly. However, *interaction* between robots is typically hand-crafted by humans. While this has proven effective for mass production, it is difficult to adapt these systems to meet changing demands. Manufacturing a new product requires a new interaction design from engineers, and many weeks of re-programming the robots to obey the new design. This is a slow process, which limits product diversity and customisation. It is also an inefficient process: humans often rely on intuition, which frequently results in suboptimal decisions. In response to these shortcomings, there has been an increased desire to upgrade these systems to ones where humans declare a high-level goal, and the robots calculate how they should operate in order to achieve this goal. This current automation direction, deemed Industry 4.0, aims to improve the interoperability, transparency, accessibility, and decentralisation of automated industries [1]. This would shift automated systems to use a number of robots that act independently while adhering to higher level instructions.

In essence, automation is moving from single-robot systems to multi-robot systems (MRSs).

First researched in the late 1980s, multi-robot system research quickly displayed relevance to artificial intelligence, game theory, economics, theoretical biology, distributed computing, and artificial life [2]. MRSs offer advantages such as increased spatial distribution, improved robustness through redundancy and data fusion, and improved versatility and scalability [3].

MRSs have had impact on industries including warehouse automation [4], agriculture [5], disaster response [6], search and rescue [7], environment monitoring [8], healthcare assistance [9], mining [10], and assembly [11]. Another industry of particular importance is transportation, where self-driving cars are predicted to revolutionise how people and products move around [12]. Efficient operation of MRSs in these fields will have significant cost savings for the employer, cost reductions for customers, and reduced environmental impact from energy usage.

While MRSs have greater capabilities than their singular counterparts, they are more complex to design and operate. They require decisions regarding communication (who should robots communicate with, and what should they say?), negotiation (how should robots reason if they have conflicting goals?), and control (how should robots move to avoid collisions?).

This thesis explores the two fundamental questions of task allocation (*given a number of tasks to be completed, which robots should complete them, and in what order?*), and localisation (*where are the robots located?*).

Table 1.1: Multi-robot task allocation categorisation [13].

Category	Option 1	Option 2
Robot Type	Single-Task (ST), robots complete one task at a time.	Multi-Task (MT), robots complete multiple tasks at the same time.
Task Type	Single-Robot (SR), each task is completed by one robot.	Multi-Robot (MR), a task requires multiple robots to be completed.
Allocation Type	Instantaneous Assignment (IA), robots are given tasks to complete immediately.	Time-Extended Assignment (TA), robots are given a schedule of tasks to complete.

1.2 Task Allocation

In any robotic system, tasks must be allocated to robots for completion. Given multiple tasks, a single robot system must determine the order of completion. A multi-robot system, however, must also determine which robot to use for each task. These two problems are linked, and should therefore not be solved independently [13] i.e., tasks should be allocated to robots that can complete them efficiently, but how efficiently a robot completes a task depends on the other tasks that it has been allocated.

The techniques used to solve the the multi-robot task allocation problem (MRTA) are largely dependent on the type of problem. MRTA has been categorised to reflect the various types of problems [13]. These categories are shown in Table 1.1. For example, The ST-SR-TA (single-task, single-robot, time-extended) category would apply to automated transportation, where each vehicle completes one delivery at a time, and the order of the deliveries impacts total delivery time. The ST-MR-TA (single-task, multi-robot, time-extended) category would apply to robotic manufacturing. Each robot builds one part at a time, and each product requires multiple robots working on it for it to be made. The ST-SR-IA (single-task, single-robot, instantaneous assignment) would apply to a simple operating system scheduler. The order of tasks is not important, it only needs to know the next task to process.

This thesis is focussed on ST-SR-TA problems. In particular, problems where tasks have spatial locations. To complete a task, robots must move to the task location and spend a specified amount of time working to complete the task. We do not define the nature of the work, but assume that it can be completed by a single robot. This type of task allocation problem applies to agriculture, search and rescue, transportation, cleaning, and many others [14].

This class of task allocation problem is also known by other names, such as multi-robot routing, the Vehicle Routing Problem (VRP), and the Multiple Travelling Salesman Problem (mTSP), which is an extension on the well-known Travelling Salesman Problem (TSP). The original TSP is as follows: *a salesman wishes to travel to every city in their country before returning home. In what order should they visit the cities in order to minimise travel time?* This translates into a robotics problem: *a robot wishes to complete every task in their area before returning to recharge. In what order should they complete the tasks in order to minimise travel time?* For a small number of tasks, this is a simple problem. One could calculate each possible allocation and keep track of the allocation that produces the minimum travel time. As we increase the number of tasks, however, this strategy becomes too costly to calculate. With N tasks, the number of possible allocations is N factorial ($N!$). For example, a TSP with 40 tasks has 8.2×10^{47} possible allocations. This is a textbook *NP-hard* problem, where the problem size scales larger than polynomially. With such problems, we care about both allocation quality and required processing time.

The mTSP has multiple salesmen (robots), and each city (task) must be visited (completed) at least once. In addition to the added complexity, the mTSP requires several extra decisions to be made. For example, the robots may start at the same or different locations. We may be interested in the most energy efficient solution, the fastest solution, or a balance between them. The robots may be identical (homogeneous), or may be suited for different types of tasks (heterogeneous).

It is important to note that there are many similar problems to the mTSP. While they are not addressed in this thesis, it is useful to briefly discuss them to frame the scope of this thesis. If the order of tasks is unimportant, it becomes an assignment problem. In the assignment problem, each robot only needs to be given a single task to solve. This can be solved exactly in polynomial-time [15], meaning heuristics and metaheuristics are usually not needed.

If tasks are unknown in advance, it is possible to apply mTSP algorithms by re-calculating each time a new task is discovered. However, this can be impractical in certain dynamic systems, and algorithms specifically designed for dynamic systems should be used for those cases [16].

If robots only discover tasks if they are nearby, it becomes a searching problem. Balance must be found between searching for unknown tasks and solving known tasks [17]. In some cases, research has also dealt with communication limitations, where it is desired for robots to pass information regularly so that recently explored areas are not re-explored, and that a home base is updated with the current situation [18].

It is possible for task completion costs to be stochastic, where there are estimated probabilities for the amount of time a robot will take to complete a certain task. Stochastic algorithms can then find solutions that perform well and with a high level of confidence [19]. This is particularly relevant if certain tasks must be completed before other tasks are able to be started, where one task taking longer than normal may force robots to wait.

A significant portion of this thesis involves the development and analysis of new algorithms for solving the mTSP. Existing algorithms are reviewed in Chapter 2.

1.3 Localisation

It is important to know where robots are within their environment, known as localisation. This is a fundamental requirement for navigation, collision avoidance, and task completion. We often require position and orientation, known as *pose*, as well as linear and angular velocity, known as *twist*. This information, and potentially others, make up a robots *state*.

Single robots localise using two classes of on-board sensors. The internal state of robots are measured by *interoceptive* sensors, such as gyroscopes (measuring angular velocity), accelerometers (measuring acceleration), and wheel encoders (measuring wheel rotation). Interoceptive sensors reliably provide data, but have errors that accumulate over time. *Exteroceptive* sensors interact with the environment, such as GPS, cameras, magnetometers and pulsed lasers (LIDARs). Exteroceptive sensors typically have far less error accumulation, but they are sensitive to environmental conditions. For example, localisation using GPS requires satellite signals, localisation from LIDARs require static terrain or recognisable features, and cameras require certain lighting conditions.

With so many sources of data, it is necessary to fuse them to form a single best-estimate of the robot's state. One of the most common data fusion techniques is the Kalman Filter, which uses a combination of sensor inputs and system dynamics to produce pose and twist estimates. A common extension to this filter is the Extended Kalman Filter (EKF), which can be used for non-linear systems such as robot localisation [20].

Robots in multi-robot systems can make use of singular robot localisation techniques, but are also presented with an additional opportunity. The robots are able to help one another localise. This is known as Collaborative (or Cooperative) Localisation

(CL). There are two major research areas for CL, one is known as Collaborative Simultaneous Localisation and Mapping (C-SLAM), where robots independently produce maps of the environment and then share and combine these maps. C-SLAM was a key contributor for the winning team of the MAGIC 2010 competition [21], where robots had to autonomously survey and map a 500m x 500m dynamic urban environment. Communication was not always available, so individual mapping and map fusion was necessary to continue surveillance during communication down-times. C-SLAM has also been used for tasks such as mapping a large area with aerial vehicles [22], localising underwater vehicles to reduce the need for surfacing [23], and to identify and track dynamic targets [24]. C-SLAM can be powerful, but it has requirements that make it unsuitable in certain systems. Firstly, each robot must have SLAM capabilities. This can inflate the cost of multi-robot systems, as environment mapping often makes use of high quality sensors such as 3D LIDARs. Each robot must also be capable of processing data quickly, either through on-board processing or communication, and is therefore not suitable for systems with inexpensive processors or unreliable communication. Secondly, SLAM performance is dependent on the type and number of landmarks in the environment [25]. SLAM does not operate well in open areas where there are few recognisable features.

The other major area for CL research involves measuring and communicating inter-robot observations. This differs from C-SLAM in that no map sharing occurs. Robots observe one another, estimate each other's position, and communicate their estimates to the observed robots. There are no requirements for how robots localise and perform inter-robot measurements, allowing individual robots to have different sensors, processing capabilities, and internal representations of the environment. There is also less dependence on the environment, and can successfully operate provided robots are able to detect one another. This method of CL is what is referred to in this thesis.

1.4 Research Aims

The intent of this thesis is to develop and analyse new state-of-the-art multi-robot task allocation algorithms, specifically for heterogeneous robots (Chapter 3), and robots that use tools (Chapter 4). It also analyses the conditions where collaborative localisation is beneficial (Chapter 5), and uses that information as part of a new task allocation algorithm that considers collaborative localisation as part of the allocation process (Chapter 6). Specific objectives are listed in Chapter 2, where research gaps are identified.

Superior task allocation and localisation will improve the efficiency, effectiveness, and reliability of multi-robot systems. It has significant relevance to a variety of industries, particularly transportation, warehouse automation, and defence.

1.5 References

[1] Mario Hermann, Tobias Pentek, and Boris Otto. Design principles for industrie 4.0 scenarios. In *System Sciences (HICSS), 2016 49th Hawaii International Conference on*, pages 3928–3937. IEEE, 2016.

[2] Y Uny Cao, Alex S Fukunaga, and Andrew Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–27, 1997.

[3] Lynne E Parker. Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents*, 2(1):5–14, 2008.

[4] Roelof Hamberg and Jacques Verriet. *Automation in warehouse development*. Springer, 2012.

[5] Anthony King. The future of agriculture. *Nature*, 544(7651):S21–S23, 2017.

[6] Mathew DeDonato, Velin Dimitrov, Ruixiang Du, Ryan Giovacchini, Kevin Knoedler, Xianchao Long, Felipe Polido, Michael A Gennert, Taskin Padir, and Siyuan Feng. Human-in-the-loop control of a humanoid robot for disaster response: A report from the DARPA robotics challenge trials. *Journal of Field Robotics*, 32(2):275–292, 2015.

[7] Geert-Jan M Kruijff, M Janícek, Shanker Keshavdas, Benoit Larochelle, Hendrik Zender, Nanja JJM Smets, Tina Mioch, Mark A Neerincx, Jurriaan Van Diggelen, and Francis Colas. Experience in system design for human-robot teaming in urban search and rescue. In *Field and Service Robotics*, pages 111–125. Springer, 2014.

[8] Anand Nayyar, Vikram Puri, Nhu Gia Nguyen, and Dac Nhuong Le. *Smart Surveillance Robot for Real-Time Monitoring and Control System in Environment and Industrial Applications*, pages 229–243. Springer, 2018.

[9] Yin Zhang, Meikang Qiu, Chun-Wei Tsai, Mohammad Mehedi Hassan, and Atif Alamri. Health-cps: Healthcare cyber-physical system assisted by cloud and big data. *IEEE Systems Journal*, 11(1):88–95, 2017.

[10] Joshua A Marshall, Adrian Bonchis, Eduardo Nebot, and Steven Scheding. *Robotics in mining*, pages 1549–1576. Springer, 2016.

[11] Ross A Knepper, Todd Layton, John Romanishin, and Daniela Rus. Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 855–862. IEEE, 2013.

[12] Todd Litman. *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute, 2017.

[13] G Ayorkor Korsah, Anthony Stentz, and M Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512, 2013.

[14] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[15] Lantao Liu and Dylan A Shell. Assessing optimal assignment under uncertainty: An interval-based algorithm. *The International Journal of Robotics Research*, 30(7):936–953, 2011.

[16] Francesco Bullo, Emilio Frazzoli, Marco Pavone, Ketan Savla, and Stephen L Smith. Dynamic vehicle routing for robotic systems. *Proceedings of the IEEE*, 99(9):1482–1504, 2011.

[17] Torsten Andre and Christian Bettstetter. Collaboration in multi-robot exploration: To meet or not to meet? *Journal of Intelligent & Robotic Systems*, 82(2):325, 2016.

[18] Jacopo Banfi, Alberto Quattrini Li, Nicola Basilico, Ioannis Rekleitis, and Francesco Amigoni. Asynchronous multirobot exploration under recurrent connectivity constraints. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 5491–5498. IEEE, 2016.

[19] Roberto Tadei, Guido Perboli, and Francesca Perfetti. The multi-path traveling salesman problem with stochastic travel costs. *EURO Journal on Transportation and Logistics*, 6(1):3–23, 2017.

[20] Matthew B Rhudy, Roger A Salguero, and Keaton Holappa. A kalman filtering tutorial for undergraduate students. *International Journal of Computer Science & Engineering Survey (IJCSES)*, 8:1–18, 2017.

[21] Robert Reid and Thomas Bräunl. Large-scale multi-robot mapping in magic 2010. In *2011 IEEE 5th International Conference on Robotics, Automation and Mechatronics (RAM)*, pages 239–244. IEEE, 2011.

[22] Christian Forster, Simon Lynen, Laurent Kneip, and Davide Scaramuzza. Collaborative monocular SLAM with multiple micro aerial vehicles. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3962–3970. IEEE, 2013.

[23] Jacopo Banfi, Alberto Quattrini Li, Nicola Basilico, and Francesco Amigoni. Communication-constrained multirobot exploration: Short taxonomy and comparative results. In *Proceedings of the IROS workshop on on-line decision-making in multi-robot coordination (DEMUR2015)*, pages 1–8, 2015.

[24] Diluka Moratuwage, Ba-Ngu Vo, and Danwei Wang. Collaborative multi-vehicle SLAM with moving object tracking. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5702–5708. IEEE, 2013.

[25] Gregory Dudek and Michael Jenkin. *Computational principles of mobile robotics*. Cambridge University Press, 2010.

Chapter 2

Background Theory and Literature Review

2.1 Introduction

This chapter will cover two key problems for multi-robot systems: task allocation (given a number of tasks to be completed, which robots should complete them, and in what order?), and localisation (where are the robots located?). State-of-the-art techniques are reviewed, and gaps in academic knowledge are listed.

2.2 Graph Theory

Multi-robot system algorithms often make use of graph theory [1]. Graph theory has been around for hundreds of years, and modelling MRS problems as graphs enables the use of a large history of mathematical formulation. A graph consists of *vertices*

and *edges*. Edges are the traversal between vertices. They have an associated weight, or cost, that is applied whenever they are traversed.

Both multi-robot task allocation and collaborative localisation utilise graph theory. In the MRTA case, the vertices represent tasks, and the edges represent the time or distance required for robots to move to and complete a task [2]. In the CL case, vertices represent robots, and edges represent communication of information [3].

It is often helpful to represent graphs visually, such as in Figure 2.1. It is possible to draw a graph any number of ways; the positions of the points and lines are simply there to illustrate connections. The points are not necessarily to scale, and the lines do not necessarily imply that they will be traversed in a straight line. A graph is considered *complete* if every pair of vertices is joined by an edge. Otherwise, it is called *incomplete*, such as when a robot can only communicate with its neighbours. A graph is considered *directed* if each edge is strictly one-way. Otherwise, it is considered *undirected*. A *path* is represented by a series of vertices (or edges), indicating traversal within a graph.

The *triangle inequality* is a property of triangles which states that that the sum of the lengths of any two sides must be greater than or equal to the length of the remaining side. This property can also be true for graphs: if travelling from vertex i to vertex j, it is not beneficial to detour through another vertex k. In particular, this is true for graphs where vertices represent spatial locations. This property is useful for many MRS algorithms, such as guaranteeing worst-case solution quality [4].

2.3 Task Allocation

The multi-robot task allocation problem can be represented as a graph. Consider a set of robots $R = \{r_1, r_2, \ldots, r_n\}$ and initial positions (depots) $D = \{d_1, d_2, \ldots, d_n\}$, and a set of target positions (tasks), $T = \{t_1, t_2, \ldots, t_m\}$. We form a graph with vertices $V = D \cup T$, and edges E joining any two vertices in V. The cost function for a robot $r \in R$ to traverse edge $e \in E$ is $c_e^r \in \mathbb{R}$, which reflects the time it takes that robot to traverse that edge.

For the mTSP, valid allocations consist of a list of robot paths which result in each vertex $v \in V$ being visited by a robot. Each robot will have a path $\mathbf{p}^r$ consisting of the edges that robot plans to traverse. For example, a path of length l for robot $r \in R$ is defined as a set of edges $\mathbf{p}^r = \{e_1, e_2, \ldots, e_l\}$. It is common requirement for robots to also return to their start position. A robot's path cost, *RPC*, is the sum of all edge costs in their path.

$$RPC(\mathbf{p}^r) = \sum_{i=1}^{k} c_{\mathbf{p}_i^r}^r \tag{2.1}$$

Allocation quality is defined by how well it meets a given objective. Two objectives are commonly used. If we wish to complete all tasks as energy efficiently as possible, we must minimise the sum of all robot paths (MiniSum). If we wish to complete all tasks as quickly as possible, we must minimise the maximum robot path (MiniMax).

$$\text{Minimise energy usage:} \qquad \text{MiniSum} = \min_{\mathbf{p}} \sum_{r} RPC(\mathbf{p}^r) \tag{2.2}$$

$$\text{Minimise completion time:} \qquad \text{MiniMax} = \min_{\mathbf{p}} \max_{r} RPC(\mathbf{p}^r) \tag{2.3}$$

While these could both be described as a minmax problem, this definition of MiniSum is linear, which allows us to use linear algorithms. This will be explored further in Chapter 2.

It is common to refer to optimisation problems, such as MRTA, as problems where one must search the *solution space* to find good solutions [5]. In this case, a 'solution' is an allocation of robots to tasks that results in all tasks being completed. The concept of a solution space is based on the idea that solutions can be sorted by similarity. Similar solutions, called neighbours, are defined as solutions that differ by a single transformation. For example, neighbours could be found by reversing a robot's path or swapping two tasks between robots. The solution space is rarely ever displayed graphically. For task allocation theory, it is sufficient to use an intuitive understanding that some allocations are more similar than others. As such, if an algorithm is said to search *nearby* or *local* solutions, this is synonymous to evaluating *similar* solutions. In comparison, *global* solutions refers to all possible solutions, regardless of similarity. A graphical representation can be seen in Figure 2.4.

MRTA problems can have very large solution spaces, which has led to a variety of algorithms that balance between solution quality and required processing time.

2.3.1 Algorithms

Integer Linear Programs

Linear programs can be used to solve problems which require minimising or maximising a linear function subject to linear constraints. Commercial solvers can solve linear programs optimally i.e. the solution that minimises or maximises the problem according to the cost function. The type of linear program is defined by its variables.

An integer linear program (ILP) is one where the variables are integers [6]. The definition is:

$$\text{minimise} \quad \mathbf{cx} \tag{2.4}$$

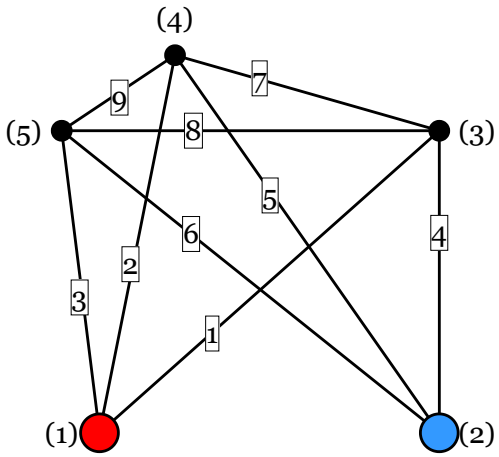
$$\text{subject to} \quad \mathbf{Ax} \leq \mathbf{b} \tag{2.5}$$

$$\mathbf{A_{eq}x} = \mathbf{b_{eq}} \tag{2.6}$$

In these equations, $\mathbf{x}$ represents the vector of decision variables to be solved, $\mathbf{c}$ is a vector of costs associated with each decision variable, $\mathbf{A}$ and $\mathbf{A_{eq}}$ are inequality matrices, and $\mathbf{b}$ and $\mathbf{b_{eq}}$ are inequality vectors.

Consider the mTSP in Figure 2.1. There are two robots and three tasks at given locations. We calculate a cost matrix that describes the time it takes for a robot to traverse edges. This cost matrix can also be described as a cost vector $\mathbf{c}$. The edge-usage vector $\mathbf{x}$ reflects how many times each edge is used in a solution. This vector is what we are solving for using the ILP. Example values of $\mathbf{x}$, with their graphical interpretation, are shown in Figure 2.2.

ILP constraints first require the definition of some variables. The set of edges, E, is split into two sets, edges between a robot and a task E_r, and edges between tasks E_t. We define a function that takes nodes as an input, and returns edges that are connected to those nodes. The function, $\delta(S) = \{(i, j)) \in E : i \in S, j \notin S\}$, takes a set of vertices and returns all edges that connect vertices in the set with vertices that are not in the set. The special case, $\delta(i)$, takes a single vertex $i \in V$ and returns all edges it is connected to. Our objective is to complete all tasks as energy efficiently as possible, which is achieved by minimising the sum of edge costs, known as the MiniSum objective.



Cost Matrix

Node from \ Node to	(1)	(2)	(3)	(4)	(5)
(1)	-	-	12.0	10.2	8.1
(2)	-	-	8	12.2	12.8
(3)	12.0	8	-	7.3	10
(4)	10.2	12.2	7.3	-	3.6
(5)	8.1	12.8	10	3.6	-

Cost Vector

Edge	1	2	3	4	5	6	7	8	9
	12.0	10.2	8.1	8	12.2	12.8	7.3	10	3.6

Fig. 2.1: An example multiple Travelling Salesman Problem (mTSP). Two robots (large coloured circles) must complete three tasks (small black circles). All vertices (robot or task) have been labelled from (1) to (5). Relevant edges have been labelled from 1 to 9. The time taken to traverse an edge can be reflected as a cost matrix or a cost vector.

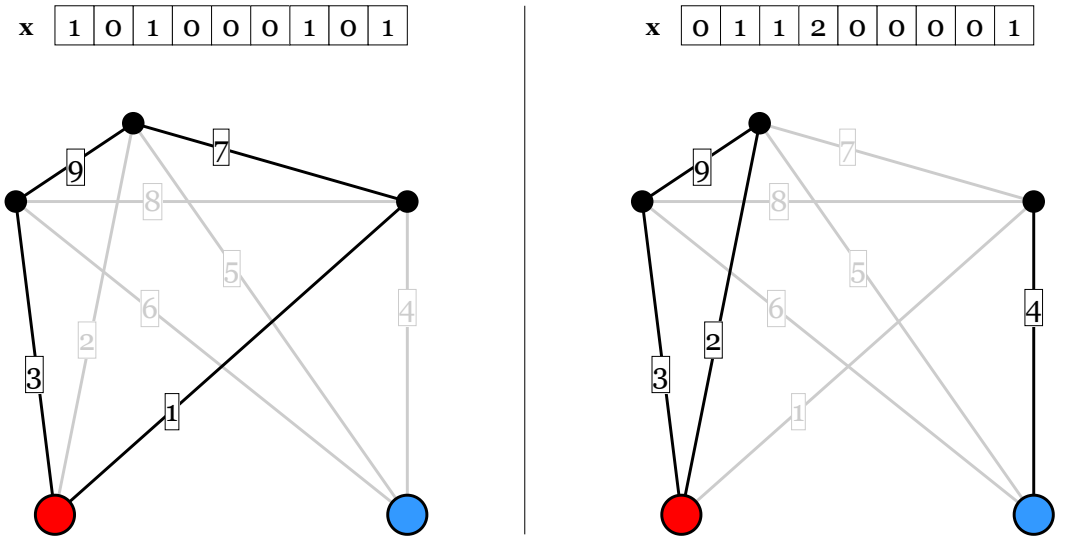


Fig. 2.2: Example solutions to a multiple Travelling Salesman Problem (mTSP). Two robots (large coloured circles) must complete three tasks (small black circles). A variable vector x indicates how many times each edge is used.

The ILP formulation for the mTSP is as follows:

$$\text{minimise} \quad \sum_{e \in E} \mathbf{c}_e \mathbf{x}_e \tag{2.7}$$

$$\text{subject to} \quad \mathbf{x}_{\delta(i)} = 2 \qquad \forall i \in T \tag{2.8}$$

$$\mathbf{x}_e \in \{0,1\} \qquad \forall e \in E_t \tag{2.9}$$

$$\mathbf{x}_e \in \{0,1,2\} \qquad \forall e \in E_r \tag{2.10}$$

$$\mathbf{x}_{\delta(S)} \geq 2 \qquad \forall i \in S, S \subseteq T \tag{2.11}$$

Equation (2.7) specifies the objective function, namely energy efficiency. The other common objective, fast task completion (MiniMax), is unable to be expressed as a linear equation, and is therefore unable to be solved using ILPs. Equation (2.8) ensures that exactly two connected edges are used for each vertex, because valid solutions require that each vertex is traversed to and traversed from. Equation (2.9) ensures that edges between tasks are used 0 or 1 times, because the optimal solution will never use the same edge more than once between tasks. This is a consequence of the triangle inequality, i.e., it is never beneficial to complete a task once it has already been completed. Equation (2.10) ensures that edges connected to a robot start position are used 0 times (robot is not used), 1 time (robot uses this edge to leave home, and uses another edge to return home), or 2 times (robot uses the same edge to leave and return home). Equation (2.11) is known as a sub-tour elimination constraint. Without this, ILPs may decide that the optimal solution is for robots to move to a task, teleport away, complete a small sub-tour of tasks, then teleport back. This is clearly impossible, but does not violate any of our other constraints. This particular sub-tour elimination constraint forces all vertex subsets to be connected to the rest of the vertices, preventing the use of teleportation. However, sub-tour constraints have limitations, as the number of possible subsets, S, scales enormously with the number

of vertices. As such, this constraint is applied iteratively during calculation, where we only constrain sub-tours that the ILP attempts to use.

The edge-usage variable, $\mathbf{x}$, can be calculated using a commercially available ILP solver, such as CPLEX or MATLAB's Optimization Toolbox. The solver dynamically removes large portions of the search space that are guaranteed to not contain optimal solutions, using techniques such as branch and cut [7]. Commercial solvers are often used due to the significant amount of optimisation that can be performed with parallelism and dynamic searching.

Several adaptations need to be made for solving the heterogeneous mTSP, although the underlying concept remains the same [8]. Each robot $k \in R$ has its own cost vector $\mathbf{c}_e^k$ and edge-usage variable $\mathbf{x}_e^k$. We also need to employ a binary variable $\mathbf{y}_i^k$ indicating if robot $k \in R$ is completing task $i \in T$. This binary variable is solved by the ILP, so we append it to $\mathbf{x}$ for actual calculation.

The heterogeneous formulation is as follows:

$$\text{minimise} \quad \sum_{k=1}^{n} \sum_{e \in E} \mathbf{c}_e^k \mathbf{x}_e^k \tag{2.12}$$

$$\text{subject to} \quad \mathbf{x}_{\delta(i)}^k = 2\mathbf{y}_i^k \qquad \forall i \in T, k \in R \tag{2.13}$$

$$\mathbf{x}_{\delta(S)}^k \geq 2\mathbf{y}_i^k \qquad \forall i \in S, S \subseteq T, k \in R \tag{2.14}$$

$$\sum_{k=1}^{n} \mathbf{y}_i^k = 1 \qquad \forall i \in T \tag{2.15}$$

$$\mathbf{x}_e^k \in \{0,1\} \qquad \forall e \in E_t, k \in R \tag{2.16}$$

$$\mathbf{x}_e^k \in \{0,1,2\} \qquad \forall e \in E_r, k \in R \tag{2.17}$$

$$\mathbf{y}_i^k \in \{0,1\} \qquad \forall i \in T, k \in R \tag{2.18}$$

Equation (2.12) specifies our objective function, once again minimising the sum of all costs. Equation (2.13) ensures that exactly two connected edges are used for each vertex for the robot that completes that task. All other robots should not use edges connected to that vertex. Equation (2.14) is the same sub-tour elimination constraint as before. Equation (2.15) ensures that each task is completed by one robot. Equation (2.16) ensures that edges between tasks are used 0 or 1 times. Equation (2.17) ensures that edges connected to a robot start position are used 0, 1, or 2 times. Equation (2.18) ensures that tasks are completed by a given robot 0 or 1 times.

Integer linear programs have been used to solve the Travelling Salesman Problem for over thirty years [9]. The guarantee of optimality makes them an incredibly useful tool. They are suitable for adding additional constraints such as time windows, carbon emissions, and cargo limitations [2, 10]. They are not without their weaknesses, however. It can be difficult representing problem objectives and constraints in a suitable format, leading to the inability to solve for certain objectives, notably the objective to complete tasks as quickly as possible. Application of constraints may also require clever thinking in order to apply them in a desirable way, such as the sub-tour elimination constraints of Equation (2.11). Another key weakness is the time taken to find solutions. ILP solvers apply search space reduction techniques dynamically, which means processing time can vary significantly. They also suffer from scaling, for example, a problem with 29 tasks took 0.6 seconds, another with 101 tasks took 20 minutes [8].

Nevertheless, there are many problems where these weaknesses are acceptable. Even for problems that require faster solutions, they are often used as a performance benchmark.

Auctions

In some cases of the multiple Travelling Salesman Problem, finding a good solution quickly is more important than waiting for a better solution. Many algorithms have been designed with this requirement in mind, known as *heuristics*. The fastest algorithms for many problems are typically greedy, making good choices for the short-term in the hope that the long-term result is a good solution [11]. For the mTSP, this is commonly performed using a task *auction*. One or more tasks are held for auction, the robots bid on these tasks, and then one or more of the tasks are sold to the robots. Examples of this include ordered single-item auctions, which auction and sell one task at a time; parallel single-item auctions, which auction and sell all tasks in a single auction round; and sequential single-item auctions, which auction all tasks but only sell one task each auction round. Sequential single-item auctions have been found to perform the best [12, 9]. An example of the sequential single-item auction process is shown in Figure 2.3.

In addition, sequential single-item auctions have been tailored for different types of problems and perform well against other algorithms. In harsh communication environments, sequential single-item auctions were able to successfully allocate to more robots that other auctioning methods [13]. For task allocation with temporal constraints, a variant on sequential single-item auctions produced more compact schedules, more tasks completed, and reduced computation time than other algorithms [14]. For task allocation requiring formation of robot coalitions, sequential single-item auction bidding was used to produce near-optimal coalitions [15].

It is also possible to auction bundles of tasks in combinatorial auctions, but both bidding and auction resolution for combinations are NP-hard [16]. Therefore usage of combinatorial auctions requires smart selections of combinations. In practice, se-

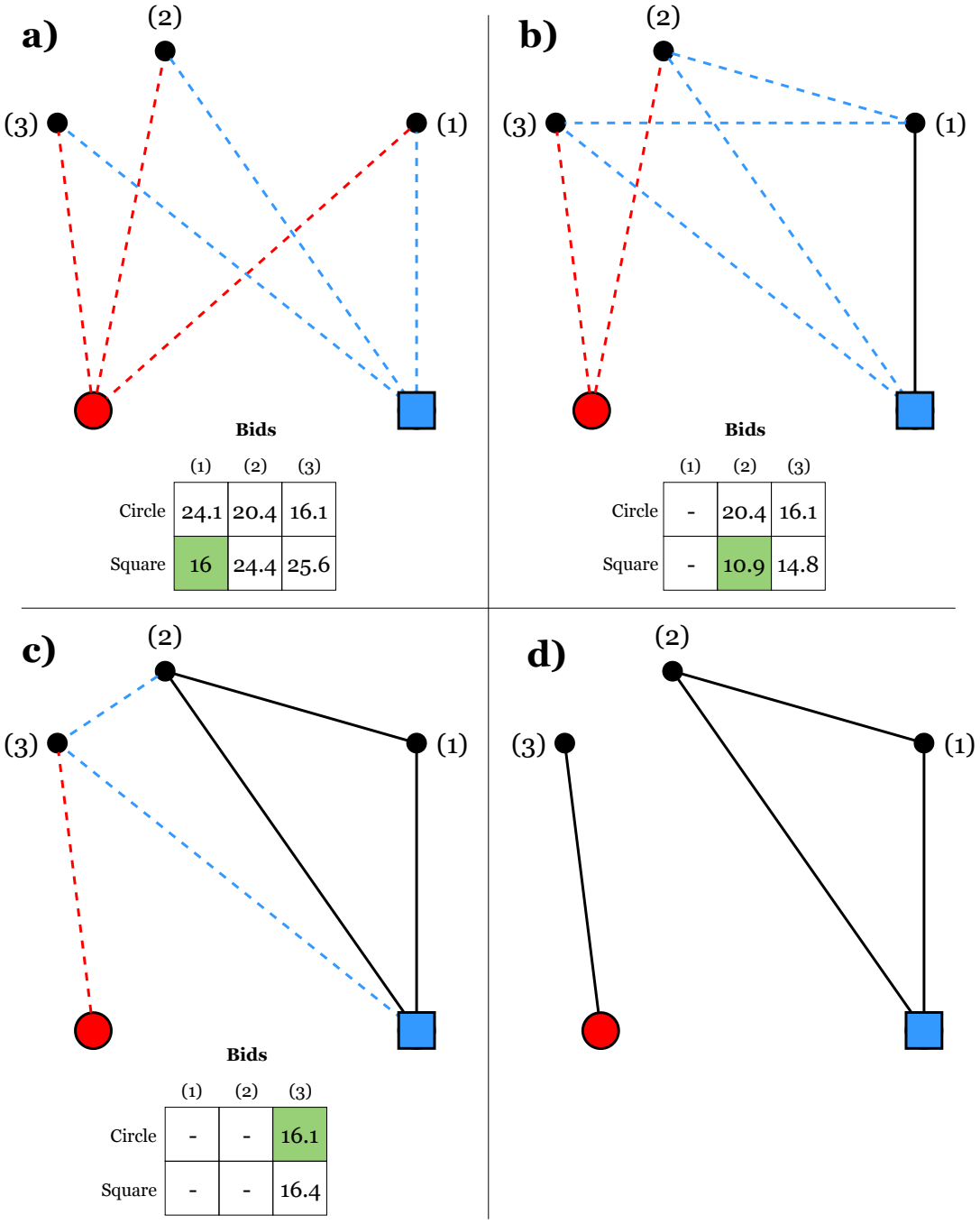


Fig. 2.3: The sequential single-item auction process. Robots (large circle and square) independently bid on tasks using the estimated time required to complete them. The lowest bidder gets allocated the task, and bids are updated accordingly. This repeats until all tasks are allocated to the robots.

quential single-item auctions provide similar results with far simpler implementations [16].

Sequential single-item auctions also provide some guarantees of allocation quality. In particular, for the energy efficiency objective, solutions are guaranteed to be no worse than two times the optimal case. In practice, solutions are usually close to 25% worse than optimal [4]. For the fastest completion objective (MiniMax), solutions are guaranteed to be no worse than $2N$ worse than optimal, where N is the number of robots. In practice, results are much closer, although it is difficult to quantify, and are system specific. Finding optimal solutions for the fast completion objective is an open problem due to its non-linearity.

Metaheuristics

Heuristic approaches use local decisions in the hope that the result is a globally good solution. These are often fast, sub-optimal, and specifically designed for each type of problem. Metaheuristics, however, are problem-independent. They provide an algorithmic framework, where problem-specific components are inserted to produce a heuristic. The generalisation is useful, but is not the only benefit of metaheuristics. They are able to overcome local optima that often limit heuristics. The book by Gendreau and Potvin on metaheuristics defines this succinctly [5]:

> "Meta-heuristics are problem-independent algorithms that develop heuristic solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space."

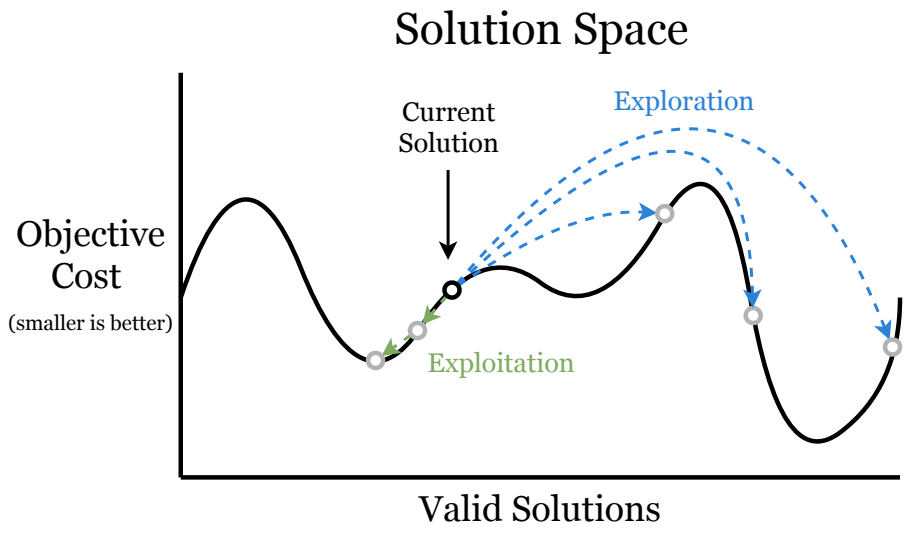


Fig. 2.4: A illustration of metaheuristic processes on the solution space. The objective is to find a solution with the smallest objective cost within a given time limit. Exploitation performs local improvement to find good solutions quickly, but is limited by local minima. Exploration can overcome local minima by finding different solutions, but will take much longer to find good solutions. A combination of these two processes are used by metaheuristics.

Metaheuristics divide the search process into exploration and exploitation [17]. Exploration searches promising areas of the solution space in an attempt to find new solutions that are significantly different to what has been found before. Exploitation refers to the local improvement procedures that search for similar solutions, ideally finding solutions that keep the good components while improving on the bad components. It is difficult to define what parts of a solution are good and bad, so local improvement is done by semi-randomly applying functions that modify the solution. A graphical representation of exploration and exploitation can be seen in Figure 2.4.

Finding a balance between exploration and exploitation is difficult. Prioritising local improvement increases the likelihood of finding good solutions quickly, but may struggle to overcome local optima. Prioritising exploration increases the likelihood of overcoming local optima, but will take a much longer time. As such, metaheuristics require tuning to find good solutions within an acceptable time frame. Metaheuristics are also unable to offer any guarantee of solution quality. The use of randomness

means it is possible to find optimal solutions, but it is also possible for it to not find any good solutions.

While metaheuristics can be applied to many different optimisation problems, it has been proven that no one metaheuristic can be best suited for all optimisation problems [18]. Hence, a large number of metaheuristics exist, and there are textbooks that discuss them all in detail [5]. An overview of commonly used metaheuristics for the mTSP is provided here:

Simulated annealing is inspired by manufacturing, where a material is initially heated then cooled slowly to achieve certain material properties. Simulated annealing metaheuristics start off 'hot', where they are willing to consider worse solutions to overcome local optima, and gradually become 'cool', where they only perform local improvement procedures. It is one of the simplest metaheuristics to implement, leading to its use for a wide variety of problems. It has been used to improve layout design for cellular manufacturing [19], along with fuzzy theory to assist tuning [20], as well as unmanned aerial vehicle inspections [21], and extended with GPU acceleration to improve computation speed [22]. It has also been used in conjunction with hierarchical task networks to generate new plans [23].

Tabu search performs local search until it reaches a local optima. It then evaluates worse solutions in an attempt to overcome the local optima, and prevents exploring previously evaluated solutions through tabu lists, which record recent search history. Other methods have attracted more recent research focus than tabu search, but remains a common candidate for hybrid metaheuristics, where tabu search is used to perform local improvement [24–26].

Ant colony optimisation is inspired by ants laying pheromones to guide other ants. In nature, it has been observed that ant movements are initially exploratory, but after some time the majority of ants follow efficient paths between a food source

and the nest [27]. The fundamental principle of ant colony optimisation is that ants iteratively build solutions, leaving behind a pheromone whose strength scales with solution performance. New ants will either follow pheromones or search for new solutions based on a stochastic decision. Eventually, the strongest smelling local decisions will produce an effective global solution. While often outperformed by other metaheuristics for the TSP and its variants [5], it has shown great performance in problems that change characteristics while being solved, such as ad hoc networking [28]. Nevertheless, it has been applied to wind farm optimisation [29], mTSP [30, 31], and multi-objective mTSP [32, 33].

One of the most common metaheuristic is the *Genetic Algorithm* (GA), inspired by biological evolution. A number of genes (solutions) are grouped to form a population. The genes can breed (crossover) and mutate to form new genes. The least-fit (worst scoring) genes are removed from the population. This process continues for many generations (iterations), so that the resulting population is likely to contain much better solutions than the original population. Genetic algorithms are one of the most popular metaheuristics for the mTSP and its variants [34–36], and thus is the metaheuristic utilised in this thesis. Many practical decisions need to be considered when implementing a GA.

GA Representation. Solutions need to be encoded as genes for use in crossover and mutation operations. Solutions to the mTSP need to encode which robot each task is allocated to, and the order that the robots will complete them. This can be done with a one-chromosome representation [37], which indicates robot allocations using separation tokens; the two-chromosome representation [38], which indicates robot allocations using a separate chromosome; or the two-part chromosome representation [39], which indicates robot allocations by positioning and path-length. These representations can be seen in Figure 2.5. The two-part chromosome representation is

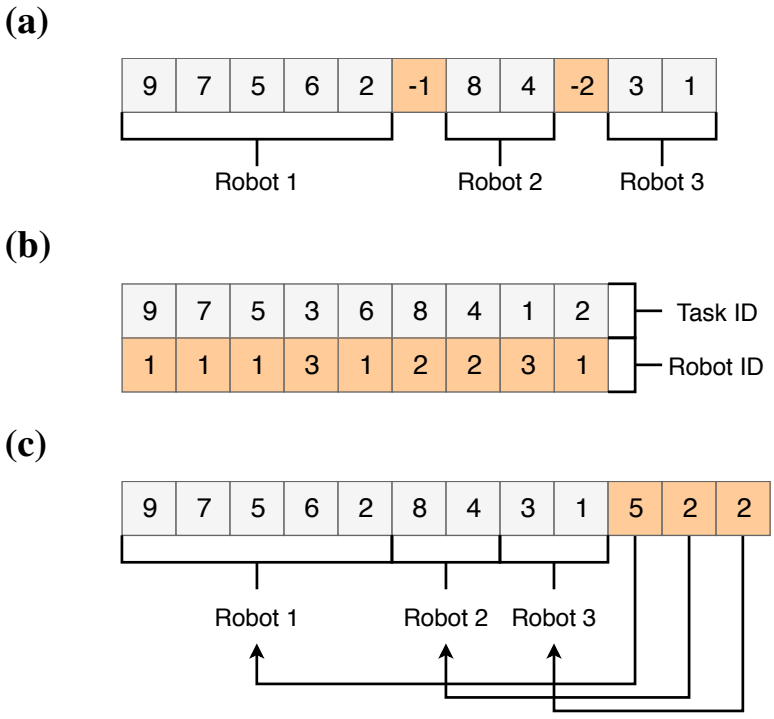


Fig. 2.5: Chromosome representations for the Multiple Travelling Salesman Problem (mTSP), where three robots must complete nine tasks. All three chromosomes represent the same solution. a) One-chromosome representation, which uses separation tokens. b) Two-chromosome representation, which uses an allocation chromosome. c) Two-part chromosome representation, which indicates allocations using robot path length.

considered standard because it has been shown to have fewer redundant solutions than other representations [39].

GA Population. A large population size provides greater opportunities to overcome local optima, but at the cost of extra computation time. Unfortunately, there is no formula to select population size [5], but those designed for the mTSP often use values between 10 and 250, scaling with number of tasks and robots [40, 36].

GA Initialisation. Initialising the population with random solutions provides a good initial spread for the GA to work from. Alternatively, seeding the GA with high-quality solutions from other algorithms can improve results by giving the GA

a head-start, but may induce premature convergence [41], i.e., getting stuck in local optima.

GA Selection. Members of the population are selected for breeding or mutation through a selection process. A number of selection processes exist, such as score-based or rank-based roulette wheel, where the chance to get selected scales with solution quality. There is also tournament selection, where solutions are grouped randomly, and the best solution in each group is selected. It has been shown that for the TSP case, tournament-based selection is best for small problem sizes, while rank-based selection scales better [42].

GA Crossover. Breeding is performed by crossover functions, which perform the exploration process. They take two solutions and produce a new valid solution that shares traits of both of the parents. While many generic functions exist, it is common to use one that is designed for the mTSP [41].

GA Mutation. Mutation performs the exploitation process. For the mTSP, common mutations include task-swapping between robots, and sub-path reversal. The latter performs the same process as a classic local improvement operator, called 2-opt [43].

GAs are also commonly used for multi-objective problems. Performing well in one objective usually requires performing poorly in another, so the goal is to produce a range of solutions. This makes population selection difficult, because it is unclear which solutions are the most fit. The most commonly used multi-objective GA, the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [44], handles this problem by sorting the population into Pareto fronts. Pareto fronts are constructed using the concept of solution *dominance*. One solution dominates another if it is greater for at least one objective, and not worse for any objectives. Solutions in the population that are not dominated by any other are put in the first Pareto front. Solutions that are

dominated by those in the first front are put in the second front, and so on. Ranking within a front is performed using a crowding distance metric to promote creating a wide range of solutions. Algorithms for the mTSP have used NSGA-II [40] and new multi-objective algorithms are often inspired by NSGA-II [32].

What we can learn from literature is that even though metaheuristics are problem-independent, applying them to problems still requires research on solution representation, exploration and exploitation functions, and tuning values. As such, research on their application to multi-robot systems is still in its infancy.

2.3.2 Heterogeneity

With heterogeneous robots, the multiple Travelling Salesman Problem becomes the heterogeneous multiple travelling salesman problem (H-mTSP). While heterogeneous systems are more complex than their homogeneous counterparts, there are a number of advantages to using them. It is more cost effective to use robots with different skill-sets, rather than outfitting every robot to have every skill. It can also be infeasible to create a single robot with every necessary skill, such as a nimble robot that carries enormous weights.

The H-mTSP has only been considered recently, and while some algorithms now exist that handle some levels of heterogeneity, there are many open questions about the effectiveness of homogeneous algorithms on heterogeneous problems, and how to design heterogeneous robot teams [45].

One aspect all reviewed papers have in common is representing heterogeneous skill sets using different cost vectors for each robot. In cases where robots are completely unable to complete certain tasks (i.e. costs are infinite), these are often excluded for selection by representing them as problem constraints.

Exact solutions with integer linear programs exist [8], and are discussed in more detail in Section 2.3.1.

As for heuristics, auctions designed for homogeneous systems have been successfully applied to heterogeneous systems. For example, this has been done in a multi-robot system for healthcare facilities [46], an allocation process for dynamically appearing tasks [47], allocation for tightly-coupled systems [15], disaster response [48]. It has also been used as part of distributed systems [49], and in a hierarchical cloud-computed robotic systems [50]. This is despite the fact that sequential single-item auctions lose their mathematical guarantees when applied to heterogeneous systems, as is shown in greater detail in Chapter 3. There has not been research on whether other auctioning techniques may be superior in heterogeneous systems.

Metaheuristics originally applied to homogeneous systems have been applied to the heterogeneous case for many flavours of the H-mTSP problem [51–55]. Unlike auctions, however, there is nothing to suggest that heterogeneity has not been accounted for, as metaheuristics are tuned.

To summarise, there is potential for auctions to be improved when applied to heterogeneous systems. In addition to this, no published works could be found for allocating tasks to robots that can change their abilities, such as by equipping tools.

2.4 Collaborative Localisation

Collaborative (or Cooperative) localisation (CL) is the act of multiple robots helping one another localise. Regular localisation is often done by fusing sensor data in an Extended Kalman Filter. Consider a single robot which we wish to localise, i.e., we

Table 2.1: Variables for an Extended Kalman Filter.

Variable	Description	Dimension
$\hat{\mathbf{x}}_{k\|k}$	State estimate (current)	n_x x 1
$\hat{\mathbf{x}}_{k-1\|k-1}$	State estimate (previous)	n_x x 1
$\hat{\mathbf{x}}_{k\|k-1}$	Predicted state	n_x x 1
$\mathbf{P}_{k\|k}$	State covariance (current)	n_x x n_x
$\mathbf{P}_{k-1\|k-1}$	State covariance (previous)	n_x x n_x
$\mathbf{P}_{k\|k-1}$	Predicted state covariance	n_x x n_x
$\mathbf{u}_k$	Control input	n_u x 1
f	State transition function	-
$\mathbf{F}_k$	State transition Jacobian	n_x x n_x
$\mathbf{Q}_k$	State transition covariance	n_x x n_x
$\mathbf{z}_k$	Sensor measurement	n_z x 1
$\mathbf{R}_k$	Sensor measurement covariance	n_z x n_z
$\tilde{\mathbf{y}}_k$	Innovation	n_z x 1
$\mathbf{S}_k$	Innovation covariance	n_z x n_z
$\mathbf{K}_k$	Kalman gain	n_x x n_z
h	Observation function	-
$\mathbf{H}_k$	Observation Jacobian	n_z x n_x

want to know where it is. Let's say it started in a known position, and we have told it to move 1 metre forwards. There are a few sources of information that we could use.

- Assume it did what it was told to do, i.e., we use the *control input*.
- Use a sensor to measure position.
- Estimate position from velocity. More generally, use the current state to calculate a future state according to the robots motion model, known as *state transition*.

Each source of information has its own strengths and weaknesses. Control input is always available, but does not measure what is really happening. It therefore accumulates error, resulting in growing localisation uncertainty. Sensors measure what is happening, but may not always be available, and may be subject to errors and noise. State transition accumulates error, but is always available and may reflect what is really happening, depending on how the previous state was calculated.

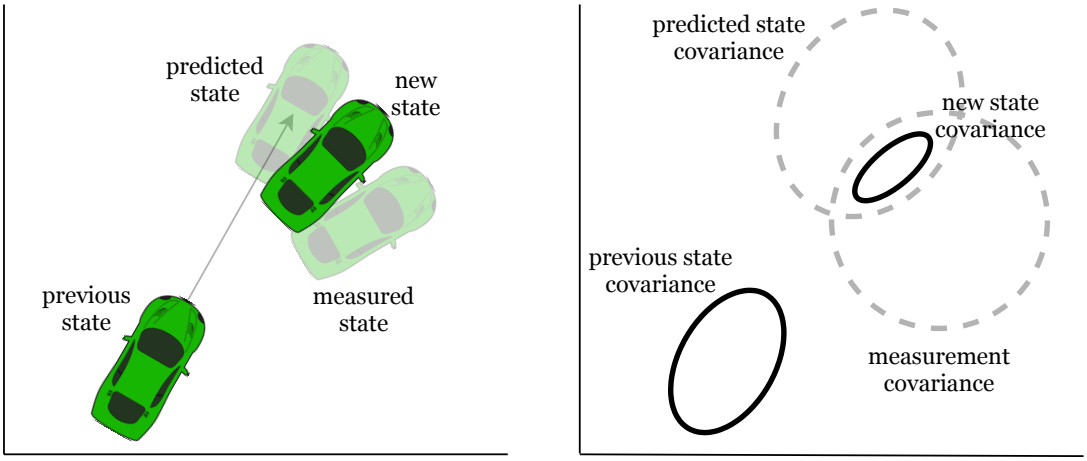


Fig. 2.6: A visual representation of a Kalman filter. It predicts a state from a previous state, and fuses it with sensor measurements to form new state estimates. It requires each state to have a covariance, which specifies the uncertainty of that state.

2.4.1 Extended Kalman Filter

Instead of picking a single source of information, a Kalman Filter fuses all of them together. It consists of two phases. A *predict* phase uses control input and a state transition to predict how the robot has moved since our last state estimate. An *update* phase fuses sensor measurements into the predicted state. The fusion is a weighted average of the predicted state and the measured state. This weighting, defined as the Kalman gain $\mathbf{K}$, is calculated from the self-reported accuracy of each state. The self-reported accuracy is known as *covariance*, stored as a square matrix. A visual diagram of Kalman filtering can be seen in Figure 2.6, and a list of variable definitions can be seen in Table 2.1. Kalman filters operate on linear systems, but vehicle localisation is non-linear. A non-linear extension on the Kalman filter is the Extended Kalman Filter. The prediction equations are as follows:

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k) \tag{2.19}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k \tag{2.20}$$

The predicted state ($\hat{\mathbf{x}}_{k|k-1}$) is calculated by state-transitioning (f) the previous state ($\hat{\mathbf{x}}_{k-1|k-1}$) along with control input ($\mathbf{u}_k$). The predicted state covariance ($\mathbf{P}_{k|k-1}$) is calculated using a linear approximation of the state transition ($\mathbf{F}_k$), the previous covariance ($\mathbf{P}_{k-1|k-1}$), and adding an error accumulation constant ($\mathbf{Q}_k$). The linear approximation of the state transition is known as the state-transition Jacobian, which is a partial derivative of the state-transition function with respect to each variable:

$$\mathbf{F}_k = \frac{\partial f}{\partial x}\Big|_{\hat{\mathbf{x}}_{k-1|k-1},\mathbf{u}_k} \tag{2.21}$$

The update phase consists of the steps:

$$\text{Innovation:} \quad \tilde{\mathbf{y}}_k = \mathbf{z}_k - h(\hat{\mathbf{x}}_{k|k-1}) \tag{2.22}$$

$$\text{Innovation Covariance:} \quad \mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \tag{2.23}$$

$$\text{Kalman Gain:} \quad \mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1} \tag{2.24}$$

$$\text{Updated State:} \quad \hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \tag{2.25}$$

$$\text{Updated Covariance:} \quad \mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \tag{2.26}$$

The innovation ($\tilde{\mathbf{y}}_k$) is the difference between the measured state ($\mathbf{z}_k$) and the predicted state ($\hat{\mathbf{x}}_{k|k-1}$), looking only at the variables that are measured by that sensor using the observation function (h). The covariance of the innovation ($\mathbf{S}_k$) uses a linear approximation of the observation function ($\mathbf{H}_k$), the predicted state covariance ($\mathbf{P}_{k|k-1}$), and the measurement covariance ($\mathbf{R}_k$). The Kalman gain ($\mathbf{K}_k$) is calculated from the predicted state covariance ($\mathbf{P}_{k|k-1}$), the linear approximation of the observation function ($\mathbf{H}_k$), and the innovation covariance ($\mathbf{S}_k$). The new state estimate ($\hat{\mathbf{x}}_{k|k}$) is then calculated using a weighted average of the predicted state ($\hat{\mathbf{x}}_{k|k-1}$) and the measured state ($\mathbf{z}_k$) using the innovation ($\tilde{\mathbf{y}}_k$). The new state covariance ($\mathbf{P}_{k|k}$) uses the predicted

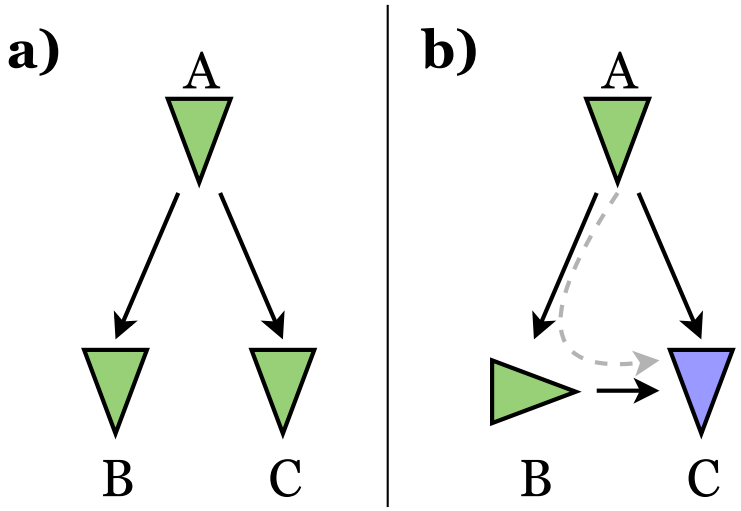


Fig. 2.7: Robots (triangles) communicate localisation information (arrows). a) The information received by each robot is independent. b) Robot C receives duplicate information, once from A directly, again from A through B. Special filters are required to fuse the information appropriately.

state covariance ($\mathbf{P}_{k|k-1}$), and lowers it according to the weighting ($\mathbf{K}_k$) of the linear approximation of the observation function ($\mathbf{H}_k$).

EKFs assume that all information is independent of one another. While this is true for most robot systems, where each source of information is from a different sensor, it can be false when using communicated information. For example, consider Figure 2.7. Robot *A* affects robot *B*'s localisation, which in turn will affect robot *C*. If all data is treated as being independent, the result will be biased towards repeated information, which will result in localisation overconfidence. EKFs provide guaranteed convergence as long as the filter never becomes overconfident, meaning that this improper data fusion is not only suboptimal, but may cause it to lose many helpful mathematical properties. This problem is known as *data incest* [56]. This term is attention-grabbing, but also rather fitting: it occurs when one uses related data in a way that should only be done if they are not related.

2.4.2 Handling Data Incest

New algorithms have been designed to prevent data incest. The oldest such algorithm is a centralised EKF [57]. Instead of each robot using their own EKF, a central EKF can be used to fuse all sensor information and inter-robot observations. The state vector ($\hat{\mathbf{x}}$) is expanded to contain the poses and twists of every robot in the system. Information does not need to propagate from robot to robot, so it will not be duplicated. While effective, it requires reliable and fast communication that is not scalable to larger systems. Many algorithms use this as a benchmark for comparison.

Covariance Intersection [58, 3] is a filter that is fundamentally similar to the EKF, but assumes all information is dependent. The same predict phase is used. For the update phase, it still averages between the predicted and measured states, but uses a different weighting according to a parameter ω, $0 \leq \omega \leq 1$. This parameter is chosen by any standard optimisation method to minimise the trace or determinant of the resulting state covariance ($\mathbf{P}_{k|k}$), i.e., choose the value of ω that results in the most likely state estimate. The update equations are as follows:

$$\mathbf{P}_{k|k}^{-1} = \omega \mathbf{P}_{k|k-1}^{-1} + (1-\omega)\mathbf{R}_k^{-1} \tag{2.27}$$

$$\hat{\mathbf{x}}_{k|k} = \mathbf{P}_{k|k}[\omega \mathbf{P}_{k|k}^{-1}\hat{\mathbf{x}}_{k|k-1} + (1-\omega)\mathbf{R}_k^{-1}\mathbf{z}_k] \tag{2.28}$$

A graphical representation of this method is shown in Figure 2.8. Given two estimates with covariances P_1 and P_2, the resulting covariance will occur within the intersection. If P_1 and P_2 are completely independent, the resulting covariance will be smaller than the intersection. If they are completely dependent, the result exactly matches the intersection. For example, if a robot were given multiple sensor readings of position, it should become more confident in its position, expressed as a shrinking covariance. If instead it were given the same sensor reading repeatedly, it should not

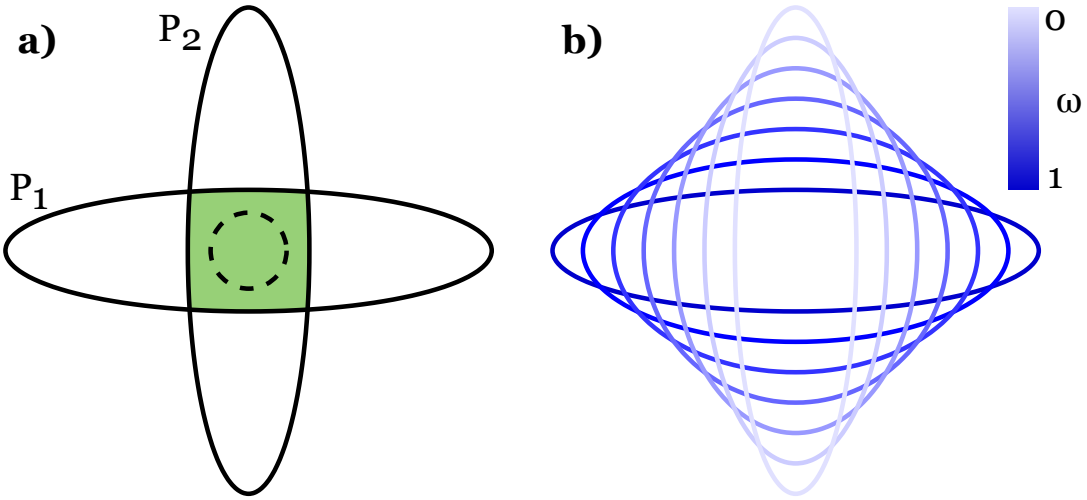


Fig. 2.8: A graphical representation of fusion of multiple sources of information. a) Two estimates with covariances P_1 and P_2 are fused. If they are dependent, the result should be the intersection (shaded region). If they are independent, the result should be smaller (dashed circle). b) The output of the Covariance Intersection algorithm, which guarantees a Gaussian result that will never be overconfident (smaller) than the best result. It has a parameter ω which can be used to find the resulting covariance with the smallest size.

become more confident because it has not received new information. The resulting covariance from Covariance Intersection can be seen in Figure 2.8. We can see that regardless of the choice of ω, the resulting covariance will always be larger than the intersection, hence the result will never be overconfident. The result is also guaranteed to be Gaussian.

While it prevents data incest, it is very pessimistic in its estimations. More recently, a split covariance intersection filter has been used [59], which splits the covariance matrix ($\mathbf{P}_{k|k}$) into one that stores the maximum possible dependence with other states ($\mathbf{P}_{d,k|k}$) and one that stores the known degree of independence ($\mathbf{P}_{i,k|k}$). This treats inter-robot detections as potentially dependent, while treating on-board sensors as

independent. The split covariance intersection update phase is as follows:

$$\mathbf{P}_1 = \mathbf{P}_{d,k|k-1}/\omega + \mathbf{P}_{i,k|k-1} \tag{2.29}$$

$$\mathbf{P}_2 = \mathbf{R}_{d,k}/(1-\omega) + \mathbf{R}_{i,k} \tag{2.30}$$

$$\mathbf{K}_k = \mathbf{P}_1(\mathbf{P}_1 + \mathbf{P}_2)^{-1} \tag{2.31}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{x}}_{k|k-1}) \tag{2.32}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k)\mathbf{P}_1 \tag{2.33}$$

$$\mathbf{P}_{i,k|k} = (\mathbf{I} - \mathbf{K}_k)\mathbf{P}_{i,k|k-1}(\mathbf{I} - \mathbf{K}_k)^T + \mathbf{K}_k\mathbf{R}_{i,k}\mathbf{K}_k^T \tag{2.34}$$

$$\mathbf{P}_{d,k_k} = \mathbf{P}_{k|k} - \mathbf{P}_{i,k|k} \tag{2.35}$$

This technique can also be applied to other filters. For example, the Ensemble Kalman Filter (EnKF) is a Monte Carlo variant of the Kalman filter which uses samples to represent state estimates. It operates similarly to a particle filter, but assumes all probability distributions are Gaussian. Some advantages of this format are that it readily supports non-linear prediction and observation models, and scales better with the size of the state vector. A Common Past-Invariant EnKF (CPI-EnKF) splits sample sets into dependent and independent sets, in a manner similar to the covariance split performed for Split Covariance Intersection. The CPI-EnKF can inherently retain correlation information from the ordered random sample sets. As such, a CPI-EnKF has shown good results for collaborative localisation [56].

Recent research has been addressing the application of these algorithms on real systems. Real systems have information delays, non-Gaussian noise, and unreliable detections. Recent research in this area addresses methods for lowering communication requirements [60], applying these algorithms on real robots for improved SLAM [61] and centralised tracking [62], and implementing these algorithms on established robot hierarchies [63, 64].

There are still a number of unanswered questions about collaborative localisation. Decisions regarding the number of robots, the required accuracy of sensors, the speed of the robots, and the rate of inter-robot detections will all influence CL effectiveness. Yet it is unclear what system properties make collaborative localisation algorithms effective, and thus deciding how one should structure a multi-robot system to best utilise CL.

2.4.3 Localisation with Allocation

The most common MRTA objectives, as mentioned earlier, involve minimising the total time and energy usage required to complete all tasks. It has been noted that the majority of studies consider the same few objective functions and constraints, despite many other functions being relevant to industry [10]. Hence, there has been some emerging work on problems with additional requirements. In particular, it is often assumed that all robots can localise sufficiently well, are able to communicate at all times, and will detect their own hardware failure. While this is true in some cases, there are situations where robots do not have reliable GPS, are not able to report that they are broken, and can roam beyond their communication range. This could be accommodated by using collaboration between robots during task completion.

Literature has addressed certain problems where robots must collaborate for certain tasks to be completed, such as transportation vehicles needing to meet at the same location to transfer goods [65], refuelling stations only servicing one robot at a time [66], and drones being used for short-range delivery while returning to a truck for long-range movement [67]. These problems represent collaboration as constraints which only require the consideration of the time of arrival at each task, and can therefore be expressed as linear equations. It is common to use an Integer-Linear Program to solve these problems, discussed in Section 2.3.1. ILPs can actually perform faster with

additional constraints, as it allows them to remove larger portions of the search space. However, some constraints apply to the entirety of a robot's path, such as ensuring robots stay within distance of one another. These use Euclidean distance, which cannot be represented as linear, so ILPs cannot be used.

Some research considers inter-robot distance constraints without consideration of task allocation, i.e., all robots work to complete a single task. This includes the optimal placement of robots to provide the best possible network connectivity [68], as well as searching strategies while maintaining sufficient connectivity to relay video [69]. A comparison of four communication-based exploration techniques is available, along with a taxonomy of communication-constrained exploration types [70]. When applied to real systems, it is common for an outer control layer to make a plan and for an inner control layer to ensure connectivity remains strong [71, 72]. If we were to apply these techniques to a system with multiple tasks, the robots will be much slower and less energy efficient than a system that uses its multiple robots to complete tasks in parallel.

Algorithms have been developed to address the problem of multi-robot task allocation with inter-robot distance constraints. A sequential-auction based heuristic has been created to constrain distance between robots, known as Connected Nearest Neighbour [73], which alternates between target allocation and distance maintenance phases. A genetic algorithm has been used to search for valid solutions, using a customised improvement mechanism that turns solutions that exceed the inter-robot distance into ones that do not [74]. However, both these approaches assume that robots move in straight lines between tasks, which would potentially violate the inter-robot distance constraint if the robots use obstacle avoidance. For example, two robots could move different ways around an obstacle, causing them to move too far away from one another.

Inter-robot distance requirements have not been sufficiently considered for multi-robot task allocation. Forcing robots to stay near one another allows them to localise using collaborative localisation, detect robot failure through visual detection, and communicate new plans to handle dynamic scenarios. While some algorithms for this problem exist, their performance is not guaranteed when applied to real robots, such as when including collision avoidance.

2.5 Research Gaps and Objectives

As can be seen from the literature review, a number of gaps exist for heterogeneous task allocation and collaborative localisation. Heterogeneous multi-robot systems that require fast solutions often make use of sequential single-item auctions, which have been designed for homogeneous systems. A heterogeneous version may be able to exploit known heterogeneity to improve solution quality. These systems also assume that robot abilities are constant, even though it is possible for robots to alter their abilities through tools and modular reconfiguration. Collaborative localisation is a powerful tool for making robots robust to hardware failure, but theory is lacking quantitative analysis on what systems properties make the best candidates for improvement using CL. In addition, the only available task allocation algorithms that enable collaborative localisation are insufficient for use in most real situations.

This thesis aims to fill these gaps through development towards heterogeneous task allocation and collaborative localisation.

Objective 1: Improve the performance of sequential single-item auctions for heterogeneous systems

Many researchers have used sequential single-item auctions to find fast solutions to the heterogeneous multiple Travelling Salesman Problem. However, they were designed and analysed for homogeneous systems. New bidding rules and auction resolution rules that consider heterogeneity could improve the speed and quality of solutions found.

Objective 2: Create and characterise solution methods for systems with dynamic heterogeneity

Heterogeneous robots are often assumed to have static capabilities. This is not true if robots can equip and use tools as necessary, or alter their structure using modular reconfiguration. It is unclear how available algorithms will perform for this new problem, including auctions, metaheuristics, and integer linear programs.

Objective 3: Provide quantitative information on when collaborative localisation is effective and efficient

Collaborative localisation algorithms have been created to properly fuse inter-robot detections with on-board sensors. However, implementing observations, communications, and fusion is not a simple task. There is no research on what system conditions are suitable for CL. This would be useful information to help decide whether or not a system should implement CL.

Objective 4: Apply collaborative localisation as part of task allocation

The multiple Travelling Salesman Problem and its variants often assume that robots can operate independently. This assumption can lead to robots becoming lost if their sensors fail, unaware if a robot dies, and out of range to communicate information. An algorithm that considers inter-robot distance as part of task allocation would be beneficial in making robots more robust to these circumstances.

The following chapters consist of accepted and submitted publications addressing each objective. As such, they each contain a brief summary of the publication, a statement of authorship, and an objective-specific literature review.

2.6 References

[1] SG Shirinivas, S Vetrivel, and NM Elango. Applications of graph theory in computer science: An overview. *International Journal of Engineering Science and Technology*, 2(9):4610–4621, 2010.

[2] Çağrı Koç, Tolga Bektaş, Ola Jabali, and Gilbert Laporte. Thirty years of heterogeneous vehicle routing. *European Journal of Operational Research*, 249(1):1–21, 2016.

[3] SJ Julier and Jeffrey K Uhlmann. General decentralized data fusion with covariance intersection. *Handbook of multisensor data fusion: theory and practice*, pages 319–344, 2009.

[4] Michail G Lagoudakis, Evangelos Markakis, David Kempe, Pinar Keskinocak, Anton J Kleywegt, Sven Koenig, Craig A Tovey, Adam Meyerson, and Sonal Jain. Auction-based multi-robot routing. In *Robotics: Science and Systems*, volume 5, pages 343–350. Rome, Italy, 2006.

[5] Michel Gendreau and Jean-Yves Potvin. *Handbook of metaheuristics*, volume 2. Springer, 2010.

[6] George Dantzig. *Linear programming and extensions*. Princeton university press, 2016.

[7] Hamdy A Taha. *Integer programming: theory, applications, and computations*. Academic Press, 2014.

[8] Kaarthik Sundar and Sivakumar Rathinam. Algorithms for heterogeneous, multiple depot, multiple unmanned vehicle path planning problems. *Journal of Intelligent & Robotic Systems*, pages 1–14, 2016.

[9] Harlan Crowder and Manfred W Padberg. Solving large-scale symmetric travelling salesman problems to optimality. *Management Science*, 26(5):495–509, 1980.

[10] Jairo R Montoya-Torres, Julián López Franco, Santiago Nieto Isaza, Heriberto Felizzola Jiménez, and Nilson Herazo-Padilla. A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering*, 79:115–129, 2015.

[11] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[12] Eric Schneider, Ofear Balas, A Tuna Ozgelen, Elizabeth I Sklar, and Simon Parsons. Evaluating auction-based task allocation in multi-robot teams. In *AAMAS Workshop: ARMS*, 2014.

[13] Michael Otte, Michael Kuhlman, and Donald Sofge. Multi-robot task allocation with auctions in harsh communication environments. In *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 32–39. IEEE, 2017.

[14] Ernesto Nunes and Maria Gini. Multi-robot auctions for allocation of tasks with temporal constraints. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[15] Gautham P Das, Thomas Martin McGinnity, and Sonya A Coleman. Simultaneous allocations of multiple tightly-coupled multi-robot tasks to coalitions

of heterogeneous robots. In *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*, pages 1198–1204. IEEE, 2014.

[16] S Koenig, C Tovey, M Lagoudakis, V Markakis, D Kempe, P Keskinocak, A Kleywegt, A Meyerson, and S Jain. The power of sequential single-item auctions for agent coordination. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 1625. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press, 2006.

[17] Olusegun Olorunda and Andries P Engelbrecht. Measuring exploration/exploitation in particle swarms using swarm diversity. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 1128–1134. IEEE, 2008.

[18] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[19] Mohammad Mahdi Paydar, Iraj Mahdavi, Iman Sharafuddin, and Maghsud Solimanpur. Applying simulated annealing for designing cellular manufacturing systems using MDmTSP. *Computers & Industrial Engineering*, 59(4):929–936, 2010.

[20] Jerzy Grobelny and Rafał Michalski. A novel version of simulated annealing based on linguistic patterns for solving facility layout problems. *Knowledge-Based Systems*, 124:55–69, 2017.

[21] Lucas P Behnck, Dionisio Doering, Carlos Eduardo Pereira, and Achim Rettberg. A modified simulated annealing algorithm for SUAVs path planning. *IFAC-PapersOnLine*, 48(10):63–68, 2015.

[22] Tolgahan Turker, Guray Yilmaz, and Ozgur Koray Sahingoz. GPU-accelerated flight route planning for multi-UAV systems using simulated annealing. In

International Conference on Artificial Intelligence: Methodology, Systems, and Applications, pages 279–288. Springer, 2016.

[23] Alejandro R Mosteo and Luis Montano. Simulated annealing for multi-robot hierarchical task allocation with flexible constraints and objective functions. In *Workshop on Network Robot Systems: Toward Intelligent Robotic Systems Integrated with Environments". IROS*, 2006.

[24] Guohui Zhang, Liang Gao, and Yang Shi. A genetic algorithm and tabu search for multi objective flexible job shop scheduling problems. In *Computing, Control and Industrial Engineering (CCIE), 2010 International Conference on*, volume 1, pages 251–254. IEEE, 2010.

[25] Qiao Zhang, Hervé Manier, and M-A Manier. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. *Computers & Operations Research*, 39(7):1713–1723, 2012.

[26] John Willmer Escobar, Rodrigo Linfati, Paolo Toth, and Maria G Baldoquin. A hybrid granular tabu search algorithm for the multi-depot vehicle routing problem. *Journal of Heuristics*, 20(5):483–509, 2014.

[27] J-L Deneubourg, Serge Aron, Simon Goss, and Jacques Marie Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3(2):159–168, 1990.

[28] Hang Zhang, Xi Wang, Parisa Memarmoshrefi, and Dieter Hogrefe. A survey of ant colony optimization based routing protocols for mobile ad hoc networks. *IEEE Access*, 5:24139–24161, 2017.

[29] Ramu Srikakulapu and U Vinatha. Optimized design of collector topology for offshore wind farm based on ant colony optimization with multiple travelling

salesman problem. *Journal of Modern Power Systems and Clean Energy*, pages 1–12, 2018.

[30] Soheil Ghafurian and Nikbakhsh Javadian. An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems. *Applied Soft Computing*, 11(1):1256–1262, 2011.

[31] Weiqin Wu, Yu Tian, and Tongdan Jin. A label based ant colony algorithm for heterogeneous vehicle routing with mixed backhaul. *Applied Soft Computing*, 47:224–234, 2016.

[32] Sahar Trigui, Omar Cheikhrouhou, Anis Koubaa, Uthman Baroudi, and Habib Youssef. Fl-mtsp: a fuzzy logic approach to solve the multi-objective multiple traveling salesman problem for multi-robot systems. *Soft Computing*, 21(24):7351–7362, 2017.

[33] Xinye Chen, Ping Zhang, Guanglong Du, and Fang Li. Ant colony optimization based memetic algorithm to solve bi-objective multiple traveling salesmen problem for multi-robot systems. *IEEE Access*, pages 21745–21757, 2018.

[34] Habibeh Nazif and Lai Soon Lee. Optimised crossover genetic algorithm for capacitated vehicle routing problem. *Applied Mathematical Modelling*, 36(5):2110–2117, 2012.

[35] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012.

[36] Ali AR Hosseinabadi, Maryam Kardgar, Mohammad Shojafar, Shahaboddin Shamshirband, and Ajith Abraham. GELS-GA: hybrid metaheuristic algorithm for solving multiple travelling salesman problem. In *Intelligent Systems Design*

and Applications (ISDA), 2014 14th International Conference on, pages 76–81. IEEE, 2014.

[37] Lixin Tang, Jiyin Liu, Aiying Rong, and Zihou Yang. A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex. *European Journal of Operational Research*, 124(2):267–282, 2000.

[38] Charles J Malmborg. A genetic algorithm for service level based vehicle scheduling. *European Journal of Operational Research*, 93(1):121–134, 1996.

[39] Arthur E Carter and Cliff T Ragsdale. A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European Journal of Operational Research*, 175(1):246–257, 2006.

[40] R Bolaños, M Echeverry, and J Escobar. A multiobjective non-dominated sorting genetic algorithm (NSGA-II) for the multiple traveling salesman problem. *Decision Science Letters*, 4(4):559–568, 2015.

[41] Shuai Yuan, Bradley Skinner, Shoudong Huang, and Dikai Liu. A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. *European Journal of Operational Research*, 228(1):72–82, 2013.

[42] Noraini Mohd Razali and John Geraghty. Genetic algorithm performance with different selection strategies in solving TSP. In *Proceedings of the World Congress on Engineering*, volume 2, pages 1134–1139, 2011.

[43] David S Johnson. Local optimization and the traveling salesman problem. In *International Colloquium on Automata, Languages, and Programming*, pages 446–461. Springer, 1990.

[44] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[45] Lynne E Parker, Daniela Rus, and Gaurav S Sukhatme. *Multiple Mobile Robot Systems*, pages 1335–1384. Springer, 2016.

[46] Gautham P Das, Thomas M McGinnity, Sonya A Coleman, and Laxmidhar Behera. A distributed task allocation algorithm for a multi-robot system in healthcare facilities. *Journal of Intelligent & Robotic Systems*, 80(1):33–58, 2015.

[47] Callan Bright, Lyndon While, Tim French, and Mark Reynolds. Using market-based optimisation to solve the dynamic vehicle routing problem. In *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*, pages 1–8. IEEE, 2017.

[48] Zhiyong Wang and Sisi Zlatanova. Multi-agent based path planning for first responders among moving obstacles. *Computers, Environment and Urban Systems*, 56:48–58, 2016.

[49] Lingzhi Luo, Nilanjan Chakraborty, and Katia Sycara. Provably-good distributed algorithm for constrained multi-robot task assignment for grouped tasks. *IEEE Transactions on Robotics*, 31(1):19–30, 2015.

[50] Lujia Wang, Ming Liu, and Max Q-H Meng. A hierarchical auction-based mechanism for real-time resource allocation in cloud robotic systems. *IEEE Transactions on Cybernetics*, 47(2):473–484, 2017.

[51] Said Salhi, Arif Imran, and Niaz A Wassan. The multi-depot vehicle routing problem with heterogeneous vehicle fleet: Formulation and a variable neighbor-

hood search implementation. *Computers & Operations Research*, 52:315–325, 2014.

[52] Oscar Dominguez, Angel A Juan, Barry Barrios, Javier Faulin, and Alba Agustin. Using biased randomization for solving the two-dimensional loading vehicle routing problem with heterogeneous fleet. *Annals of Operations Research*, 236(2):383–404, 2016.

[53] Mustafa Avci and Seyda Topaloglu. A hybrid metaheuristic algorithm for heterogeneous vehicle routing problem with simultaneous pickup and delivery. *Expert Systems with Applications*, 53:160–171, 2016.

[54] David SW Lai, Ozgun Caliskan Demirag, and Janny MY Leung. A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph. *Transportation Research Part E: Logistics and Transportation Review*, 86:32–52, 2016.

[55] Baozhen Yao, Bin Yu, Ping Hu, Junjie Gao, and Mingheng Zhang. An improved particle swarm optimization for carton heterogeneous vehicle routing problem with a collection depot. *Annals of Operations Research*, 242(2):303–320, 2016.

[56] Jan Čurn, Dan Marinescu, Niall O'Hara, and Vinny Cahill. Data incest in cooperative localisation with the common past-invariant ensemble Kalman filter. In *16th International Conference on Information Fusion (FUSION)*, pages 68–76. IEEE, 2013.

[57] Zirui Xing and Yuanqing Xia. Comparison of centralised scaled unscented kalman filter and extended Kalman filter for multisensor data fusion architectures. *IET Signal Processing*, 10(4):359–365, 2016.

[58] Simon J Julier and Jeffrey K Uhlmann. A non-divergent estimation algorithm in the presence of unknown correlations. In *American Control Conference, 1997. Proceedings of the 1997*, volume 4, pages 2369–2373. IEEE, 1997.

[59] Hao Li, Fawzi Nashashibi, and Ming Yang. Split covariance intersection filter: Theory and its application to vehicle localization. *IEEE Transactions on Intelligent Transportation Systems*, 14(4):1860–1871, 2013.

[60] Leigang Wang, Tao Zhang, and Feifei Gao. Distributed cooperative localization with lower communication path requirements. *Robotics and Autonomous Systems*, 79:26–39, 2016.

[61] Ryo Kurazume, Souichiro Oshima, Shingo Nagakura, Yongjin Jeong, and Yumi Iwashita. Automatic large-scale three dimensional modeling using cooperative multiple robots. *Computer Vision and Image Understanding*, 157:25–42, 2017.

[62] Mohamed W Mehrez, George KI Mann, and Raymond G Gosine. An optimization based approach for relative localization and relative tracking control in multi-robot systems. *Journal of Intelligent & Robotic Systems*, 83:1–24, 2016.

[63] Benedetto Allotta, Riccardo Costanzi, Enrico Meli, L Pugi, Alessandro Ridolfi, and Gregorio Vettori. Cooperative localization of a team of AUVs by a tetrahedral configuration. *Robotics and Autonomous Systems*, 62(8):1228–1237, 2014.

[64] Thumeera R Wanasinghe, George KI Mann, and Raymond G Gosine. Distributed leader-assistive localization method for a heterogeneous multirobotic system. *IEEE Transactions on Automation Science and Engineering*, 12(3):795–809, 2015.

[65] Martin Fink, Guy Desaulniers, Markus Frey, Ferdinand Kiermaier, Rainer Kolisch, and François Soumis. Column generation for vehicle routing problems

with multiple synchronization constraints. *European Journal of Operational Research*, 272:699–711, 2019.

[66] Giovanni D'Urso, Stephen L Smith, Ramgopal Mettu, Timo Oksanen, and Robert Fitch. Multi-vehicle refill scheduling with queueing. *Computers and Electronics in Agriculture*, 144:44–57, 2018.

[67] Niels Agatz, Paul Bouman, and Marie Schmidt. Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, 52:739–1034, 2018.

[68] Yuan Yan and Yasamin Mostofi. Robotic router formation in realistic communication environments. *IEEE Transactions on Robotics*, 28(4):810–827, 2012.

[69] Yuanteng Pei, Matt W Mutka, and Ning Xi. Connectivity and bandwidth-aware real-time exploration in mobile robot networks. *Wireless Communications and Mobile Computing*, 13(9):847–863, 2013.

[70] Jacopo Banfi, Alberto Quattrini Li, Nicola Basilico, and Francesco Amigoni. Communication-constrained multirobot exploration: Short taxonomy and comparative results. In *Proceedings of the IROS Workshop on On-line Decision-making in Multi-robot Coordination (DEMUR2015)*, pages 1–8, 2015.

[71] James Stephan, Jonathan Fink, Vijay Kumar, and Alejandro Ribeiro. Concurrent control of mobility and communication in multirobot systems. *IEEE Transactions on Robotics*, 33(5):1248–1254, 2017.

[72] Yiannis Kantaros and Michael M Zavlanos. Global planning for multi-robot communication networks in complex environments. *IEEE Transactions on Robotics*, 32(5):1045–1061, 2016.

[73] Yun Wang and Cheng Hu. Moving as a whole: multirobot traveling problem constrained by connectivity. *Turkish Journal of Electrical Engineering & Computer Sciences*, 23(3):769–788, 2015.

[74] Guilherme Dhein, Alberto Francisco Kummer Neto, and Olinto César Bassi de Araújo. The multiple traveling salesman problem with backup coverage. *Electronic Notes in Discrete Mathematics*, 66:135–142, 2018.

Chapter 3

Fast Task Allocation for Heterogeneous Robots

This chapter is focussed on improving a technique that produces very fast task allocations, typically performed in real-time. These techniques are known as sequential single-item auctions, which allocate tasks using auctions. While designed for homogeneous robots, they are commonly applied to heterogeneous systems. It is identified that the standard auction rules can produce poor allocations when robots are heterogeneous. Several other bidding and auction rules are introduced, and their performance under a number of differing levels of heterogeneity, task distances, and system objectives are analysed. The new auction rules illustrate consistently improved performance for heterogeneous systems, while performing no worse for homogeneous systems. The applicability of this technique when robots have partial knowledge of the environment and partial communication with other robots is explored. The technique in these conditions is guaranteed to avoid deadlock provided robots never overestimate the other robots' capabilities.

Statement of Authorship

Paper Title:	Sequential Single-Item Auction Improvements for Heterogeneous Multi-Robot Routing
Status:	Accepted on 25 Feb 2019
Details:	Published in *Robotics & Autonomous Systems*, vol 115, pp 130-142, 2019

Principal Author

Name:	Nick Sullivan
Contribution Details:	Performed literature review on algorithms for allocating tasks to robots, separating them by computation time and highlighting strengths and weaknesses regarding speed, quality of solutions, performance guarantees, and applicability to non-linear problems. Contacted researchers who have written algorithms in this space and implemented their algorithms in code. Tested and observed these algorithms, discovering that auction-based algorithms operate poorly when robots are heterogeneous. Developed new auction algorithms that consider heterogeneity, and performed relevant mathematical proof. Thought up tests to illustrate performance and sensitivity of these algorithms in a variety of conditions, then wrote the code for these tests. Parsed and analysed results. Prepared the manuscript and generated all figures.
Contribution Percentage (%):	80

Signature: Date: 17 Mar, 2019

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

1. the candidate's stated contribution to the publication is accurate (as detailed above);
2. permission is granted for the candidate to include the publication in the thesis; and
3. the sum of all co-author contributions is equal to 100% less the candidates stated contribution.

Name:	Steven Grainger
Contribution Details:	Guided research direction. Supervised work development. Helped generate ideas for tests and edited manuscript.
Signature:	Date: 15 Mar, 2019

Name:	Ben Cazzolato
Contribution Details:	Guided research direction. Supervised work development. Helped generate ideas for tests and edited manuscript.
Signature:	Date: 13 Mar, 2019

Robotics and Autonomous Systems 115 (2019) 130–142

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Sequential single-item auction improvements for heterogeneous multi-robot routing

Nick Sullivan *, Steven Grainger, Ben Cazzolato

The University of Adelaide, South Australia 5005, Australia

HIGHLIGHTS

- Limitations with traditional heterogeneous task auctioning are shown.
- New auction bidding and resolution algorithms are explored.
- Consistent and significant improvements (up to 20%) can be made.
- A new auction process is introduced for heterogeneous robots with partial knowledge.

ARTICLE INFO

Article history:
Received 16 August 2018
Received in revised form 10 February 2019
Accepted 25 February 2019
Available online 28 February 2019

Keywords:
Multi-robot
Path planning
Routing
Sequential auction

ABSTRACT

We introduce new auction bidding and resolution algorithms to improve multi-robot sequential single-item auctions for heterogeneous systems. We consider two objectives, minimising the energy usage and time required to complete all tasks. Sequential single-item auctions are computationally inexpensive while producing efficient task allocations for homogeneous robots, but produce less efficient allocations for heterogeneous robots. Our algorithms provide consistent and significant (up to 20%) improvements for both objectives for a number of scenarios relative to the standard auction process, as tested in MATLAB simulations. Interestingly, our algorithms produce faster task completion even in homogeneous systems. We also introduce a new algorithm for sequential single-item auctions when robots have partial knowledge of their environment. We illustrate its improved performance and analyse its sensitivity, showing that precise tuning is not essential for faster and more efficient task completion. These improvements can reduce energy usage and task completion times for both indoor and outdoor robots in a variety of fields.

1. Introduction

The problem of Multi-Robot Task Allocation (MRTA) is fundamental in multi-robot systems. Given a set of tasks to be completed and a set of robots to complete them, it is desirable for the tasks to be allocated to the robots such that they are completed according to a given objective. Common objectives are to minimise energy usage (minimise the sum of all robot movement, *MiniSum*), or to minimise time (minimise the maximum robot movement, *MiniMax*). This problem exists in many fields, such as wireless sensor networks [1], assembly [2], cleaning [3], healthcare [4], and transportation [5].

To describe the multi-robot system using a formal taxonomy [6], we address systems with robots that complete one task at a time (single-task, ST), which plan for the future (time-extended assignment, TA), and complete tasks that require only one robot (single-robot, SR) with interrelated utilities. *Utility* is a robot's ability to complete a task at a given time. We address systems where tasks have physical locations, so a robot's utility for a task is dependent on the location of the previous task that was completed.

The number of possible allocations scale factorially, making brute-force search unusable for even a moderate number of tasks and robots. The techniques that exist in literature trade-off between solution quality and processing time. In this research area, a solution refers to a valid set of allocations such that all tasks are allocated and all robots can feasibly complete them.

Exact solutions, i.e. optimal task allocations for a given objective, are found by removing non-optimal solutions from consideration using techniques such as branch and cut [7]. The remaining solutions are then searched through to find the optimal task allocation. While optimal, these approaches take a long time to complete at large scale and cannot be applied to non-linear objectives. In particular, it cannot be used to find the allocation that results in the fastest completion of tasks. Solving this objective optimally remains an open problem.

* Corresponding author.
E-mail address: nicholas.sullivan@adelaide.edu.au (N. Sullivan).

https://doi.org/10.1016/j.robot.2019.02.016
0921-8890/

The best known algorithms for this objective are metaheuristics. They are problem independent frameworks that incorporate problem-specific information to solve optimisation problems. They can be tuned to converge within certain time-limits. A large number of metaheuristics exist, and many have been used for task allocation, including Genetic Algorithms [8], Simulated Annealing [9], and Ant Colony Optimisation [10]. Metaheuristics are often seeded with solutions from faster algorithms to reduce processing time.

The fastest algorithms to allocate tasks are known as heuristics. A popular and efficient heuristic is the sequential single-item auction [11,4,12]. One task is allocated per auction round, and robots create new bids based on the tasks that they have received. Robots initially have no tasks allocated to them. They then bid on each task. The overall best bid is the winner, and a robot is allocated the task. For each remaining unallocated task, the robot calculates the cost of adding the task to its tour, and bids accordingly. Another auction round is held and the process repeats until all tasks are allocated. It has been empirically shown to outperform other auction methods such as round-robin, ordered single-item auction, and parallel single-item auction [13]. Sequential single-item auctions are also guaranteed to be no worse then twice optimal for the energy efficiency objective [14]. Because of their speed, sequential single-item auctions are commonly used in dynamic systems where tasks must be re-allocated during operation [15,16].

Robots in these systems may be heterogeneous. Heterogeneity may refer to robots that are differ structurally, such as different speeds, or differ functionally, where they have different task completion capabilities. We consider both structural and functional heterogeneity. The motivating example is a system where robots have a set of certain skills, such as driving, viewing, or picking up objects. Each robot can complete a subset of tasks, and some robots can complete certain task types faster than other robots. While heterogeneous systems are more complex than their homogeneous counterparts, there are advantages to using these systems. It is more cost effective to use robots with different skillsets, rather than outfitting every robot to have every skill. It can also be infeasible to create a single robot with every necessary skill, such as a nimble robot that carries enormous weights.

A robot's ability to complete certain task types is referred to as the robot's *expertise* for those task types. If a robot does not have the required skill to complete a task, its expertise is 0. Expertise may be binary, representing if the robot can or cannot complete the given task; or continuous, representing the time it takes to complete a task relative to a baseline robot. An integer-linear program that finds optimal allocations for the energy efficiency objective exists for systems with binary expertise [7]. They use fractional linear programming for ground and air vehicles in combination with a branch-and-cut algorithm that selects resulting integer solutions.

Non-optimal solutions to continuous expertise systems have been addressed previously in a healthcare facility scenario [4]. Multiple heterogeneous robots performed tasks in various healthcare facility rooms. Sequential single-item auctions were held to allocate tasks, with bid calculations that were scaled by the expertise of the robots. If a robot has an expertise of 0.1, its bid will be 10 times worse than robots with a perfect expertise of 1. These expertise values reflected skills needed in a healthcare environment, including navigation, vision, speech, and cleaning. There has been research on task allocation for tightly-coupled multi-robot tasks [17]. Heterogeneous robots must form coalitions to complete tasks. They take into account robot expertise to form near-optimal coalitions using a sequential auction algorithm. Auction-based methods have also been developed for task allocation with reconfigurable teams [18]. Binary heterogeneity has also been considered for swarms using auction-based methods [19].

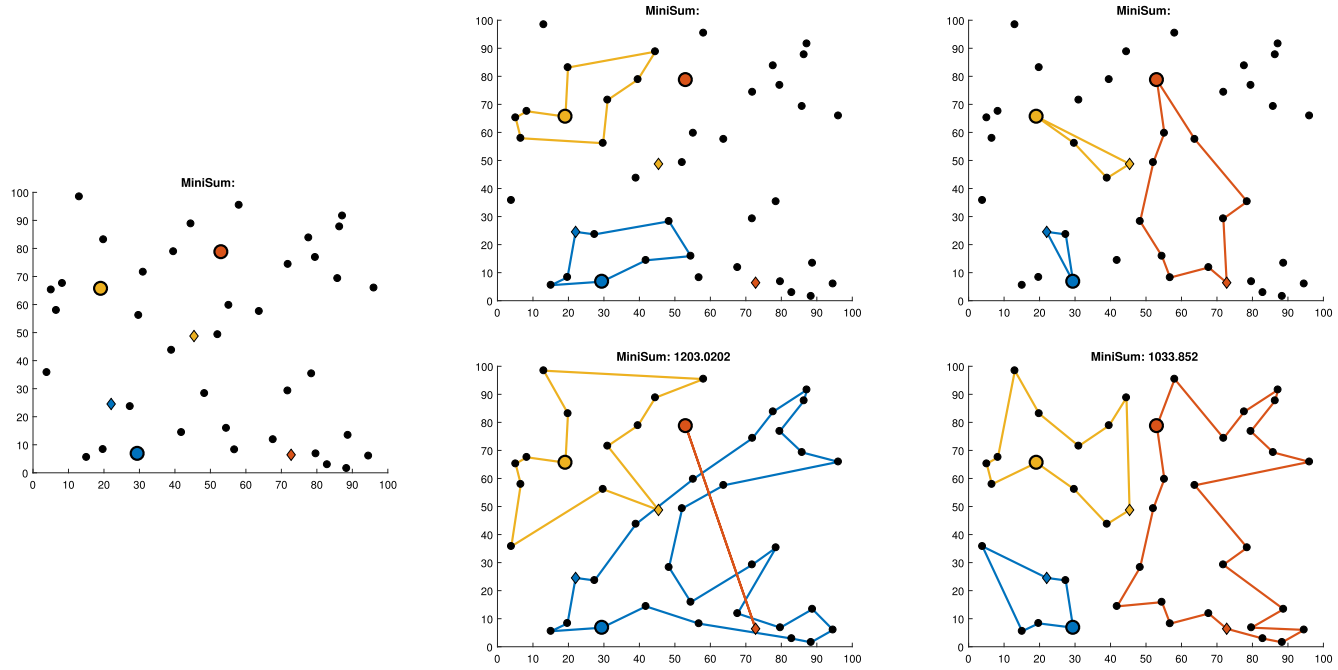
Robots have limited vision and communication. In this aspect, heterogeneous systems produces new challenges which do not occur in homogeneous systems. Consider such a system where robots have limited vision range and can only communicate to nearby robots. If these robots are functionally homogeneous, the best solution is simply for robots to complete the tasks that they are aware of, and search for more when they are complete. If they are heterogeneous, however, robots that are ill-equipped for nearby tasks may better meet the system objective by leaving their area to search and complete other tasks.

Robot exploration has been addressed using heuristics. In [20], robots were to explore rooms as efficiently as possible. Opening rooms required two robots, so robots had to decide between exploring opened rooms and opening up new rooms. The authors develop a number of heuristics to determine if a given robot should explore or open up a new area, and if the latter, which other robot it should open it up with. These heuristics show improved searching even when robots have limited communication and environment knowledge. Another technique to address whether robots should roam or stay is through the use of robot behaviours. Robots are allocated to tasks based on their local observations and perceptions. A behavioural task allocation architecture known as ALLIANCE [21] was recently implemented in ROS [22]. Robots in ALLIANCE make use of observations of task progression to determine their actions. When tasks are not completed satisfactorily, impatience parameters increase, eventually causing robots to take over from robots that are not performing their task well.

We propose new bidding rules and auction resolution algorithms to improve the performance of sequential single-item auctions. While the commonly used algorithms work well for homogeneous systems, it is possible for them to produce poor results in heterogeneous systems. A simple example of this is shown in Fig. 1. Sequential single-item auctions allocate tasks near the robots first to procedurally generate good paths for the homogeneous system. This hill-climbing can cause robots to build a path in one direction, but later be required to complete a task in a different direction, as seen in the middle column. Inspired by game theory, our algorithms change the order that tasks are allocated in. We first allocate tasks that have low levels of competition, in order to avoid path-building in the wrong direction.

In addition to the sequential single-item auction improvements, we propose a new algorithm that uses sequential auctions to decide whether robots should complete tasks they are aware of or search for tasks they are more suited for. An internal auction mechanism is used by each robot to predict if an unknown robot is more suitable for a task, and if so, it will ignore that task. This algorithm acts as an extension to sequential single-item auctions. We then experimentally analyse sensitivity and prove that deadlock (where all robots think that another will complete a task) cannot occur.

Following the introduction and literature review in Section 1, we define the problem and objectives in Section 2. The sequential single-item algorithms are specified in Section 3, which describe the bidding and auction resolution process. The simulation is described in Section 5. Section 6 outlines the experiments, including the four types of heterogeneous and homogeneous scenarios that are tested. Vision and communication constraints are applied in Section 4, which introduces and analyses our partial knowledge algorithm. The results of these experiments are found in Sections 7 and 8, along with worst-case and sensitivity analysis. The limitations of our algorithms are discussed in Section 9, followed by a conclusion in Section 10.



(a) Initial problem. (b) Allocation by Lowest Bid (LB). (c) Allocation by Fewest Bids (FB).

Fig. 1. An example of how tasks are allocated using the standard Lowest Bid (LB), versus the introduced Fewest Bids (FB). The robots (large circles) must form paths to complete all tasks (black dots), while minimising total energy usage (MiniSum objective). There are also tasks that can only be completed by particular robots (diamonds). The top row shows the allocation after 15 iterations. LB does not prioritise early inclusion of these special tasks. FB allocates the special tasks first, resulting in lower energy usage.

2. Problem definition

We first define terms used to describe multi-robot task allocation. Consider a heterogeneous set of robots with ID's $R = \{1, 2, \ldots, n\}$ and initial positions (depots) $D = \{d_1, d_2, \ldots, d_n\}$, and a set of tasks, $T = \{t_1, t_2, \ldots, t_m\}$. We form a graph with vertices $V = D \cup T$, and edges E consisting of a set of edges joining any two vertices in V. The cost function for a robot $r \in R$ to traverse an edge $e \in E$ is c_e^r. The cost includes the time taken to travel to the task, as well as the time taken to complete the task once the robot has arrived. A binary vector y_i^r specifies whether or not robot r completes task i. A binary vector x_e^r specifies whether or not robot r traverses edge e. This is the variable we are solving for. A function $\delta()$ takes nodes as an input, and returns edges that are connected to those nodes. Specifically, $\delta(S) = \{(i, j) \in E : i \in S, j \notin S\}$ takes a set of vertices and returns all edges that connect vertices in the set with vertices that are not in the set. The special case, $\delta(i)$, takes a single vertex $i \in V$ and returns all edges it is connected to.

We focus on two team objectives, minimising energy usage and minimising task completion time [11].

MiniSum: Minimise the sum of robot path costs over all robots.

MiniMax: Minimise the maximum robot path cost over all robots.

The problem can be formalised as follows:

$$\text{minimise} \quad \sum_{r=1}^{n} \sum_{e \in E} c_e^r x_e^r \qquad \text{(MiniSum)} \qquad (1)$$

$$\text{minimise} \quad \max_{r=1}^{n} \sum_{e \in E} c_e^r x_e^r \qquad \text{(MiniMax)} \qquad (2)$$

Subject to the following constraints:

$$x_{\delta(i)}^r = 2y_i^r \qquad \forall i \in T, r \in R \qquad (3)$$

$$x_{\delta(S)}^r \geq 2y_i^r \qquad \forall i \in S, S \subseteq T, r \in R \qquad (4)$$

$$\sum_{r=1}^{n} y_i^r = 1 \qquad \forall i \in T \qquad (5)$$

$$x_e^r \in \{0, 1\} \qquad \forall e \in E_t, r \in R \qquad (6)$$

$$x_e^r \in \{0, 1, 2\} \qquad \forall e \in E_r, r \in R \qquad (7)$$

$$y_i^r \in \{0, 1\} \qquad \forall i \in T, r \in R \qquad (8)$$

Eqs. (1) and (2) specify two objective functions, which is to minimise the sum of all costs (energy efficiency objective) or to minimise the maximum cost (shortest time objective). We solve each objective separately. Eq. (3) ensures that exactly two connected edges are used for each vertex for the robot that completes that task (one for moving to the task, one for moving away from it). All other robots should not use edges connected to that vertex. Eq. (4) is known as a sub-tour elimination constraint. This prevents solutions involving robots teleporting between locations. This particular sub-tour elimination constraint forces all vertex subsets to be connected to the rest of the vertices. Eq. (5) ensures that each task is completed by one robot. Eq. (6) ensures that edges between tasks are used 0 or 1 times, because the optimal solution will never use the same edge more than once between tasks. This is a consequence of the triangle inequality property. Eq. (7) ensures that edges connected to a robot start position are used 0, 1, or 2 times. 0 for if the robot is not used. 1 for if the robot completes multiple tasks. 2 for if the robot completes a single task. Eq. (8) ensures that tasks are completed by a given robot 0 or 1 times.

An edge e can be referred to as the movement between two vertices (i, j). Some of the introduced algorithms require splitting the edge cost c_e^r into the travel cost $tc_{(i,j)}^r$ and the base cost bc_j. The travel cost is the Euclidean distance between the two vertices divided by robot speed. The base cost for a task j is denoted bc_j, which is independent of the robot completing it. In a system with

functionally heterogeneous robots, robots have different abilities to complete tasks. We refer to a robot's ability to complete a task as a robot's expertise for that task, $0 \leq \lambda_j^r \leq 1$. Base costs are normalised such that expertise values lie between 0 and 1.

The cost function we use for a robot r to traverse edge (i, j) is:

$$c^r_{(i,j)} = \begin{cases} tc^r_{(i,j)} + bc_j/\lambda^r_j, & j \in T \\ tc^r_{(i,j)}, & j \in D \end{cases} \tag{9}$$

To clarify Eq. (9), if a robot is returning to its start position, the edge cost is the time it takes the robot to move. If a robot is completing a task, the edge cost is the time it takes the robot to move to the task location plus the time it takes to complete that task.

The expertise value acts to inflate task completion time, representing robots requiring different amounts of time to perform a given task. For example, two cleaning robots may take the same amount of time to move to a room, but the larger robot with more arms will be able to clean it faster. Expertise can be 0, so certain tasks may have infinite costs to complete. This implies that robots may be unable to complete some tasks. We assume that for each task there is at least one robot with non-zero expertise. This cost function representation does not violate any previous assumptions, satisfies the triangle inequality, and applies to realistic robotic scenarios.

3. Multi-robot task allocation algorithms

In auction-based methods, robots may prompt an auction at any time to re-allocate tasks. This may be when new tasks are discovered, if a robot has stopped responding, or if any other dynamic change occurs. The prompting robot becomes the auctioneer and will hold multiple auction rounds. One task is allocated each auction round to the robot with the winning bid. Bids are independently calculated by the robots each auction round.

Every time robots are allocated a task, they extend their path. We make use of the insertion heuristic [23]. Given a (possibly empty) initial path, it will insert a task in the path at a position that creates the smallest extra cost. It is guaranteed to produce a path that is less than twice the cost of the optimal path when the triangle inequality holds and is on average 25% worse. It is used in sequential auctions due to its ability to incrementally build a path, rather than being re-built each auction round. While it is possible to use an optimal solver for this component, optimal solvers take much more time. This is not desirable as paths must be recalculated multiple times each auction.

In our implementation, we hold reverse auctions, where the lowest bidder wins. Robots bid task-completion costs, and the objective of the auction is to minimise the cost. This is functionally equivalent to a standard auction.

To participate in a sequential single-item auction, robots need two decision-making algorithms: A bidding algorithm to determine bids for each task, and an auction resolution algorithm to determine which task is allocated each round.

3.1. Bidding algorithms

Robots must independently calculate bids for each task in each auction round. The bidding algorithm determines what tasks they bid on, and how much they bid. In each round, there is a set of unallocated tasks, and robots have (initially empty) paths.

1. **MiniSum** [11] This algorithm was created as a means to achieve the MiniSum objective. Every task is bid on. The bidding amount is determined by the following:
   - Calculate the cost of completing currently allocated tasks
   - Calculate the cost of completing both the currently allocated tasks and the task being bid on
   - The bid is the *difference* between these two costs

   This bidding method will cause tasks to be allocated to robots that can complete them with the smallest extra cost.
2. **MiniMax** [11] This algorithm was created as a means to achieve the MiniMax objective. Every task is bid on. The bidding amount is determined by the following:
   - Calculate the cost of completing both the currently allocated tasks and the task being bid on
   - The bid is the cost of this new path

   This bidding method will cause tasks to be allocated somewhat evenly between robots.
3. **MiniAve** [11] This algorithm often produces solutions that are a balance between MiniSum and MiniMax. Every task is bid on. The bidding amount is determined by the following:
   - Calculate the cumulative cost of completing currently allocated tasks
   - Calculate the cumulative cost of completing both the currently allocated tasks and the task being bid on
   - The bid is the difference between these two costs

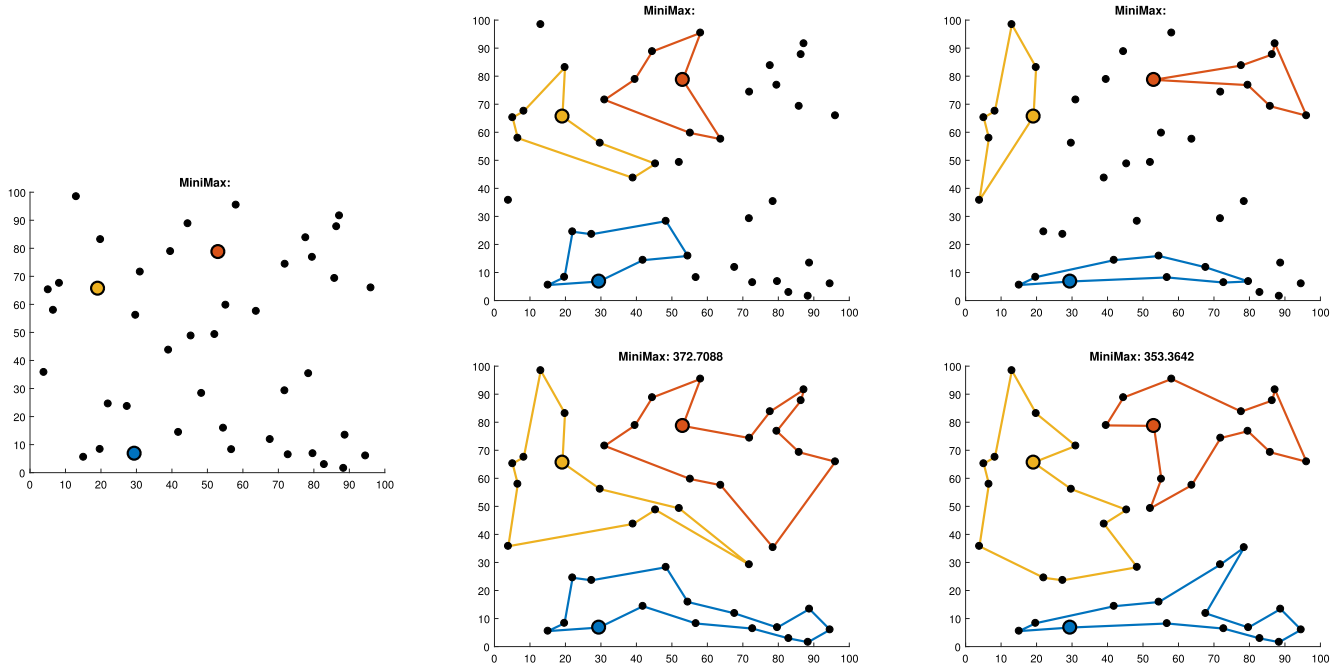
These algorithms are standard, and were developed for homogeneous systems, but can be immediately applied to heterogeneous ones. As mentioned previously, the cost function will be infinite for tasks that a robot is unable to complete. A bid of infinity is perfectly acceptable and is treated as the robot not being able to complete the task. When applying these bidding algorithms in heterogeneous systems, they do not necessarily produce solutions as close to optimal as in homogeneous ones. This is shown in Fig. 1. In this figure it can be seen that the auction resolution algorithms do not take into account functional heterogeneity, and while they perform well for homogeneous systems, they do not form good solutions for the heterogeneous example. To improve on these bidding algorithms, we introduce a sequential single-item auction algorithm that takes into account the functional heterogeneity of robots. This algorithm is built on top of the existing ones and use the same bidding values. They cause tasks to be allocated in a different order.

4. **RelativeExpertiseFirst (REF)** Robots do not bid on every unallocated task every round. Rather, they bid on the tasks that they are functionally well suited for relative to other robots. This algorithm only specifies which tasks the robot should bit on, the values are calculated using an underlying bidding algorithm, such as the traditional MiniMax or MiniAve. The pseudo-code for this algorithm is shown in Algorithm 1. Calculations of relative expertise and maximum relative expertise are defined by Eqs. (10) and (11) respectively.

$$\text{relative expertise}^r_t = \frac{\lambda^r_t}{\sum_{s \in R} \lambda^s_t} \tag{10}$$

$$\text{max relative expertise}^r = \max_{t \in U} \frac{\lambda^r_t}{\sum_{s \in R} \lambda^s_t} \tag{11}$$

   where robot r is bidding on a set of unallocated tasks $U \subseteq T$. This algorithm prioritises allocating tasks to robots that are functionally capable of completing them. It then bids on more common tasks afterwards, until all tasks are allocated. This algorithm is expected to produce larger travel times in order to obtain decreased task completion times.



(a) Initial problem. (b) Allocation by Lowest Bid (LB). (c) Allocation by Least Contested Bid (LCB).

Fig. 2. An example of how tasks are allocated using the standard Lowest Bid (LB), versus the introduced Least Contested Bid (LCB). The robots (large circles) must form paths to complete all tasks (black dots), while minimising task completion time (MiniMax objective). The top row shows the allocation after 20 iterations. LB causes paths to grow outwards from each robot. LCB causes paths to grow away from one another, resulting in faster task completion.

Algorithm 1 RelativeExpertiseFirst (REF)

// Robot $r \in R$ is bidding on unallocated tasks $t \in U$
// with expertise λ_t^r.
procedure REF BIDDING ALGORITHM
  // Calculate relative expertise.
  for all $t \in U$ **do**
    $\text{sumexp}_t = \sum_{s \in R} \lambda_t^s$ ▷ sum of expertises
    $\text{relexp}_t = \lambda_t^r / \text{sumexp}_t$ ▷ relative expertise
  end for
  $\text{relexp}_{max} = \max_{t \in U} \text{relexp}_t$ ▷ max relative expertise
  // Submit bids.
  for all $t \in U$ **do**
    if $\text{relexp}_t == \text{relexp}_{max}$ **then**
      $\text{bid}_t =$ (underlying bidding algorithm)
    else
      $\text{bid}_t = \infty$ ▷ do not bid for this task
    end if
  end for
end procedure

3.2. Auction resolution algorithms

The robot that prompts an auction becomes the auctioneer, and must be capable of determining which tasks to allocate to which robots based on their bids. It is standard for this to be the lowest bid:

1. **Lowest Bid (LB)** [11] This is the standard sequential auction resolution algorithm. The auctioneer takes the overall lowest bid and assigns that task to its bidder. As a consequence of this, robots only need to communicate their lowest bid to the auctioneer.

We offer new auction resolution algorithms:

2. **Biggest Bid Difference (BD)** The auctioneer allocates a task with the largest difference between the minimum and maximum bids. There may be one or more tasks that have the largest difference in bids, but only one task is allocated. The robot that submits the lowest bid to one of these tasks will be assigned a task. Unlike Lowest Bid, robots must submit bids on all tasks. The reasoning behind this algorithm is to prioritise early allocation of tasks that some robots are unable or ineffective at completing.
3. **Fewest Bids (FB)** The auctioneer allocates a task with the fewest bids. There may be one or more tasks that have the fewest bids, but only one task is allocated. The robot that submits the lowest bid to one of these tasks will be assigned a task. Robots must submit bids on all tasks that they can complete. The reasoning behind this algorithm is to prioritise early allocation of tasks that few robots are capable of completing. If robots are homogeneous, this is identical to Lowest Bid.
4. **Least Contested Bid (LCB)** Allocates the task which has the largest difference between the lowest bid and second lowest bid. In the event of a tie, the task with the overall lowest bid is assigned. The reasoning behind this algorithm is to prioritise early allocation of tasks that can be completed much better by one robot.

4. Partial knowledge

The algorithms in the previous Section assume robots have full knowledge of tasks and robots. This allows them to auction, allocate tasks, and form paths at the very beginning. They

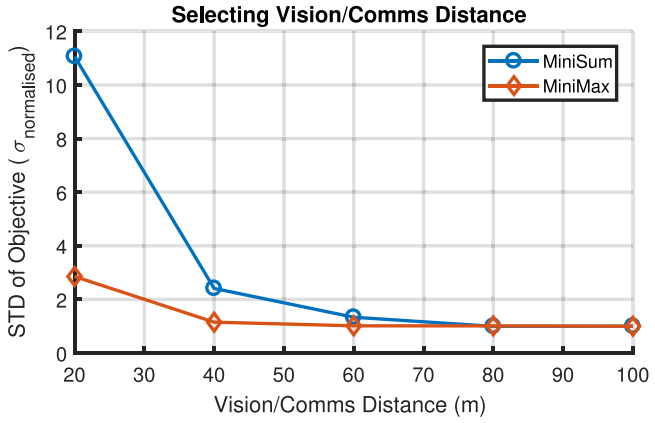


Fig. 3. Homogeneous robots with limited vision and communications completed location-based tasks in a 100 × 100 m area. The robots used random searching to find undiscovered tasks. The standard deviation of time of two system objectives (MiniSum, MiniMax) were calculated for each communications distance, then compared to the standard deviation for the same system where robots have full communication. This is used to determine at what range random searching dominates results.

then execute those paths without requiring re-allocation. This is representative of environments where communication is readily available, such as warehouses, healthcare facilities and farms.

It may be the case that robots do not have full connectivity, leaving them to operate under partial knowledge. In the next set of experiments, robots have finite visual and communication capabilities. They must re-allocate and form new paths based on the discovery of tasks. This is representative of environments where communication is not widely available and tasks may be unknown, such as environmental monitoring, transport, and defence. All robots are provided the total number of tasks in the environment, to prevent infinite roaming once all tasks are completed.

To extend on this, we perform experiments where robots are initially given partial global knowledge. They are aware of task types and robot expertise, but not the locations of tasks and robots. This provides robots with an interesting problem of whether they should complete tasks they are aware of, or to roam to find tasks they can complete at a lower cost. It is not unreasonable to assume that robots may have this information; robots often have some idea of the type of tasks they are looking for, and what other robots are in the system.

The vision and communication range was selected experimentally using the homogeneous scenario, as can be seen in Fig. 3. We did not want random search to dominate task completion times, because we are not testing search strategies. We plot the normalised standard deviation of path costs against vision and communication range. The normalised standard deviation is the standard deviation for a given vision and communications range divided by the standard deviation when the robots have infinite vision and communications range. An increase in standard deviation indicates that robots are roaming more. There is a sharp increase of standard deviation when reducing vision and communication range beneath 4 m. Therefore, we selected a range of 40 m for all tests involving robots with partial knowledge.

4.1. Bidding algorithm

We introduce a new bidding algorithm for robots operating with local knowledge. This algorithm is built on top of any standard bidding algorithm that assumes full knowledge, such as those listed in Section 3.

StayOrRoam (SOR) This algorithm is designed for robots operating with partial knowledge of tasks and robots. It uses information of heterogeneous expertise to choose whether a robot should stay and complete tasks it knows about, or roam to find tasks that it is more suited for. At the start of an auction (now referred to as an external auction), every participating robot performs their own internal auction process, where they estimate the bids from robots that are not within communication range. If a robot thinks that another robot can complete a task at a lower cost (i.e. a robot loses its own internal auction), it will not bid for that task in the external auction.

- Before bidding, the robot performs an internal parallel auction
- Each robot considers themselves as B units closer to the task than other robots with known capabilities but unknown position, where B is a value set by the user
- Known tasks are allocated using the bidding method in Eq. (12)
- For any tasks that the robot does not win in the internal auction, the robot will not bid on in the external auction
- Once all robots have completed their internal auctions, the external auction proceeds as a standard sequential auction

For each robot $r \in R$, there will be a subset of other robots $A \subset R$ that participate in the same auction, and a subset of robots $\bar{A} \subset R$ that do not participate in that auction. For each known task t, Robot r located at node i produces internal bids for each robot $s \in R$ according to the following rules:

$$bid_t^s = \begin{cases} \frac{bc_t}{\lambda_t^r}, & s = r \\ B + \frac{bc_t}{\lambda_t^s}, & s \in \bar{A} \\ \infty, & s \in A \end{cases} \tag{12}$$

In general terms, Eq. (12) splits internal bids into three categories: the robot holding the internal auction; robots participating in the external auction; and robots not participating in the external auction. The pseudo-code for StayOrRoam is shown in Algorithm 2.

In the StayOrRoam algorithm, each robot performs an internal auction, estimating the bids of other robots that are not within communication range. After the internal auctions, the robots perform an external auction (a standard sequential auction). If a robot loses its internal auction, it will not bid on that task in the external auction. If every robot loses its internal auction, then no robot will bid on that task in the external auction, resulting in the task never being completed. To avoid this, we require that at least one robot wins its internal auction. The robot with the highest expertise for a task will always win its internal bid, provided we select B to be greater than 0. Therefore, at least one robot will bid on each task, and deadlock will not occur. This then guarantees that every task will be allocated to a robot.

We select:

$$B = \max(C - tc_{(i,t)}^r, 0.0001) \tag{13}$$

Where C is an estimate of the distance between an unknown robot and the task being bid on. This information is unknown, so we instead use a constant ($C = 60$ for our experiments). In Section 8.5 we illustrate the sensitivity of C selection. Additionally, if all robots are within communication range, this algorithm is equivalent to its underlying bidding algorithm in Section 3 (e.g. MiniMax, MiniSum, MiniAve)

Algorithm 2 StayOrRoam (SOR)

// Robot $r \in R$ is bidding on unallocated tasks $t \in U$
// with expertise λ_t^r. There are other robots in the auction A
// and robots outside of communication range $\bar{A}$.
procedure SOR Bidding Algorithm
  // Internal auction (estimating other robots bids).
  for all $t \in U$ **do**
    for all $s \in R$ **do** ▷ r estimating bids from s
      if $s == r$ **then**
        $\text{ibid}_t^s = bc_t/\lambda_t^r$
      else if $s \in A$ **then** ▷ in the external auction
        $\text{ibid}_t^s = \infty$
      else ▷ not in the external auction
        C = estimated distance between s and t
        $tc_{(i,t)}^r$ = distance between r and t
        $B = \max(C - tc_{(i,t)}^r, 0)$
        $\text{ibid}_t^s = B + bc_t/\lambda_t^s$
      end if
    end for
    $\text{winner}_t = \underset{s \in R}{\text{argmin}}\, \text{ibid}_t^s$ ▷ winning robot for t
  end for
  // External auction (communicated with other robots).
  for all $t \in U$ **do**
    if $\text{winner}_t == r$ **then**
      bid_t = (underlying bidding algorithm)
    else
      $\text{bid}_t = \infty$ ▷ do not bid for this task
    end if
  end for
end procedure

The StayOrRoam algorithm will take slightly longer to process than sequential single-item auctions with full knowledge. For a system with N tasks and M robots, sequential single-item auctions with full knowledge involve each robot bidding on N tasks for N rounds, with the auctioneer resolving M bids for N rounds, resulting in a time order of $O(N^2 + M * N)$. The StayOrRoam algorithm involves extra internal auctions, where every round robots bid on behalf of robots they cannot communicate with, resulting in a total time order of $O(M * N^2 + M * N)$.

5. Simulation

Simulations were performed using a Multi-Robot Task Allocation (MRTA) system written in MATLAB at The University of Adelaide [24]. MATLAB was chosen for its deterministic results. Simulations are identical barring the change in task allocation algorithms.

The simulation flow can be seen in Fig. 4. The user specifies the robot details. This includes the names of the bidding algorithm and auction resolution algorithm discussed in Section 3. The simulator also requires robots expertise for the different skill types in the form of a vector of values between 0 (cannot complete) and 1 (can complete perfectly). Sensor information is also provided. In particular, the range and sweep angle that robots can see tasks and other robots, and the range that robots can communicate with one another.

Then the simulation enters the Run stage. Each loop consists of several segments. Firstly, robot sensors provide information of task and robot locations. This paper does not intend to address stochastic knowledge, so it is assumed that robots can perfectly identify robots and tasks, as well as completion status and position. If robots wish to communicate, a communication

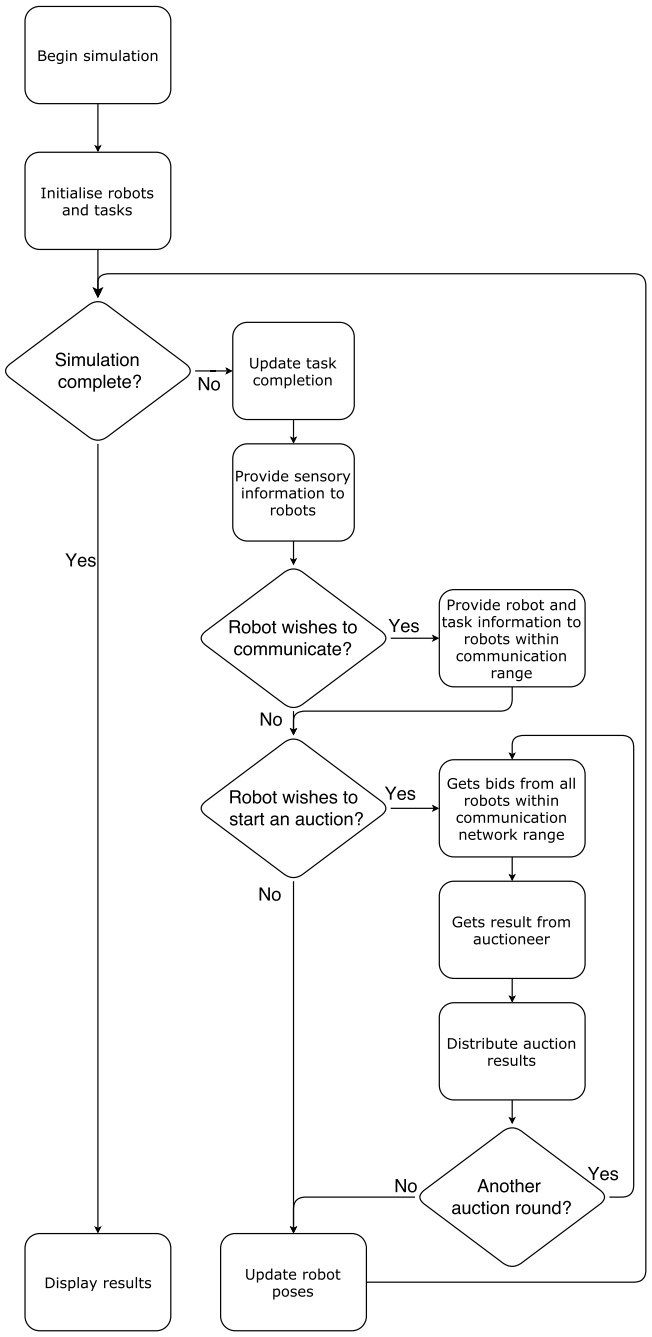


Fig. 4. The software diagram for the multi-robot task allocation simulation, written in MATLAB.

message is sent to nearby robots. These messages include all known robots, task information, as well as timestamps of when each piece of information was last updated. If robots wish to start an auction, all robots within the local communication network are informed. A communication network includes the auctioneer, robots within communication distance to the auctioneer, robots within communication distance to those robots, etc. All bids are communicated to the auctioneer, and resulting task allocations are communicated to all bidders. This process repeats until the auctioneer does not request another auction round. Finally, robots have their poses updated according to their desired pose and physics limitations. This loop repeats until the simulation is complete or times out.

5.1. Robots

Robots store information about tasks and other robots. They keep track of IDs, types, locations, completion statuses, and timestamps of when they learned new information. They update their information whenever they receive information from their sensors, or if they receive a message with newer information. As described previously, the robots choose what and when to communicate and if they want to start an auction. They raise communication and auction flags whenever a trigger is activated. Robots raise the communication flag for any of the following:

- They have not communicated in the last five seconds
  - This is a communication heartbeat, used to regularly distribute information across a network of robots [25, 26]
- They learn of a new task
- They learn of a new robot
- They learn that a task has been completed

Robots raise the auction flag for any of the following:

- They learn of a new task
- They learn that a task has been allocated to more than one robot
- There is an unallocated task that they want to complete

The intention of these rules is to propagate important information without using excessive bandwidth. If robots have knowledge of all robots and task locations, this results in a single auction. The allocations from this auction are then carried out without any further auctioning. If the robots have limited vision and communication range, however, new auctions will be held as a response to new information.

This paper does not specifically consider robots that become unable to complete tasks they have been allocated, but this can easily be accommodated as long as it can be detected e.g. the robot reports it or robots monitor each other. This would prompt a new auction so that the task will be re-allocated. Auction-based methods are well-suited for these systems where tasks can be re-allocated, as they are processed faster than alternative methods.

6. Experiments

Four robots are simulated in a 100 × 100 m area. In this area, 50 tasks exist that must be completed. The robots must complete all tasks and return to their starting positions. The position of the robots and tasks are randomised at the start of each sample run. Each algorithm is sampled 200 times to find an average and standard deviation. Four scenarios are compared:

1. Heterogeneous Specialists
   - There are five task types. One task type can be completed by all robots. The other four can only be completed by each robot respectively.
   - This is representative of multi-robot systems with particularly hard and easy tasks. The hard tasks require specialised robots, while the easy tasks can be completed by any robot.
2. Heterogeneous Random Expertise
   - There are five task types. All task types have uniformly random expertise values, from 0 to 1. There are 10 tasks for each task type.
   - This is representative of multi-robot systems with robots which are capable of all tasks but have differing levels of capability. It indicates the average performance of the algorithms under a wide variety of heterogeneity.
3. Heterogeneous Specialists with Overlapping Expertise
   - There are five task types. Robots 1 to 4 have perfect expertise for tasks 1 to 4 respectively. They also have no expertise for some tasks, and random expertise for the remaining tasks. This can be viewed in Table 1.
   - This is representative of multi-robot systems with robots that have specialised capabilities, but also some level of ability to complete other tasks as well. This can be seen in systems which aim to improve reliability and robustness through overlapping expertise. When a specialised robot breaks or becomes inundated with too many tasks, other robots are capable of assisting, despite having inferior capabilities.
4. Homogeneous
   - While we focus on heterogeneous scenarios, it is desirable for the algorithms to not produce inefficient or slow task allocations in the homogeneous case, as a heterogeneous system can result in a homogeneous scenario.
   - In this experiment, there is only one task type. All robots have perfect expertise for this task.

Table 1
Expertise table for scenario 3: specialists with overlapping expertise.

	Task 1	Task 2	Task 3	Task 4	Task 5
Robot 1	1	0	rand	rand	rand
Robot 2	0	1	0	rand	rand
Robot 3	rand	0	1	0	rand
Robot 4	rand	rand	0	1	rand

Table 2
Auction resolutions and bidding rules.

Algorithm	Type
Auction resolution	Lowest Bid (LB),[a] Biggest Difference (BD), Fewest Bids (FB), Least Contested Bid (LCB)
Bidding rules	MiniSum,[a] MiniMax,[a] MiniAve,[a] 3xRelativeExpertiseFirst (REFSum, REFMax, REFAve)

[a]Standard in sequential single-item auctions.

7. Results — full knowledge

Four scenarios are tested, as discussed in Section 6. A total of 24 combinations of auction types were tested for each scenario, as listed in Table 2. Robots have knowledge of all other robots and tasks, so a single auction process is performed at the start of each experiment.

Each experiment is repeated with different task base costs. To keep the results generalised, results are presented as a function of the portion of time spent travelling. The benchmark algorithms are MiniSum(LB) and MiniMax(LB) for the energy efficiency and completion time objectives respectively. Only the best-performing algorithms are presented graphically. Optimal solutions are presented for the homogeneous and specialist scenarios, but could not be calculated easily for the others, as this remains an open problem. A brief discussion of the auction types that are not graphed are presented here:

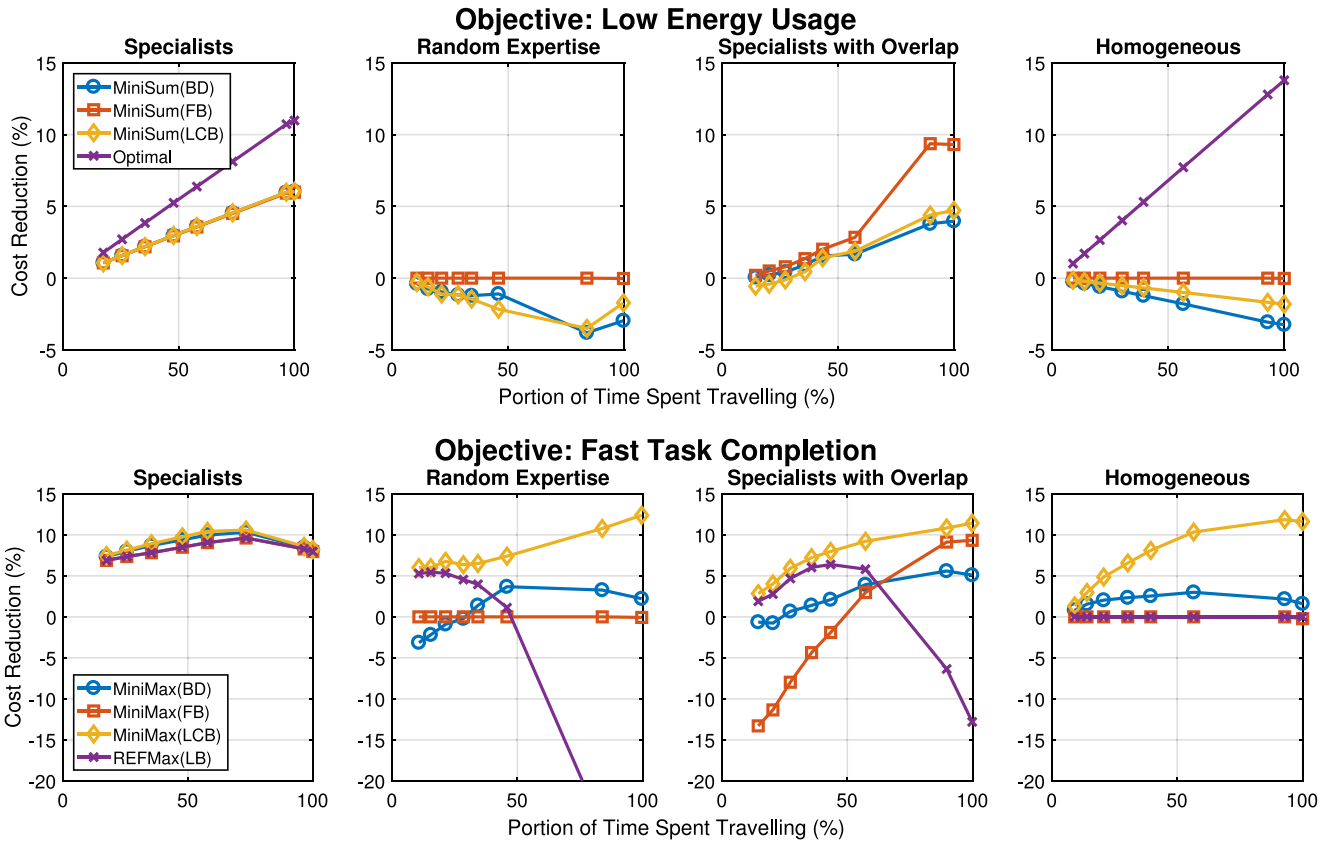


Fig. 5. Multiple robots completed location-based tasks using sequential single-item auctions to allocate tasks. The time taken to complete a task once arrived was altered. The experiments were done for two objectives: minimise the sum of all robot times (top), and minimise the maximum robot time (bottom). Four scenarios were compared: robots with different, non-overlapping abilities (far-left); robots with randomised abilities (middle-left); robots with different, overlapping abilities (middle-right); and robots with identical abilities (right). The performance of different sequential single-item auctions (listed in Table 2) are compared to the standard.

- MiniAve consistently produced results in between MiniMax and MiniSum, as expected.
- REFSum was consistently worse than MiniSum.
- REFAve was consistently worse than MiniAve.

7.1. Heterogeneous specialists

In the case of heterogeneous specialists, optimal solutions were able to be found for the energy efficiency objective using a integer-linear program [7]. The results for this scenario can be seen in the far-left column of Fig. 5. Note the different legends for the two objectives. All introduced algorithms (BD, FB, LCB, REF) produce similar and significant improvements for both objectives. In this scenario, tasks have big bid differences, a small number of bids, uncontested bids, and high relative expertise. This means that all algorithms produce very similar task allocation ordering.

It can also be seen that cost reduction tends to increase in magnitude as more time is spent travelling between tasks. This result may be unintuitive, one might expect that the usage of algorithms made to take into account differing task costs would perform best when base task costs are significant. The reasons against this are twofold: firstly, large base task costs produce large path costs, so even if the net cost reduction is large, the percent cost reduction will not necessarily be large as well. Secondly, when the portion of time spent travelling is low, the order in which tasks are allocated becomes less important.

7.2. Heterogeneous random expertise

The results for the heterogeneous random expertise scenario can be seen in the middle-left column of Fig. 5. Both MiniSum(BD) and MiniSum(LCB) produce worse paths than the benchmark MiniSum(LB), while MiniSum(FB) produces identical results. For the time objective, MiniMax(LCB) performs the best, producing consistently superior paths. REFMax(LB) produces similar paths when the majority of time is spent completing tasks rather than travelling, and produces significantly worse results when travelling time is large. MiniMax(BD) can be slightly better or slightly worse than the benchmark, while MiniMax(FB) produces identical results to the benchmark.

7.3. Heterogeneous specialists with overlapping expertise

The results for the heterogeneous specialists with overlapping expertise scenario can be seen in the middle-right column of Fig. 5. For the energy usage objective, MiniSum(FB) outperforms the others and shows consistent improvement, roughly twice as much as MiniSum(BD) and MiniSum(LCB). For the time objective, MiniMax(LCB) shows the most improvement. Once again, REFMax(LB) performs well when not much time is spent travelling. MiniMax(FB) shows the opposite, it performs well when almost all time is spent travelling. MiniMax(BD) shows small improvements.

7.4. Homogeneous

The results for the homogeneous scenario can be seen in the far-right column of Fig. 5. For the energy usage objective, MiniSum(BD) and MiniSum(LCB) performed worse than the benchmark MiniSum, while MiniSum(FB) produced identical results. For the time objective, MiniMax(LCB) produced significant improvement. This result was unexpected, but it can be seen that

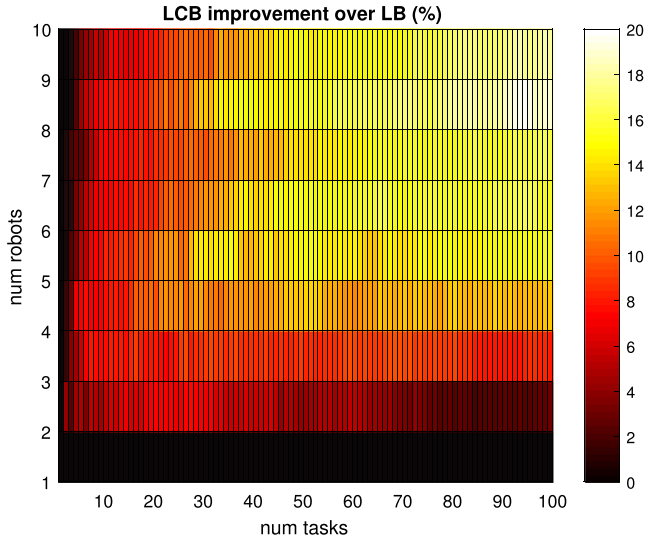
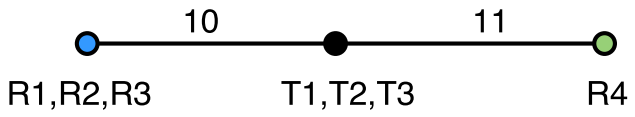


Fig. 6. A heat-map illustrating the how much faster tasks are completed using the Least Contested Bid (LCB) algorithm over the standard Lowest Bid (LB). Each rectangle is an average of 100 randomly generated scenarios in a 100 × 100 area.



	Task 1	Task 2	Task 3
Robot 1	1	0	0
Robot 2	0	1	0
Robot 3	0	0	1
Robot 4	1	1	1

Fig. 7. The scenario counter-proof for the upper limit of the Lowest Bid (LB) algorithm in heterogeneous scenarios. Four robots and three tasks are arranged above, with the expertise table specifying the tasks that each robot can complete.

changing the order in which tasks are allocated can offer significant improvement even in the homogeneous scenario. MiniMax(BD) also produced improvement, and MiniMax(FB) and REFMax(LB) produced identical results to the benchmark.

To further analyse the performance of MiniMax(LCB) in the homogeneous scenario, the number of robots and tasks were changed, and the results are averaged over 100 randomly generated tasks. The results are shown in Fig. 6. At low numbers of robots and tasks, MiniMax(LCB) offers very slight improvement over the standard algorithm. As the complexity of the problem increases, however, MiniMax(LCB) shows significant improvement.

In summary, these algorithms have been shown to offer significant and consistent cost reduction for sequential single-item auctions. MiniSum(FB) and MiniMax(LCB) outperformed other algorithms for energy efficiency objective and fastest time objective respectively. Examples of these algorithms are shown in Figs. 1 and 2.

7.5. Worst case performance

It is useful to identify the worst possible performance of the introduced algorithms, as the standard Lowest Bid algorithm provides guarantees on performance. In particular, MiniMax will be less than $2n$ times optimal cost, where n is the number of robots, and MiniSum will be less than 2 times the optimal cost. The proofs are detailed elsewhere [11], but the basic idea for the energy usage objective is:

- A minimum spanning forest is guaranteed to be less than or equal to an optimal solution.
- Incrementally adding cheapest edges will create a minimum spanning forest (Prim's algorithm).
- A robot increasing its path using a cheapest edge will increase its path cost by at most two times the cheapest edge (triangle inequality).
- Therefore incrementally adding cheapest edges will produce robot paths that are less than two times the optimal cost.
- The edges selected by Lowest Bid (LB) are the cheapest edges.
- Therefore LB will produce solutions within two times optimal.

This proof only applies to homogeneous case. We offer a simple counter example to show how a heterogeneous system can have a cost greater than two times optimal when using the standard auction resolution algorithm. Consider a system with four robots and three tasks, as shown in Fig. 7. The optimal solution is clearly for robot 4 to complete all three tasks, resulting in a total path length of 22. Allocating the lowest bids, however, would allocate robot 1 to task 1, robot 2 to task 2, and robot 3 to task 3. This results in three path lengths of 20, producing a sum of 60. $60/22 > 2$, and it is clear that we could increase the number of robots and tasks to create arbitrarily bad solutions.

The Fewest Bids (FB) algorithm has consistent worst-case performance to the standard Lowest Bid (LB). When applied to a homogeneous system, the algorithms act identically. When applied to heterogeneous systems, neither have an easily provable upper limit.

For the MiniMax case, however, the upper limit proof for Least Contested Bid does not apply it does not necessarily select the cheapest edge. This is not particularly significant as the original upper limit is $2n$, which is a loose upper bound.

8. Results partial knowledge

Robots with partial knowledge do not initially have information of the location of tasks and robots. They may discover tasks and robots when they are within range. Four scenarios are tested, as discussed in Section 6. The tested algorithms are listed in Table 3. Graphs are presented with varying base task costs. The benchmark algorithms are MiniAve(LB) and MiniMax(LB) for the energy usage and time objectives respectively. The reason for MiniAve(LB) is because it consistently outperformed MiniSum(LB).

With limited sensing and communication, the algorithms are 4% (when robots spend all their time travelling) and 26% (when robots spend most of their time completing tasks) less energy efficient than with full knowledge (MiniSum objective). They are between 2.4% (when robots spend all their time travelling) and 9.2% (when robots spend most of their time completing tasks) slower than with full knowledge (MiniMax objective).

A brief discussion of the auction types that are not graphed are presented here:

- MiniSum was consistently worse than MiniAve.
- REFSum was consistently worse than MiniSum.
- REFAve was consistently worse than MiniAve.

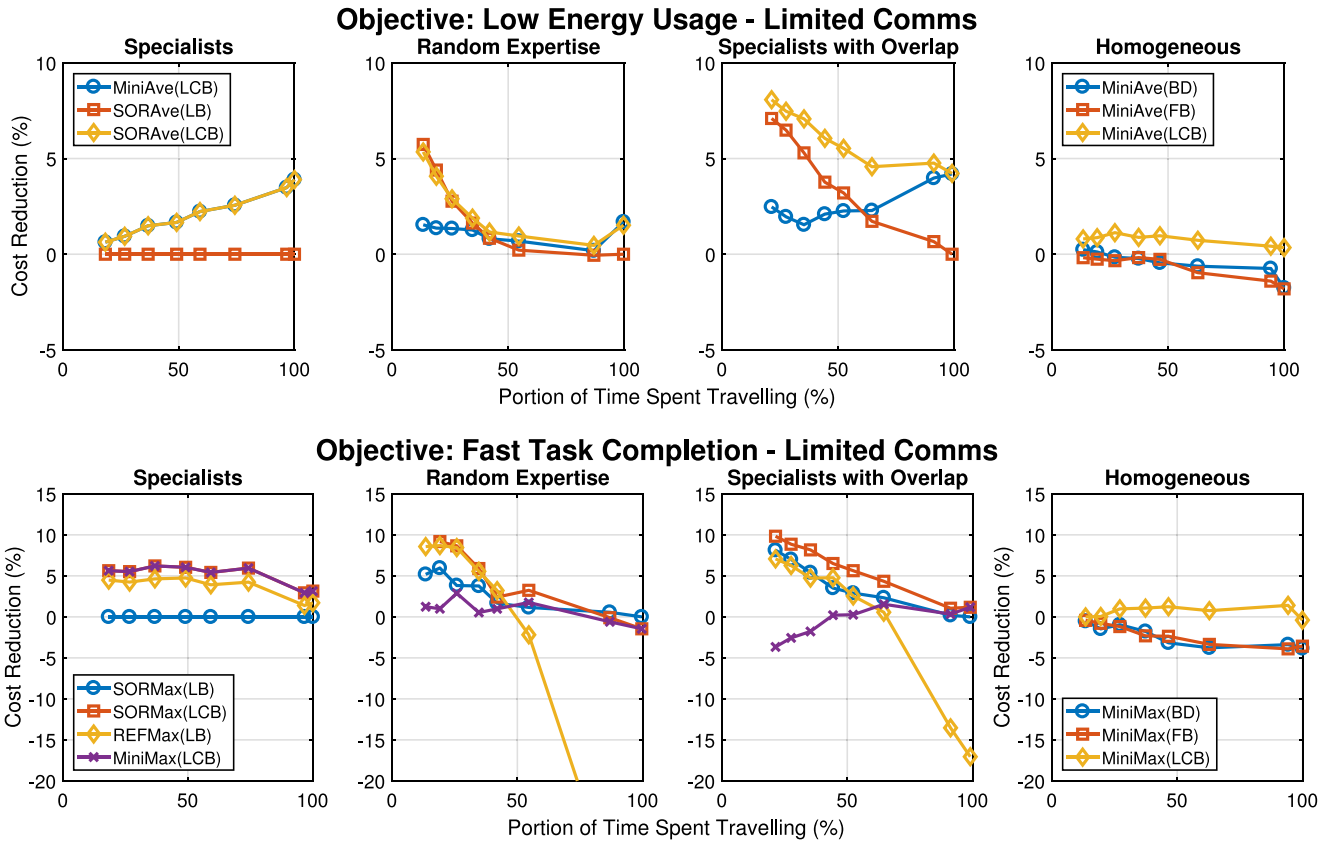


Fig. 8. Multiple robots completed location-based tasks using sequential single-item auctions to allocate tasks. The time taken to complete a task once arrived was altered. Robots had limited sensing and communication. The experiments were done for two objectives: minimise the sum of all robot times (top), and minimise the maximum robot times (bottom). Four scenarios were compared: robots with different, non-overlapping abilities (far-left); robots with randomised abilities (middle-left); robots with different, overlapping abilities (middle-right); and robots with identical abilities (right). The performance of different sequential single-item auctions (listed in Table 3) are compared to the standard.

Table 3
Auction resolutions and bidding rules for robots with partial knowledge.

Algorithm	Type
Auction resolution	Lowest Bid (LB),[a] Biggest Difference (BD), Fewest Bids (FB), Least Contested Bid (LCB)
Bidding rules	MiniSum,[a] MiniMax,[a] MiniAve,[a] 3xRelativeExpertiseFirst (REFSum, REFMax, REFAve), 3xStayOrRoam (SORSum, SORMax, SORAve)

[a]Standard method used in sequential single-item auctions.

8.1. Heterogeneous specialists

The results for this scenario can be seen in the far-left column of Fig. 8. In the specialist scenario, SORMax(LB) performs the same as the benchmark, and SORMax(LCB) performs the same as MiniMax(LCB). This is because all tasks can either be completed equally by all robots or only completed by one robot. It can be seen that using LCB produces the shortest paths for both objectives.

8.2. Heterogeneous random expertise

The results for this scenario can be seen in the middle-left column of Fig. 8. For the random expertise scenario, we can see that SORAve(LB) and SORAve(LCB) show improvement over the benchmark and over MiniAve(LCB). REFMax(LB) shows the same falloff seen earlier. SORMax(LCB) is equally good as REFMax(LB) at low travelling times, but does not have the significant cost increases at high travelling times. SORMax(LB) performs slightly worse than SORMax(LCB) but still better than the benchmark. MiniMax(LCB) shows slight improvement.

8.3. Heterogeneous specialists with overlapping expertise

The results for this scenario can be seen in the middle-right column of Fig. 8. This shows similar results to the previous scenario. StayOrRoam consistently reduces path costs for both objectives.

8.4. Homogeneous

The results for this scenario can be seen in the far-right column of Fig. 8. For the homogeneous objective, REF and SOR perform identically in the homogeneous case, as it depends entirely on the auction resolution algorithm for performance, hence the different legend. For the MiniMax objective, MiniMax(LCB) is slightly better than the benchmark, while the others are all consistently worse than standard. This result is the same for the MiniSum objective but to a much smaller extent.

In summary, StayOrRoam has been shown to offer significant and consistent cost reduction for sequential single-item auctions under partial knowledge conditions. SORAve(LCB) and SORMax(LCB) outperformed other algorithms for the MiniSum and MiniMax objective respectively.

It would be possible to manufacture an example for an arbitrarily large amount of improvement. For example, consider robots in rooms filled with tasks for which they have very low expertise, but incredibly high expertise for the tasks in the room next to them. Nevertheless, improvement was shown for realistic scenarios.

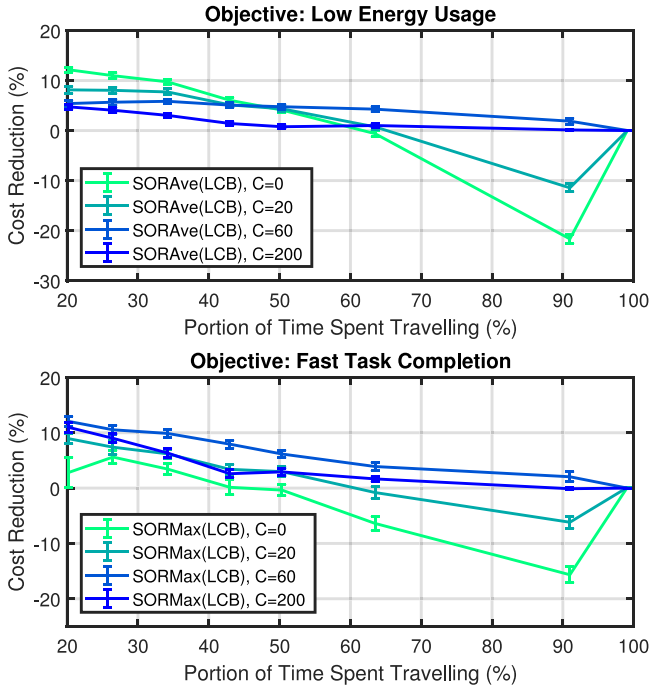


Fig. 9. The tuning value C for the StayOrRoam (SOR) algorithm is varied for multiple robots. For clarity, only four values are shown. Linear interpolation between these values are accurate estimates for the true results. Two system objectives are tested: minimise the sum of robot times (top), and minimise the maximum robot time (bottom).

8.5. Tuning sensitivity

The StayOrRoam algorithm was introduced in Section 4, including a tuning value C. Fig. 9 shows various values of C to show how sensitive this tuning value is, using the heterogeneous scenario: specialists with overlapping expertise. The lower limit for C is 0, which would cause robots to always roam unless they have the maximum expertise out of any robot. There is no upper limit for C, but as it approaches infinity the robots will never choose to roam.

For $C = 0$, it produces poor results when the majority of time is spent travelling. The robots always think that unknown robots are nearby, so they end up roaming far too much. It does, however, produce very good results for low levels of time spent travelling. For $C = 200$, robots always think that unknown robots are very far away, so they stay unless there are other robots that are significantly better at completing a task. The best value in this case was $C = 60$, as it had the best improvements without ever being worse than the base case. Other cases are not shown, but they closely match interpolation between the cases presented.

In short, if C is overestimated then there is less improvement over the standard algorithms, but the improvement is much more reliable.

9. Algorithm limitations

While the introduced auction resolution algorithms have been shown to produce lower cost paths than the benchmark algorithms, as seen in Fig. 5, they are not without their limitations. A common change for highly dynamic systems is to limit the path length [4]. The concept behind this is that if the system is changing unpredictably, then planning too far ahead is redundant. The introduced algorithms do not necessarily allocate nearby tasks first; if used with a limited path length, this can result in very bad solutions.

The StayOrRoam algorithm relies on accurate information about robot and task expertise. This may not be the case. If there is a known expertise uncertainty, deadlock can be prevented by ensuring B is greater than the maximum uncertainty. It may also be worthwhile to occasionally prompt robots to check that the knowledge of other robots is still valid.

10. Conclusion

We have introduced and analysed new bidding rules and auction resolution algorithms that show improved performance over standard sequential single-time auctions. For robots with global information, the use of auction resolution algorithms Fewest-Bids (FB) and Least-Contested-Bid (LCB) for the energy usage and fast completion objectives show significant and consistent improvements for heterogeneous scenarios. Interestingly, LCB produces reduced path costs even for homogeneous systems.

We also introduced StayOrRoam, a sequential single-item auction algorithm for systems where robots have partial task and robot knowledge. Robots decide whether to stay and complete the tasks known to them, or roam to find more suitable tasks. While it would be possible to create a scenario that exploits this for an arbitrary amount of benefit, we show that it outperforms the standard algorithms for realistic expertise values. We also show that accurate tuning of this algorithm is not required to obtain consistent improvements.

These algorithms have a positive impact on the performance of multi-robot systems requiring fast task allocation, or as seeds for metaheuristics that provide better solutions as the cost of processing time. Robots use less energy, allowing them to perform their tasks longer. Task completion time is reduced, improving overall performance of multi-robot systems. These algorithms are easy to implement, and do not require much adjustment from the traditional sequential single-item auction techniques in use today.

Acknowledgements

We would like to thank Dr. K. Sundaar (Los Alamos National Laboratory, New Mexico), for his help calculating optimal solutions to minimise the sum of all robot times.

This research was supported by the Phoenix High Performance Computing (HPC) service at the University of Adelaide, an Australian Government Research Training Program (RTP) Scholarship, and by the Commonwealth of Australia (represented by the Defence Science and Technology Group) through a Defence Science Partnerships agreement.

References

[1] Peng Yang, Randy A. Freeman, Geoffrey J. Gordon, Kevin M. Lynch, Siddhartha S. Srinivasa, Rahul Sukthankar, Decentralized estimation and control of graph connectivity for mobile sensor networks, Automatica 46 (2) (2010) 390–396.

[2] Ross A. Knepper, Todd Layton, John Romanishin, Daniela Rus, Ikeabot: An autonomous multi-robot coordinated furniture assembly system, in: Robotics and Automation, ICRA, 2013 IEEE International Conference on, IEEE, 2013, pp. 855–862.

[3] Emaad Mohamed H. Zahugi, Mohamed M. Shanta, T.V. Prasad, Design of multi-robot system for cleaning up marine oil spill, Int. J. Adv. Inf. Technol. 2 (4) (2012) 33.

[4] Gautham P. Das, Thomas M. McGinnity, Sonya A. Coleman, Laxmidhar Behera, A distributed task allocation algorithm for a multi-robot system in healthcare facilities, J. Intell. Robot. Syst. 80 (1) (2015) 33–58.

[5] Paolo Toth, Daniele Vigo, Vehicle Routing: Problems, Methods, and Applications, SIAM, 2014.

[6] G. Ayorkor Korsah, Anthony Stentz, M. Bernardine Dias, A comprehensive taxonomy for multi-robot task allocation, Int. J. Robot. Res. 32 (12) (2013) 1495–1512.

[7] Kaarthik Sundar, Sivakumar Rathinam, Algorithms for heterogeneous, multiple depot, multiple unmanned vehicle path planning problems, J. Intell. Robot. Syst. (2016) 1–14.
[8] Shuai Yuan, Bradley Skinner, Shoudong Huang, Dikai Liu, A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms, European J. Oper. Res. 228 (1) (2013) 72–82.
[9] Lucas P. Behnck, Dionisio Doering, Carlos Eduardo Pereira, Achim Rettberg, A modified simulated annealing algorithm for SUAVs path planning, IFAC-PapersOnLine 48 (10) (2015) 63–68.
[10] Xinye Chen, Ping Zhang, Guanglong Du, Fang Li, Ant colony optimization based memetic algorithm to solve bi-objective multiple traveling salesmen problem for multi-robot systems, IEEE Access (2018).
[11] Michail G. Lagoudakis, Evangelos Markakis, David Kempe, Pinar Keskinocak, Anton J. Kleywegt, Sven Koenig, Craig A. Tovey, Adam Meyerson, Sonal Jain, Auction-based multi-robot routing, in: Robotics: Science and Systems, vol. 5, 2005, pp. 343–350, Rome, Italy.
[12] Maitreyi Nanjanath, Maria Gini, Repeated auctions for robust task execution by a robot team, Robot. Auton. Syst. 58 (7) (2010) 900–909.
[13] Eric Schneider, Elizabeth I. Sklar, Simon Parsons, A. Tuna Özgelen, Auction-based task allocation for multi-robot teams in dynamic environments, in: Conference Towards Autonomous Robotic Systems, Springer, 2015, pp. 246–257.
[14] Sven Koenig, C. Tovey, M. Lagoudakis, V. Markakis, David Kempe, Pinar Keskinocak, A. Kleywegt, Adam Meyerson, Sonal Jain, The power of sequential single-item auctions for agent coordination, in: Proceedings of the National Conference on Artificial Intelligence, vol. 21, 2006, p. 1625, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.
[15] Eric Schneider, Elizabeth I. Sklar, Simon Parsons, A. Tuna Ozgelen, Auction-based task allocation for multi-robot teams in dynamic environments, in: Conference Towards Autonomous Robotic Systems, Springer, 2015, pp. 246–257.
[16] Callan Bright, Lyndon While, Tim French, Mark Reynolds, Using market-based optimisation to solve the dynamic vehicle routing problem, in: Computational Intelligence, SSCI, 2017 IEEE Symposium Series on, IEEE, 2017, pp. 1–8.
[17] Gautham P. Das, Thomas Martin McGinnity, Sonya A. Coleman, Simultaneous allocations of multiple tightly-coupled multi-robot tasks to coalitions of heterogeneous robots, in: Robotics and Biomimetics, ROBIO, 2014 IEEE International Conference on, IEEE, 2014, pp. 1198–1204.
[18] Zack Butler, Jacob Hays, Task allocation for reconfigurable teams, Robot. Auton. Syst. 68 (2015) 59–71.
[19] Muhammad Irfan, Adil Farooq, Auction-based task allocation scheme for dynamic coalition formations in limited robotic swarms with heterogeneous capabilities, in: Intelligent Systems Engineering, ICISE, 2016 International Conference on, IEEE, 2016, pp. 210–215.
[20] Torsten Andre, Christian Bettstetter, Collaboration in multi-robot exploration: To meet or not to meet? J. Intell. Robot. Syst. 82 (2) (2016) 325.
[21] Lynne E. Parker, Alliance: an architecture for fault tolerant, cooperative control of heterogeneous mobile robots, in: Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, vol. 2, IEEE, 1994, pp. 776–783.
[22] Wallace Pereira Neves dos Reis, Guilherme Sousa Bastos, Implementing and simulating an alliance-based multi-robot task allocation architecture using ROS, in: Latin American Robotics Symposium, Springer, 2016, pp. 210–227.
[23] Gregory Gutin, Abraham P. Punnen, The Traveling Salesman Problem and Its Variations, volume 12, Springer Science & Business Media, 2006, pp. 398–399.
[24] Nick Sullivan, Steven Grainger, Ben Cazzolato, Robust heterogeneous multi-robot routing for low-intelligence agents, in: Proceedings of Australasian Conference on Robotics and Automation, ACRA, ARAA, 2017.
[25] Joseph W. Durham, Ruggero Carli, Paolo Frasca, Francesco Bullo, Discrete partitioning and coverage control for gossiping robots, IEEE Trans. Robot. 28 (2) (2012) 364–378.
[26] Eduardo Feo Flushing, Luca M. Gambardella, Gianni A. Di Caro, On decentralized coordination for spatial task allocation and scheduling in heterogeneous teams, in: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 988–996.

Nick Sullivan received his bachelor degrees in mechatronics engineering and computer science from the University of Adelaide, Australia, in 2016. He is currently working toward the Ph.D. degree in robotics at the University of Adelaide. His research interests include multi-robot task allocation, unmanned ground vehicles, and machine learning.

Steven Grainger obtained his Ph.D. on the control of electric drives from Glasgow Caledonian University, Scotland and holds B.E. degrees in computing and electronic engineering. He is a lecturer in Control and Embedded Systems at the University of Adelaides School of Mechanical Engineering. His current research interests include nanopositioning systems and autonomous vehicles.

Ben Cazzolato received his B.E. in mechanical engineering at the University of Adelaide, Australia, in 1990. At the same university, he received his Ph.D. in the field of active control for sound transmission in 1998. He is currently a professor at the University of Adelaide, teaching and researching in the fields of dynamics and control. Current research interests include modelling of complex electro-mechanical systems, control of unstable vehicles, active control and nano-positioning.

Chapter 4

Task Allocation for Robots with Adaptive Heterogeneity

In this chapter, three techniques are developed for allocating tasks to multiple robots that can change their abilities. For example, they may pick up or put down a tool from the tool rack, or alter their physical structure. Adaptations to sequential single-item auctions and Genetic Algorithms (GAs) allow the problem to be solved directly. It is shown that it is important to consider whether or not the tools are restricted to given tasks, or can be used on multiple tasks as 'general-purpose' tools. This is important because general-purpose tools introduce a significant number of solutions that may make it harder to find desirable solutions. A transformation is introduced that converts the problem into a traditional Heterogeneous Multiple Travelling Salesman Problem (H-mTSP). The transformation is guaranteed to maintain the optimal solution under certain, easily calculable, general-purpose tool conditions. Finally, the algorithms are applied to a publicly available dataset for quantitative comparisons with other techniques.

Statement of Authorship

Paper Title:	Algorithms for Multi-Robot Routing with Adaptive Heterogeneity
Status:	Submitted for publication
Details:	Submitted to *Journal of Heuristics*, 9 July 2018

Principal Author

Name:	Nick Sullivan
Contribution Details:	Performed literature review on algorithms for allocating tasks to robots, separating them by computation time and highlighting strengths and weaknesses regarding speed, quality of solutions, performance guarantees, and applicability to non-linear problems. Discovered the lack of research on algorithms where robots can change their capabilities. Developed adaptations to auctions and genetic algorithm to solve these problems. Developed a transformation to convert the problem into a standard one, and performed the mathematical proof to show when this is optimal. Developed tests to illustrate performance and benchmark these algorithms in a variety of conditions, then wrote the code for these tests. Parsed and analysed results. Prepared the manuscript and generated all figures.
Contribution Percentage (%):	80
Signature:	Date: 17 Mar, 2019

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

1. the candidate's stated contribution to the publication is accurate (as detailed above);
2. permission is granted for the candidate to include the publication in the thesis; and
3. the sum of all co-author contributions is equal to 100% less the candidates stated contribution.

Name:	Steven Grainger
Contribution Details:	Guided research direction. Supervised work development. Helped generate ideas for tests and edited manuscript.
Signature:	Date: 15 Mar, 2019

Name:	Ben Cazzolato
Contribution Details:	Guided research direction. Supervised work development. Helped generate ideas for tests and edited manuscript.
Signature:	Date: 13 Mar, 2019

Algorithms for Multi-Robot Routing with Adaptive Heterogeneity

Nick Sullivan[1], Steven Grainger, Ben Cazzolato

The University of Adelaide, Australia

Abstract

This paper presents and compares methods for solving the multi-robot routing optimisation problem when robots can adjust their abilities through tool changes or modular reconfiguration. A transformation is used to convert the problem to a standard multi-robot routing problem, where state of the art algorithms can then be applied. An easy to calculate bound is provided to prove when optimality is preserved in this transformation. Adaptations to sequential auctions and genetic algorithms have been created to solve the problem without transformation, and are empirically compared to illustrate their performance. Finally, we benchmark these algorithms using a public dataset for future quantitative comparisons.

Keywords: Travelling Salesman Problem, multi-robot routing, task allocation

1. Introduction

In multi-robot systems, robots are used to complete tasks. In this paper, we consider tasks with spatial locations. To complete these tasks, robots must move to the task locations according to planned routes. It is trivial to generate routes that complete all tasks, but non-trivial to generate routes that complete all tasks while optimising system objectives such as speed or energy efficiency. The multi-robot routing problem, therefore, is interested in finding routes that complete all tasks while minimising (or maximising) system objectives within an acceptable time frame. This is fundamentally an optimisation problem.

Multi-robot routing is an example of the Multiple Travelling Salesmen Problem (mTSP) and the Multiple Depots Vehicle Routing Problem (MDVRP), which are NP-hard problems [1]. The difficulty lies in the large number of possible routes, where brute-force search is unusable for even moderately-sized problems. Therefore, a number of techniques exist that trade-off between solution quality and processing time.

Exact solutions, i.e. optimal solutions, can be found through techniques that reduce the number of searchable routes while ensuring the optimal route remains within the search. This allows optimal solutions to be found orders of magnitude faster than brute ofrce. The most common exact formulation is as an integer linear program, which has been applied to this problem for over 30 years [2]. While optimal, such approaches are unable to find solutions quickly at large scale and cannot be used for certain problems. It is possible to find the solution that minimises the total distance travelled (MiniSum), but finding the solution that minimises the longest path (MiniMax) remains an open problem. More recent research deals with additional constraints and multiple objectives [1].

A variety of heuristics have been used to find solutions very quickly at the cost of optimality. Rather than searching for solutions as a standard optimisation problem, these often operate by greedily making locally good choices in the hope that it will result in globally good solutions. Popular heuristics in robotics use market-based allocation, where robots are able to buy and sell tasks from one another. The more efficiently a robot can complete a task, the more it is willing to bid for that task. This leads to tasks being allocated to robots that can efficiently complete them. Different types of market-based allocation have been explored, including sequential single-item auctions [3] which allocates tasks one at a time to the robots with the lowest bids. This is fast and efficient relative to other heuristics [4, 5]. Combinatorial auctions [6] allocate combinations of tasks each round and can theoretically find optimal solutions. But due to a large number of possible combinations, it is not practical to auction every combination of tasks. After allocation, robot to robot task selling [7] can be performed to adjust to uncertainty, but does not scale well.

Metaheuristics are problem independent frameworks that incorporate problem-specific information to solve optimisation problems. They operate using two procedures, exploration and exploitation [8]. Exploitation applies local improvements, which are quick but limited by local optima. Exploration finds new, different solutions as a means to overcome these local optima. Metaheuristics attempt to balance the two procedures to operate quickly

*Corresponding author

Email address: nicholas.sullivan@adelaide.edu.au (Nick Sullivan)

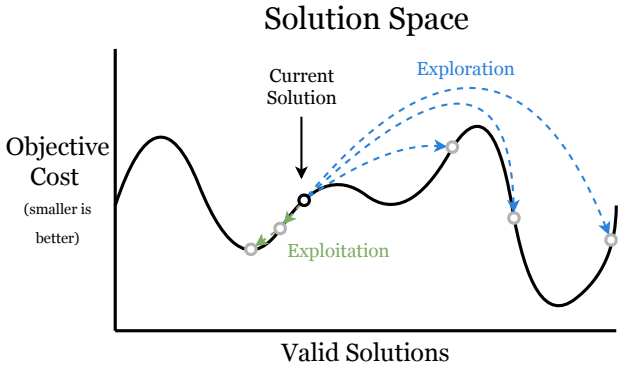


Figure 1: Metaheuristics use exploitation to improve upon solutions, and exploration to generate brand new solutions. These are used to efficiently find good solutions for optimisation problems that are difficult to solve within required time-frames.

and robustly. A graphical representation of this process can be seen in Figure 1. Metaheuristics have been used for a large number of optimisation problems, such as wind energy conversion [9], market pricing [10], and crowd evacuation [11]. Recently, there has been much research attention on applying these metahueristics to the multi-robot routing problem.

Thus far, no metaheuristic has been been accepted as the best for multi-robot routing. Genetic algorithms [12] are inspired by biological evolution. An initial population of solutions are randomly generated. New solutions are generated through breeding and mutation. The best solutions survive to the next generation and the process is repeated. After many generations, the population is likely to consist of good solutions. Genetic algorithms are one of the most popular metaherustics for multi-robot routing problems [13, 14, 15].

Other meta-heuristics include Simulated Annealing [16], where the searching jumps according to a heat value that starts hot and cools over time. This is one of the oldest and simplest metaheuristic, but has also recently been used for applications such as manufacturing layout design [17] and UAV inspections [18].

Wolf Pack Algorithm [19], is based on the pack behaviour of wolves in nature. Wolves that find good solutions search nearby solutions, and wolves that do not find good solutions head towards the wolves that do. This has been used for UAV routing [20] and transit network design [21].

Ant Colony Optimisation is a metaherustic inspired by ants [22]. Simulated ants produce pheromones and over time, the shortest paths will have the strongest pheromones. While ant colony optimisation has seen the most success in dynamic problems such as ad-hoc networks [23], they have also been applied to multi-robot routing [24, 25].

These meta-heuristics act as any-time algorithms, they can be stopped at any point and still provide a solution. In practice, they still require extensive tuning to converge upon good solutions shortly before the desired completion time. Nevertheless, they can provide good solutions within a custom amount of time.

Recently, the multi-robot routing problem has also been considered for heterogeneous systems, i.e., the Heterogeneous Multiple Travelling Salesmen Problem (H-mTSP) and the Heterogeneous Multiple Depots Vehicle Routing Problem (HMDVRP). Heterogeneous systems are prevalent in the real world, even for systems which are designed as homogeneous. A homogeneous fleet can become heterogeneous through state, such as a robot equipping a tool or using up its battery. It can also occur through hardware failure and wear and tear over time. A designer may even choose to create a heterogeneous system to improve efficiency. Robotic systems that are designed through evolution often result in specialisation, and usually outperform those that are evolved with homogeneous restrictions [26, 27]. Heterogeneous systems are more robust if the robots have a certain level of capability overlap, as this allows other robots to complete tasks under instances of high demand or robot failure. This concept is known as degeneracy [28].

Heterogeneity can refer to structural heterogeneity (e.g. a robot moves faster than another), or functional heterogeneity (e.g., a robot can complete a task that another robot cannot). There is plentiful research for systems with structural heterogeneity [29], but far less research on functional heterogeneity. We consider functional heterogeneity in this paper, where some robots do not have the capability to complete certain tasks. Heterogeneity can be reflected as binary, representing if the robot can or cannot complete a given task; or continuous, representing the extra time it takes to complete a task relative to a baseline robot.

An optimal solution exists for systems where robots have binary heterogeneity using fractional linear programming with a branch-and-cut algorithm [30]. This process, coupled with off-the-shelf optimisation software, finds potential solutions that meet relaxed non-integer constraints, and then tighten the constraints to form valid solutions. Partial solutions that cannot outperform the current best solution are cut to improve search speed.

Sequential single-item auctions have been used to allocate robots in a healthcare scenario, where the robots had continuous heterogeneity [31]. The heterogeneity reflected differing capabilities to perform tasks such as navigation, vision, speech and cleaning. No adaptations were made to the core sequential single-item auction algorithm, illustrating that it can be used 'as is' for heterogeneous systems. Our previous work has shown that alterations to the auction resolution process provide improved performance for heterogeneous systems [32], however this was limited to static heterogeneity. Ant-colony optimisation has been used for deliveries with a heterogeneous fleet, where they showed using heterogeneous vehicles is 9.2% more effective than homogeneous vehicles for China Post of Guangzhou [24].

One instance that has not been considered in depth

is the possibility of heterogeneity that can deliberately change throughout the task completion process. For example, when a robot picks up a hammer, it gains the ability to push in a nail. Tool manipulation was a large factor in the 2015 DARPA robotics challenge [33], where robots had to use handheld tools and fire hoses to successfully complete their mission. This type of problem also exists in tool-swaps for furniture assembly [34], cleaning and delivery [35], and cooking [36]. A common solution is to distribute tools at the start to match demand so that tool changing does not need to be considered [34]. This solution will clearly not work for cases where there are more tools than robots. This problem will need to be considered in order to create robots that are able to operate in human environments.

Heterogeneity can also be deliberately changed through modular reconfiguration. Robots are being designed that can detach limbs as needed in large facilities [37], and change between wheels and legs for robust space exploration [38]. A review paper on different types of modular robotics and their characteristics and applications is available [39]. Determining when these robots should change form is critical for efficient task completion.

The possibility of deliberately changing heterogeneity has not been considered for the general heterogeneous multi-robot routing case. The rest of the paper is outlined as follows.

In Section 2 we provide a definition that includes the option to change abilities. A transformation that converts the problem into a standard heterogeneous multi-robot routing problem is described in Section 3. Sections 4 and 5 describe the sequential auction and genetic algorithm adaptations respectively. Section 6 provides a worst-case analysis of the previous algorithms. These algorithms are empirically tested in experiments described in Section 7, with an analysis of experimental results in Section 8. Tabulated results for benchmarking are discussed in Section 9. The paper is concluded in Section 10.

2. Dynamic Heterogeneous Multi-Robot Routing Definition

We first define terms used to describe the system. Consider a set of robots with ID's $R = \{1, 2, \ldots, n\}$ and initial positions $D = \{d_1, d_2, \ldots, d_n\}$, and a set of task positions, $T = \{t_1, t_2, \ldots, t_m\}$.

A robot route, or path, of length l consists of l tasks that the robot is to visit and complete, $p^r = \{t_1, t_2, \ldots, t_l\}$. Solutions to the multi-robot routing problem require n routes (one for each robot) which result in all tasks being completed exactly once.

We form a graph with vertices $V = D \cup T$, and edges E joining any two vertices in V. The cost function to traverse an edge $e \in E$ is c_e. We wish for all tasks to be visited by at least one robot while minimising a system objective. We focus on two independent objectives [3]. The first objective, MiniSum, aims to minimise the sum of robot path costs over all robots. This is synonymous with completing tasks such that energy usage is minimised. The second objective, MiniMax, aims to minimise the maximum path cost. This is synonymous with completing all tasks in the fastest possible time.

To extend on this problem, we assume there are a set of tools with ID's $TO = \{to_1, to_2, \ldots, to_a\}$ which can be equipped by any robot. Each robot can equip at most one tool at a time, and we assume there are sufficient copies of tools. Tools are equipped by moving to a tool rack, which have positions $TR = \{tr_1, tr_2, \ldots, tr_b\}$. Each vertex $v \in V$ can only be visited if the robot has equipped a valid tool for that vertex. Robots must return their tool to a tool rack before returning home.

3. Transformation

At first glance, one might consider treating a tool swap as a task in itself. This makes some logical sense, to complete the task 'dig hole' one might first need to complete the task 'get shovel'. If one were to pursue this representation, the set of vertices would include tool racks; all edges between tasks that require different tools would be removed; edges between tool racks and other vertices would be added; and vertices would be able to be visited multiple times. This introduces two significant changes. Firstly, the graph is no longer complete, making many algorithms unable to be used, although some algorithms do exist for incomplete graphs [40]. Secondly, we would be unable to rule out a vertex once it has been visited. This produces an infinite number of valid solutions making many solution techniques inapplicable in their current form. For sequential single-item auctions, outlined in Section 4, there would be no guarantee that the auctions would ever finish. For genetic algorithms, outlined in Section 5, visiting vertices multiple times would cause the genome to grow. This creates an infinite number of poor solutions that can be searched, decreasing the likelihood of finding good solutions. For integer programming formulations, some of the constraints used are no longer applicable, resulting in a large (potentially infinite) increase in computation time.

Instead of treating tool swaps as tasks, we introduce a transformation that converts the problem into a standard multi-robot routing problem. We couple tool swaps with the tasks such that triangle inequality is maintained. It is inspired by a common transformation used to convert a graph without triangle inequality into one that does, using the FloydWarshall algorithm [41]. Triangle inequality is the property that for any three vertices $i, j, k \in V$,

$$c_{ij} < c_{ik} + c_{kj} \tag{1}$$

i.e., if a robot at vertex i wishes to go to vertex j, it is never quicker to detour to vertex k. If we have a TSP where this is not the case, we can check every combination of $i, j, k \in V$, and replace edge costs with the smallest costs between two vertices. In essence, the use of a lower-cost

detour is built into the cost. For tool swaps, we undergo a similar process.

For each edge $e \in E$ between vertices $i, j \in V$, we adjust the cost:

$$c_{ij} = \begin{cases} c_{ij}, & to_i = to_j \\ c_{ij} + swapcost, & to_i \neq to_j \end{cases} \quad (2)$$

The cost is altered if a different tool is required for a given task. This process is outlined in Algorithm 1. For each edge, the use of each tool rack is compared, and the one producing the shortest cost is used. Following this transformation, we have a standard (but non-Euclidean) multi-robot routing problem with finite edge costs. Theorem 1 shows that for a set of vertices (start locations, tasks, tool racks) that obey the triangle inequality (without yet considering tool constraints), the transformation will maintain the triangle inequality while applying tool constraints. When a robot traverses an edge that requires a tool change, it implies a tool swap. The cost for this tool swap is inherently included in the edge cost. To transform the multi-robot routing problem back into a tool-based multi-robot routing problem, we must record which edges were altered, and with what tool rack. For the reverse transformation, we follow each tour and replace each edge with the required tool swap. The reverse transformation is also outlined in Algorithm 1. Examples of a solution that has been transformed, solved, and transformed back are shown in Figures 2 and 3.

Algorithm 1 Tool Transformations

```
1: procedure ROUTING W/ TOOLS -> ROUTING
2:     for all (i, j) ∈ E do
3:         if to_i ≠ to_j then                    ▷ swap is required
4:             mincostswap ← inf
5:             mincostrack ← −1
6:             for all tr ∈ TR do                 ▷ select best rack
7:                 a ← c_{i,tr} + swapconst + c_{tr,j}
8:                 if a < mincostswap then
9:                     mincostswap ← a
10:                    mincostrack ← tr
11:            c_ij ← c_ij + mincostswap
12:            racksused(i, j) ← mincostrack
13:        else
14:            racksused(i, j) ← −1
15: procedure ROUTING -> ROUTING W/ TOOLS
16:    for all (i, j) ∈ path do
17:        rack = racksused(i, j)
18:        if rack ≠ −1 then
19:            path.insert(rack)
```

Theorem 1. *Consider a set of vertices consisting of tasks i, j, k and tool racks tr_x that obey the triangle inequality, i.e., for any vertices a, b, c, $c_{a,b} \leq c_{a,c} + c_{c,b}$.*

When tool requirements are included, triangle inequality is maintained if:

$$c_{i,tr_1} + c_{tr_1,j} \leq c_{i,k} + c_{k,tr_1} + c_{tr_1,j} \quad (3)$$
$$c_{i,tr_1} + c_{tr_1,j} \leq c_{i,tr_1} + c_{tr_1,k} + c_{k,j} \quad (4)$$

Through cancellation this becomes:

$$c_{i,tr_1} \leq c_{i,k} + c_{k,tr_1} \quad (5)$$
$$c_{tr_1,j} \leq c_{tr_1,k} + c_{k,j} \quad (6)$$

Which are both true given the triangle inequality of the vertices. If we consider multiple tool racks, Algorithm 1 shows that a different tool rack tr_2 will be used if and only if:

$$c_{i,tr_2} + c_{tr_2,j} \leq c_{i,tr_1} + c_{tr_1,j} \quad (7)$$

Hence, a different tool rack being used will decrease the left hand sides of Equations (3) and (4). Therefore triangle inequality is maintained when tool requirements are included.

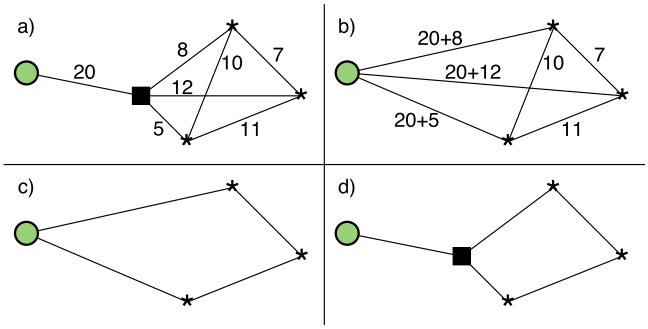


Figure 2: The process to transform between a multi-robot routing with tools problem and a standard multi-robot routing problem. a) A graph consisting of a robot (green circle), a tool rack (square), and tasks (*), with given edge costs. b) The tool swap costs are included into the relevant edges as outlined in Algorithm 1. c) Routes are generated. d) The problem is transformed back.

While the transformation assumes task-tool pairs, tool selection is not always that simple. It is possible that multiple tools may be able to complete the task with varying effectiveness. A hammer is an ideal tool for a nail, but any hard object with a flat surface could do the job. This means that there are multiple options for edge costs, depending on the tool that was used previously. The transformation is unable to be applied in this case without losing some solutions.

One could simply disregard inferior tools, e.g., do not ever use a flat rock to hammer the nail, only use the hammer. While this would allow the transformation to be used, hence allowing the use of current state-of-the-art algorithms, it disregards solutions that may be more effective, e.g., the hammer is very far away, and the robot is already holding a rock. In Section 6, we provide a bound that, if met, guarantees that the optimal solution will be maintained in the transformation for MiniSum and MiniMax. We also offer two adaptations to algorithms that

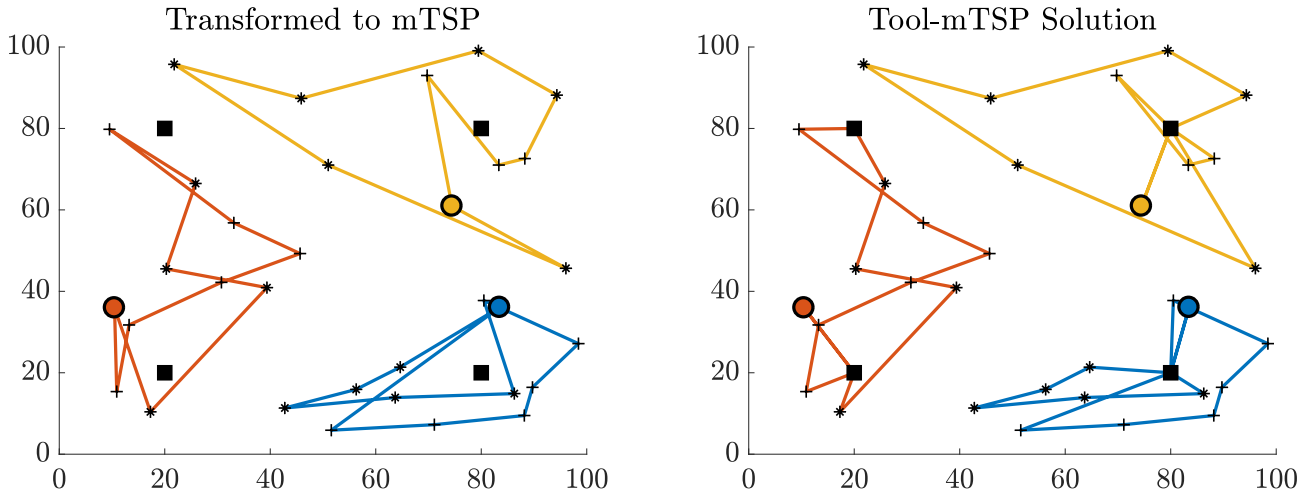


Figure 3: A multi-robot routing with tools problem has been transformed to a standard multi-robot routing problem and solved (left), then transformed back (right). The three robots (circles) form path plans to complete tasks (+ and *). They must make use of tools located at tool racks (squares), and return the tools before returning to their start location.

solve the multi-robot routing problem with tools problem directly. These can be found in Sections 4 and 5.

4. Sequential Auction

Many popular heuristics use auction-based allocation to quickly distribute tasks to robots. Robots bid on tasks, and an auctioneer allocates tasks based on robot bids. Sequential single-item auctions allocate one task per auction round, and robots create new bids based on the tasks that they have received. Robots initially have no tasks allocated to them. They then bid on each task. The overall best bid is the winner, and a robot is allocated the task. For each remaining unallocated task, the robot calculates the cost of adding the task to its tour, and bids accordingly. Another auction round is held and the process repeats until all tasks are allocated. It has been empirically shown to outperform other auction methods such as round-robin, ordered single-item auction, and parallel single-item auction [4, 5]. Sequential single-item auctions also offer a guarantee of closeness to optimal for minimising the sum of robot times [3]. The performance and objective of sequential single-item auctions can be adjusted through selection of bidding algorithms and auction resolution algorithms.

4.1. Bidding Algorithms

Robots must independently calculate bids for each task in each auction round. The bidding algorithm determines what tasks they bid on, and how much they bid for them [3]. In each round, there is a set of unallocated tasks, and robots have initially empty paths.

The robots calculate the cost of including a task in their path, and bid accordingly. In order to calculate the new path costs, robots must independently solve a TSP each auction round. We use the insertion heuristic [3] because it does not require re-building paths each auction round, hence being very quick, while still providing good solutions relative to other heuristics.

4.2. Auction Resolution

The robot that prompts an auction becomes the auctioneer, and must be capable of determining which tasks to allocate to which robots based on their bids. For multi-robot routing, it is common for this to be the Lowest Bid (LB) [3]. The auctioneer takes the overall lowest bid and assigns that task to its bidder. As a consequence of this, robots only need to communicate their lowest bid to the auctioneer.

4.3. Tool Swap

The proposed bidding format involves bidding on tool-task pairs as well as individual tasks. This would change the sequential single-item auction into a combinatorial auction, where combinations or 'bundles' or tasks are bid on. Combinatorial auctions can produce better solutions, but each additional combination increases the processing time. Combinatorial auctions can scale factorially if they use every combination of tasks, making it unusable with even a moderate number of tasks.

We add the option for robots to bid on tool-task pairs. Rather than bidding on every possible tool-task pair, which would increase the number of calculations from $NumTasks$ to $NumTasks * (NumTools + 1)$, we only need to consider swapping to the best tool for the job. This reduces the number of calculations to $NumTasks * 2$. In essence, each robot bids on each unallocated task twice: once if it were to keep the tool it has; and once if it were to swap and use the tool most suited for that task. The number of auction rounds remains the same, as one task will be allocated each auction round.

Table 1: The tuned genetic algorithm properties.

Population	100
Replacement	20%
Crossover Rate	95%
Mutate-External	2/12
Mutate-Reverse	4/12
Mutate-ToolChange	3/12
Mutate-ToolMatch	3/12

Table 2: The tuned genetic algorithm properties with seeding.

Population	100
Replacement	20%
Crossover Rate	80%
Mutate-External	1/4
Mutate-Reverse	1/4
Mutate-ToolChange	1/4
Mutate-ToolMatch	1/4

While combinatorial auctions can vastly increase the time taken to find a solution due to the large increase in options to calculate [6], our procedure only doubles the number of bids.

5. Genetic Algorithm

Genetic algorithms search for good solutions by randomly generating, breeding, and evolving solutions. Solutions are encoded as genomes. The population is first randomly generated, and the fitness of the solutions are ranked. Parents are then selected from the population using weighted random selection, and combined to produce children. The children are formed from their parents using crossover and mutation functions. The children are then added to the population (steady-state GA) or used to form the next generation (standard GA). We make use of steady-state GA. In a steady-state GA, the population is cut back through weighted random selection. The basic concept is that genes which are fitter will survive and breed, while those that are less fit will be removed. After a certain number of generations the best solution is selected. The parameters are listed in Table 1.

5.1. Representation

Solutions to the problem must be represented as a genome. For the TSP, the genome indicates the order that tasks should be completed. For multi-robot routing, the genome is extended to specify the number of tasks allocated to each robot [12]. This is shown in Figure 4. For multi-robot routing with tools, we propose a further addition that represents the tool to be used for each task. This is encoded as a longer vector, but may be represented as a second row to visually identify the tool used for each task. This is shown in Figure 5.

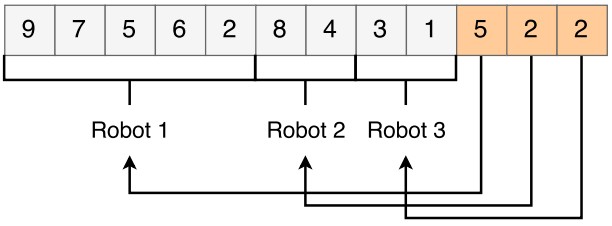


Figure 4: The genetic algorithm representation for the Multiple Travelling Salesman Problem. Robot 1 will complete tasks 9, 7, 5, 6, 2. Robot 2 will complete tasks 8, 4. Robot 3 will complete tasks 3, 1.

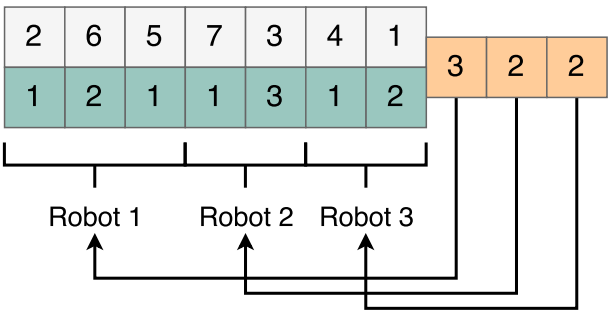


Figure 5: The genetic algorithm representation for the Multiple Travelling Salesman Problem with tools. Robot 1 will complete task 2 with tool 1, task 6 with tool 2, task 5 with tool 1. Robot 2 will complete task 7 with tool 1, task 3 with tool 3. Robot 3 will complete task 4 with tool 1, task 1 with tool 2.

5.2. Population

The population is created by randomizing the tasks and robot sections. The tools are randomised using weighted random sampling. Tools that are better at being used to complete a given task are more likely to be selected. This means that incompatible tools will never be selected, and the use of inefficient tools can be explored but are unlikely to dominate the search. We make use of steady-state GA, where new solutions replace the worst solutions in the population pool. With a population size of 100 and replacement rate of 20%, each generation will replace the worst 20 solutions with new candidates. Convergence time can be improved by seeding the GA with solutions found from heuristics. Instead of randomly generating solutions, a solution method such as a sequential auction is used. This can improve results, but may also lead to premature convergence [12]. The tuned values for the seeded genetic algorithm are listed in Table 2.

5.3. Population Selection

Each generation, a number of candidates are selected from the population. These candidates are used to create children for the next generation. Commonly used selection techniques include tournament selection, where the population is randomly grouped and the best in each group is selected; roulette wheel selection, where the likelihood of being selected scales with solution quality; and rank-based

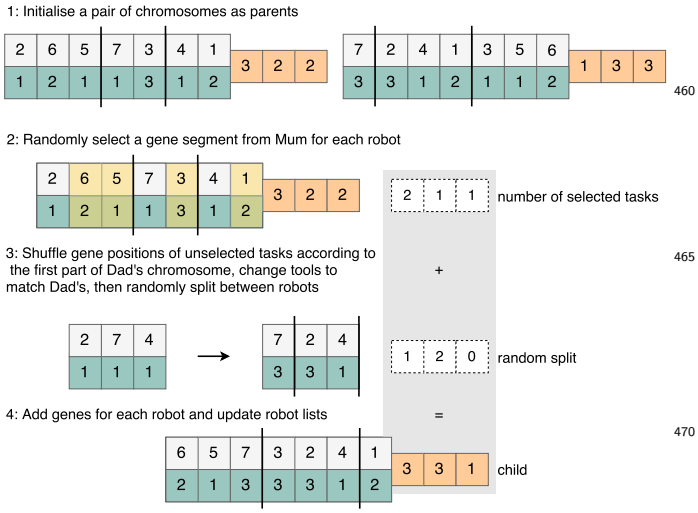


Figure 6: An overview of the Two-part Chromosome Crossover (TCX) [12] with tool usage. Two parents (Mum and Dad) are selected for crossover. A subsection of each robot path is selected from Mum, marked by yellow shading. In this case tasks 6 and 5 for robot 1, task 3 for robot 2, and task 1 for robot 3. Their corresponding tools are also selected. The remaining tasks (2, 7, 4) are rearranged to match the order in Dad (7, 2, 4), and the tools are replaced. These are randomly distributed between the robots, in this case one task is given to robot 1, two tasks are given to robot 2, and no tasks are given to robot 3. The tasks selected in step 2 are combined with the distributed tasks to form a child gene.

roulette wheel selection, where the likelihood of being selected scales with the rank of solution quality [42]. We make use of rank-based roulette wheel selection.

5.4. Crossover

When two parents have been selected from the population, there is a chance they are used to create children. This process is done with a crossover operator. The crossover operator has the purpose of maintaining good sub-tours while generating new solutions. It is also important that children represent valid solutions, otherwise time is wasted exploring invalid solutions. There are many crossover operators that have been designed. We make use of the two-part chromosome crossover (TCX) designed for the mTSP [12]. With the introduction of tools, the crossover function also has the purpose of maintaining good task-tool pairings. To adopt TCX to include tools, tasks and tools are paired together. Any movement of tasks in the TCX process will also apply to the tool for that task. This ensures the crossover will not pair a task with a tool that cannot be used to complete it. The TCX operator with tools is shown in Figure 6.

5.5. Mutation

Mutations occur to periodically introduce new solution types into the population. We apply mutation whenever a solution is selected and not used for crossover. Any time a task is moved, the tool being used goes with it. We use four mutation types:

1. External Swap: Two tasks belonging to two robots are randomly selected and swapped.
2. Reverse: A subpath belonging to one robot is randomly selected, and the subpath is reversed. This process is used in 2-opt improvement [43].
3. Tool Change: A random task is selected and its corresponding tool is changed to another valid tool.
4. Tool Match: A subpath belonging to one robot is randomly selected. All tasks in the subpath will change tools (if valid) to match that of a random task in the subpath.

External Swap and Reverse are standard mutations for multi-robot routing. Tool Change and Tool Match are introduced in this paper. The purpose of Tool Change is to explore new task-tool pairings while ensuring the combination is valid. The purpose of Tool Match is to explore solutions where several tasks can be completed without changing tools. The odds for each mutation are listed in Table 1 and 2.

6. Worst Case Analysis

We ensure the guaranteed performance of the transformation and sequential auction by analysing worst-case scenarios for time and solution quality.

6.1. Time

For the transformation outlined in Section 3, we have N robots and M tasks forming V vertices, resulting in $\frac{V(V-1)}{2}$ edges. For each edge, we compare the extra distance from using one of the TR tool racks. Therefore, we have a time of $O(TR * V^2)$ to perform the transformation.

For a standard sequential single-item auction, the time is as follows: Each auction round $O(M)$: each unallocated task $O(M)$: is temporarily inserted at each path $O(M)$: and the path cost calculated $O(M)$. This gives a total order of $O(M^4)$.

For sequential auctions with combinatorial tool swaps outlined in Section 4, this becomes: Each auction round $O(M)$: each unallocated task $O(M)$: is temporarily inserted at each path $O(M)$ and tool-swap cost is calculated for each tool rack $O(TR)$: and the path cost calculated $O(M)$ This gives a total order of $O(M^4 + TR * M^3)$.

The time to calculate the transformation is $O(TR * V^2)$, and the extra time by using sequential auctions with combinatorial tool swaps is $O(TR * M^3)$. $V > M$, so it is possible for $V^2 > M^3$, but for problems where there are more tasks than robots, it is expected that M is close to V, so that applying the transformation and solving with a standard sequential auction will be faster in most applied cases.

Genetic algorithms have customisable convergence times and do not have a worst-case time.

Table 3: The expertise table for a set of tools and task types.

	Type 1	Type 2	Type 3	...	Type M
Tool 1	E_1	E_2	E_3	...	E_M
Tool 2	1	0	0		0
Tool 3	0	1	0		0
Tool 4	0	0	1		0
...				...	
Tool M	0	0	0		1

Table 4: The possible costs to complete a task. The costs involve the cost to travel to the task (tc), the cost to complete the task once it has been reached (bc) which is amplified by tool expertise (E_i), and the extra cost to switch tools (sc). The tool expertise will depend on whether a general purpose tool is used, or whether to swap to use the best tool.

Edge	(1) Using general tool	(2) Using best tool	(2) - (1)
$T_i \to T_i$	$tc + \frac{bc}{E_i}$	$tc + bc$	$bc - \frac{bc}{E_i}$
$T_i \to T_j$	$tc + \frac{bc}{E_j}$	$tc + sc + bc$	$bc - \frac{bc}{E_j} + sc$
start $\to T_i$	$tc + sc + \frac{bc}{E_i}$	$tc + sc + bc$	$bc - \frac{bc}{E_i}$
$T_i \to$end	$tc + sc$	$tc + sc$	0

6.2. Solution Quality

We analyse the possible loss of optimal results through the transformation process. The transformation constraint is that each task must be completed using the tool that is best suited for it. We consider the existence of tools that can be used to complete more than one task type, and provide an upper bound on the expertise of these tools that maintain the optimal solution for both MiniSum and MiniMax.

Consider a set of M tasks of type T. For each robot there is an optimal path for minimising the MiniSum or MiniMax objective. If each path is maintained in the transformation process, we have not lost solution optimality. If a path violates the transformation constraints, then we may have lost solution optimality.

Without loss of generality, we represent the tool expertise in Table 3. There is a general-purpose tool which will never be included in the transformation, as there is no task which it is best for. If we can ignore this tool without increasing path cost, the optimal solution is guaranteed to be included in the transformation. Alternatively, if the path cost must increase if this tool is ignored, then the optimal solution may be discarded during the transformation.

The costs for using or ignoring the general purpose tool are listed in Table 4. The travel cost (tc) is the movement cost from one task to another. The base cost (bc) is the cost to complete the task once arrived. This is amplified by the expertise value (E) of the tool, which is 1 for the best tool for that task. The swap cost (sc) is the extra cost required to move to a tool rack, equip a new tool, then move to the task.

The path cost will increase if the third column of Table 4 is greater than 0. We know that $E \leq 1$, so $bc - \frac{bc}{E} \leq 0$. The only way for the path cost to increase is by $T_i \to T_j$ transitions, i.e., the general purpose tool is most useful when the robot would need to change tools often.

The worst case scenario is therefore an optimal path in the form:

$$\text{start} \to T_i \to T_j \to T_i \to T_j \to T_i \to \cdots \to T_i \to \text{end} \quad (8)$$

If we use constant task costs and swap costs for each segment (i.e., let tc be the biggest task cost, and let sc be the biggest swap cost) this is equal to:

$$\text{start} \to T_i + n(T_i \to T_j) + n(T_j \to T_i) + T_i \to \text{end} \quad (9)$$

where n is half the number of tasks. The cost increase from executing this path without the general purpose tool can be calculated from the last column of Table 4. Optimality is maintained if the cost does not increase, therefore optimality is maintained if:

$$0 \geq bc - \frac{bc}{E_i} + n(bc - \frac{bc}{E_j} + sc) + n(bc - \frac{bc}{E_i} + sc) + 0 \quad (10)$$

We replace individual expertise values with E_{max2}, which represents the second-highest expertise value for a given task type. $E_{max} = \max(\{E_1, E_2, E_3 \dots E_M\})$, $E_{max2} = \max(\{E_1, E_2, E_3 \dots E_M\} - \{E_{max}\})$.

$$0 \geq bc - \frac{bc}{E_{max2}} + 2n(bc - \frac{bc}{E_{max2}} + sc) \quad (11)$$

$$E_{max2} \leq \frac{bc}{\frac{2n}{2n+1} * sc + bc} \quad (12)$$

A close upper bound on this is:

$$E_{max2} \leq \frac{bc}{2 * sc + bc} \quad (13)$$

For example, say the tasks have cost 60 and the biggest swap cost is 45.

$$E_{max2} \leq \frac{60}{2 * 45 + 60} \quad (14)$$

$$E_{max2} \leq 0.4 \quad (15)$$

As long as a tool does not have a second-highest expertise of 0.4, then the optimal solution is guaranteed to remain in the transformed problem. If the second best expertise is greater than 0.4, the optimal solution may or may not be conserved when transforming the problem. This upper limit is simple to calculate, as the biggest base cost and the biggest swap cost are both found when performing the transformation.

7. Experiments

Experiments were conducted to compare the performance of the adapted sequential auctions and genetic algorithms with and without the problem transformation.

We altered the following independent variables: number of tasks, number of robots, number of tool racks, the objective (minisum/minimax), and the solution algorithm (auction, GA and multiple integer linear program with and without transformation). We then measured the path cost and processing time. Other variables such as robot speed, the number of tools, and area size were held constant to keep the number of tests feasible.

Tasks, robots, and tool racks were created randomly in a 100x100m area. Robots are assumed to travel at 1 m/s, and each task takes 10 seconds to complete with the best tool. Non-best tools can be used to complete tasks, and will take $10/E$ seconds to complete, where $0 \leq E \leq 1$ is the expertise. There is an equal amount of each task type. Each tool has an expertise of 1 for one task type, and random expertise for all other task types, up to a cap. When the cap is 0, we describe this as a system with task-tool pairs, i.e., each task can only be completed by one tool. Each test was performed 200 times and the average recorded.

The experiments were performed using MATLAB for its deterministic results. The processing was performed using the Phoenix high-performance computation cluster at The University of Adelaide. Tests were done in parallel, where each test was given a single-core with 2 GB of memory. This had very similar processing speed to a 64-bit laptop with i5-3320M CPU @ 2.60GHz.

8. Computational Results

The code and data for these results are available online, hosted by The University of Adelaide (`http://dx.doi.org/10.4225/55/5b1631ce8b456`).

8.1. Task-Tool Pairs

The first experiments considered task-tool pairs, where each task can only be completed by one tool. In Tables 5 and 6, we see the results for the MiniSum and MiniMax objectives, respectively. For MiniSum, three techniques are compared: an integer linear program that guarantees optimal results [30]; a sequential auction outlined in Section 4; and a genetic algorithm outlined in Section 5. The algorithms are tested on the multi-robot routing problem (without tools) resulting from the transformation described in Section 3 (TF), as well as the direct tool-based multi-robot routing problem without transformation (no-TF). Note that the transformation is guaranteed to maintain the optimal solution for task-tool pairs, as all other tool expertise is 0. Hence, the optimal solutions found for the transformed problem are guaranteed to be optimal for the untransformed problem For the MiniMax objective, no optimal solutions were used, as optimality for the MiniMax objective is an open problem.

For both MiniSum and MiniMax, we can see that auctioning produces similar results with and without the transformation. As the number of tool racks increases, the transformed solutions slowly become better than the untransformed solutions ($< 0.03\%$ for 1 and 4 tool racks, 1.6% for 8 tool racks). For the MiniSum objective, the auction solutions are worse than optimal for 1, 4, and 8 tool racks by 7%, 10%, and 10% respectively.

Table 5: MiniSum Performance with task-tool pairs

TR	T	R	Opt	Auction		GA		GA(Seeded)	
			tf	tf	no-tf	tf	no-tf	tf	no-tf
1	50	3	2123	2201	2201	2427	2434	2138	2141
		5	2107	2191	2190	2670	2662	2121	2124
		10	2095	2177	2177	2775	2780	2116	2117
	100	3	3084	3305	3305	3460	3489	3164	3182
		5	3072	3298	3296	4035	4006	3155	3171
		10	3057	3272	3272	4513	4563	3145	3152
	150	3	3933	4247	4247	4437	4518	4079	4100
		5	3927	4243	4243	5197	5216	4081	4095
		10	3912	4223	4223	6025	6169	4090	4087
4	50	3	1926	2073	2093	2164	2167	1959	1959
		5	1911	2073	2091	2337	2333	1964	1962
		10	1910	2076	2092	2453	2455	1982	1988
	100	3	2933	3233	3229	3317	3347	3050	3059
		5	2930	3227	3224	3669	3660	3063	3066
		10	2926	3226	3223	4002	4055	3073	3065
	150	3	3812	4203	4209	4371	4418	3996	4021
		5	3798	4184	4185	4798	4860	3995	4009
		10	3792	4184	4190	5402	5443	4039	4019
8	50	3	1832	1975	2046	2033	2046	1881	1896
		5	1825	1962	2049	2186	2183	1876	1897
		10	1823	1976	2047	2281	2313	1901	1930
	100	3	2881	3163	3202	3231	3270	2997	3007
		5	2865	3149	3201	3514	3526	2990	2998
		10	2865	3157	3199	3808	3830	3018	3019
	150	3	3760	4153	4185	4293	4336	3950	3965
		5	3751	4133	4173	4639	4673	3947	3959
		10	3750	4138	4158	5143	5178	3994	3968

The genetic algorithm produces allocations with a quality that is largely dependent on the number of tasks and robots. As the complexity of the problem increases, so too does the number of solutions, making it harder for the GA to find good solutions. Because of this, GA performs drastically worse as the problem complexity increases. It is outperformed by the auction process even for relatively small problems, although it would be expected to outperform auction for very small problem sizes (10 tasks). However, the GA is superior if initially seeded with solutions from the auction process.

In the MiniSum objective, the seeded GA solutions are worse than optimal for 1, 4, and 8 tool racks by 3%, 5%, and 5% respectively. The use of the transformation makes little difference to the GA, with and without seeding.

The processing time of each algorithm is also an important factor. Looking at Table 7, we firstly see expected timing trends: auctions are very quick; branch-and-cut is

Table 6: MiniMax Performance with task-tool pairs

TR	T	R	Auction		GA		GA(Seeded)	
			tf	no-tf	tf	no-tf	tf	no-tf
1	50	3	938	938	1039	1058	902	902
		5	667	668	754	779	637	639
		10	478	478	534	549	456	457
	100	3	1316	1316	1521	1550	1268	1271
		5	884	884	1107	1129	853	856
		10	599	599	766	766	575	575
	150	3	1634	1634	1893	1915	1586	1589
		5	1066	1066	1392	1422	1035	1036
		10	704	704	968	991	677	678
4	50	3	817	827	839	858	781	786
		5	563	565	593	599	531	540
		10	377	379	414	421	351	355
	100	3	1201	1214	1306	1335	1165	1173
		5	790	797	903	925	763	770
		10	493	495	617	627	474	475
	150	3	1528	1540	1707	1740	1484	1488
		5	995	995	1178	1215	964	965
		10	593	596	798	809	573	578
8	50	3	757	782	777	792	728	734
		5	513	529	539	551	489	496
		10	333	336	378	381	310	312
	100	3	1162	1179	1235	1263	1122	1129
		5	751	761	833	852	731	734
		10	453	458	565	573	435	436
	150	3	1485	1508	1632	1666	1442	1453
		5	951	967	1103	1124	924	932
		10	554	557	741	751	537	540

Table 7: MiniSum Time with task-tool pairs (seconds)

Tasks	Robots	Optimal(tf)		Auction(tf)		Auction		GA(tf)		GA		GA(seeded,tf)		GA(seeded)	
		mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
50	3	1.5	1.1	1.8	0.2	2.7	0.3	602.7	48.0	594.9	30.4	258.1	17.8	263.0	17.8
	5	3.5	1.5	1.8	0.2	2.7	0.3	716.6	43.4	678.1	37.0	259.1	19.5	278.6	16.2
	10	17.9	8.3	1.8	0.2	2.8	0.3	802.1	67.6	790.0	56.5	287.7	18.1	307.2	23.5
100	3	12.0	9.4	7.4	1.3	13.0	1.9	3306.9	289.4	3251.6	185.3	1613.1	97.5	1591.9	75.3
	5	31.7	21.7	7.6	1.3	12.7	2.0	3846.4	259.8	3635.0	175.5	1630.7	93.5	1723.1	87.2
	10	181.9	76.8	7.7	1.3	12.4	2.4	4592.0	317.0	4427.0	242.5	1820.7	116.6	1925.4	101.5
150	3	56.9	59.2	28.8	4.9	43.2	5.9	6970.8	365.3	6897.9	398.9	2857.9	147.5	2939.5	134.4
	5	147.7	138.2	29.2	4.5	43.3	6.4	8099.3	486.6	7563.6	562.1	2891.0	160.9	3142.2	187.2
	10	709.2	344.8	29.4	4.7	43.2	7.6	9728.2	754.6	9210.0	829.3	3263.6	211.5	3304.1	149.9

Table 8: MiniMax Time with task-tool pairs (seconds)

Tasks	Robots	Auction(tf)		Auction		GA(tf)		GA		GA(seeded,tf)		GA(seeded)	
		mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
50	3	1.3	0.1	2.3	0.2	607.5	34.3	608.9	46.4	281.9	17.1	286.5	18.4
	5	1.4	0.1	2.5	0.2	689.1	42.0	648.4	57.2	308.6	17.8	316.8	17.8
	10	1.8	0.2	3.0	0.2	859.3	70.2	883.4	53.6	392.2	26.6	387.7	17.9
100	3	3.0	0.2	7.1	0.8	3225.7	224.5	3282.2	193.5	1779.1	104.8	1846.5	80.6
	5	3.6	0.2	7.2	0.6	3751.4	245.0	3749.5	207.0	2026.3	132.4	2027.1	99.2
	10	4.2	0.2	8.8	0.6	4611.2	291.0	4519.5	232.5	2426.7	152.2	2467.5	155.9
150	3	7.6	0.5	18.8	2.1	6663.1	422.7	6997.6	684.8	3224.9	169.6	3325.1	142.6
	5	7.4	0.6	18.1	1.9	7510.6	510.1	7608.7	407.2	3713.7	198.8	3639.0	135.3
	10	8.9	0.7	20.8	1.6	9239.7	624.8	9157.4	463.9	4307.9	260.9	4452.9	167.0

very quick for small cases but has trouble at scale; and genetic algorithms have a customisable convergence time. In this case, a long convergence time was required to find good results. Of particular importance is the time differences between the transformed and non-transformed cases. For the auction, transforming the problem reduces computation time quite significantly. This is particularly relevant for the MiniMax case in Table 8, where the time required in the transformed case is less than half of the untransformed case. Seeding the GA also improves convergence time significantly.

Note that processing was done on a single processor. If multiple processors were used the auction process would be faster due to parallelised bid calculations; and the genetic algorithm performance would be greater due to multiple attempts.

8.2. Overlapping Tool Expertise

If more than one tool can be used to complete a given task, the robots have to choose between using the tool that is currently equipped, and moving to a tool rack to select another tool.

In these experiments, each tool has one task type for which it has perfect expertise, but all other expertise values are randomly generated between 0 and an expertise cap. An expertise cap of 0 produces task-tool pairings, which produces the same results as in Tables 5 and 6.

Figures 7 and 8 show algorithm performances when using tools with overlapping expertise. The marked points represent solutions without using a transformation, with a cubic interpolation drawn between them. The horizontal line is the solution cost for the transformed problem. As discussed in Section 3, the transformation process ignores all of the other expertise values because it enforces the use of the highest expertise tool for each task.

The bounds (overlap of 0 and overlap of 1) for auction and GA are as one would expect, having tools which can be used for multiple tasks means fewer tool swaps are necessary. This means that the overall cost should go down. However, this is not a linear trend. Instead, the costs increase for a while as the cap increases, before it decreases. The reason for this is that low expertise overlap introduces valid (but unlikely to be good) solutions, which makes finding good solutions more difficult. This implies that it may be worthwhile to ban certain tool usage in order to reduce the search space. The proposed transformation does this intrinsically, which explains its high performance even for rather large tool expertise overlap. This could also be done using a manual or calculated threshold, for example, we could ban any tool-task expertise below 0.4. The performance by using this threshold would largely depend on the system.

This local maxima shifts to the right as more tool racks are used, but is unaffected by the number of tasks and robots.

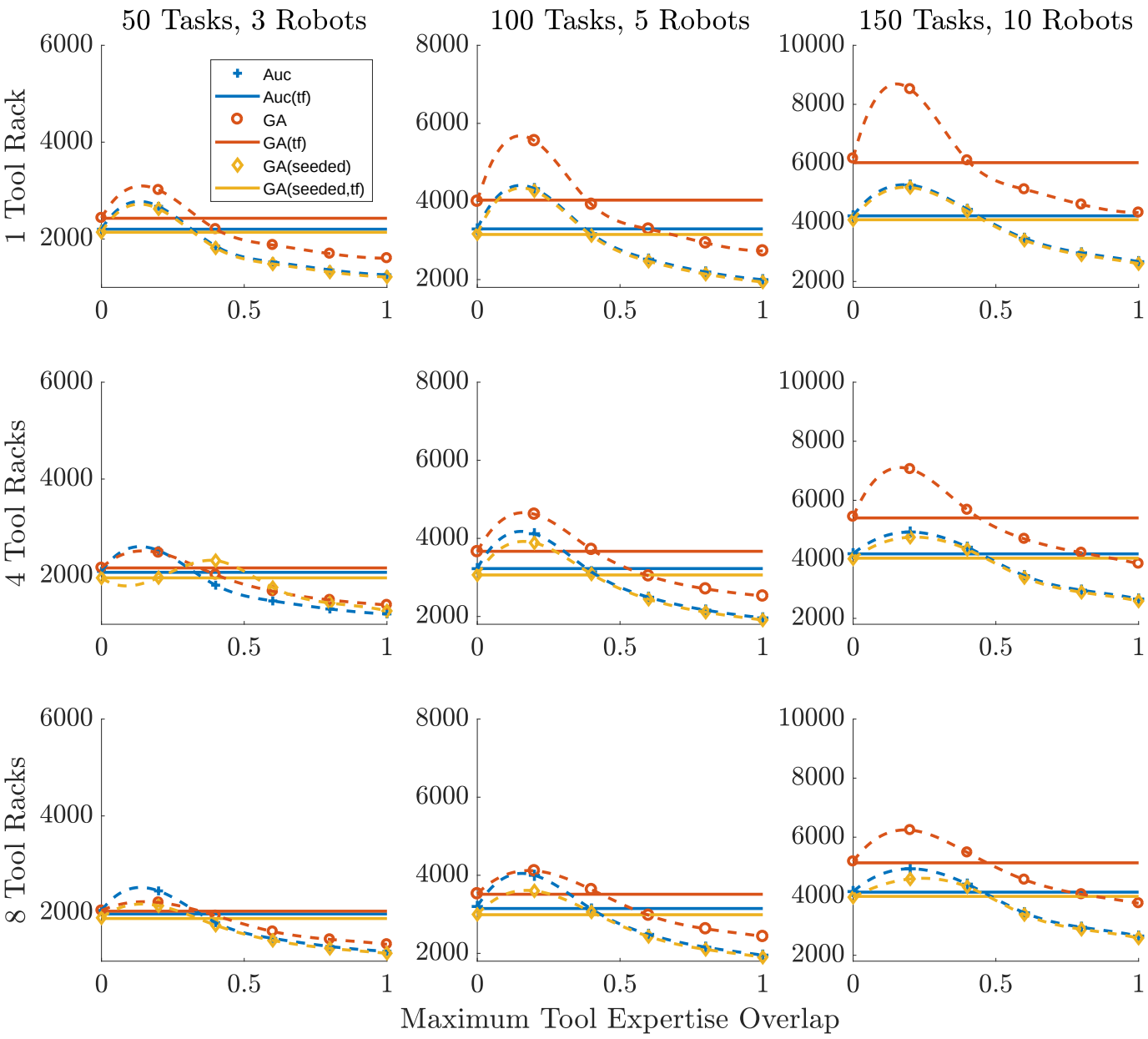


Figure 7: The performance of a sequential auction and genetic algorithm with overlapping tool capabilities for the MiniSum objective.

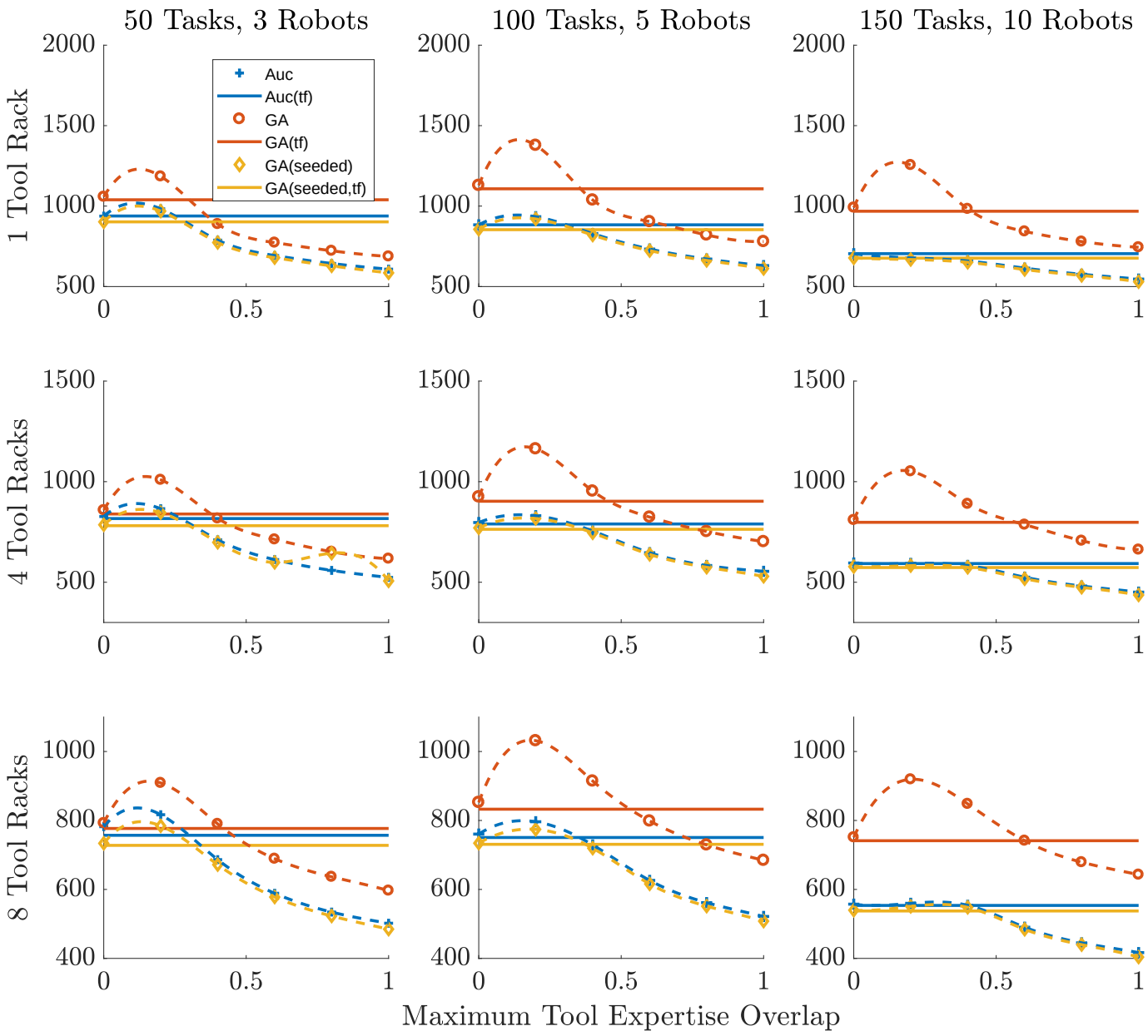


Figure 8: The performance of a sequential auction and genetic algorithm with overlapping tool capabilities for the MiniMax objective.

9. Benchmark Tests

We also test the performance of each algorithm in a number of benchmark problems from the TSPLIB library [44]. While this library does not have tool-swapping multi-robot routing problems, the TSP problems are a widely used benchmark for TSP problems, and have been adapted for use in multi-robot routing algorithms [19]. One adaptation is to set all robots to start at the first node. While this works for the MiniMax objective, the MiniSum objective is equivalent to a TSP. This can be avoided by adding additional constraints, such as minimum task completion requirements for each robot. This is not sufficient for our purposes, where a number of tool racks must also be used. Other conversions have used random placement of robots, resulting in a problem that cannot easily be reproduced. We believe this defeats the purpose of using a standard library. We use a process that converts the TSP to multi-robot routing in a reproducible, non-random way which does not clump the robots and tool racks on the same point.

Every N^{th} node (starting from node 1) represents a robot start location. Every M^{th} node (starting from node 2) represents a tool rack location. The tasks types are distributed evenly. If a node is marked as both a robot start location and a tool rack, it is a robot start location. N and M are selected first to ensure there are enough nodes for the desired number of robots and tool racks. For example, in Table 9 we wish to test up to 3 robots and up to 2 tool racks, so we can select $N = 5$ and $M = 10$. If we wish to compare the 3 robot case with the 10 robot case, we do **not** change N, as this would change the number of tasks. Instead, we use some of the robot locations and ignore the unused ones. An equation to spread the robots out is listed in Equation (16). This same equation is also used for tool racks. We use a rounding function that rounds to odd for values with a decimal value of 0.5.

$$r = \text{round}(\frac{maxrobots + 1}{numrobots + 1} * k), k \in \{1, 2, \ldots, numrobots\} \tag{16}$$

Tables 10 and 11 show the performance of algorithms for standard TSPLIB problems, Pr76, Pr152, and Pr226. An example problem and solution from the TSPLIB library is shown in Figure 9. The conversion from TSP to multi-robot routing with tools is listed in Section 7. We present our solutions for three problems with a differing number of tools, robots, and racks.

Each tool has an expertise of 1 for the diagonal in the expertise table, and all other values are 0.2. Each task has a cost of 600, and each tool swap has a cost of 1200. This results in E values of 0.013, 0.017, and 0.016 for Pr76, Pr152, and Pr226 respectively. These values were selected such that the sequential auction and GA adaptations are unlikely to perform well due to slight tool overlap, as discussed earlier, and that optimality is not guaranteed to be

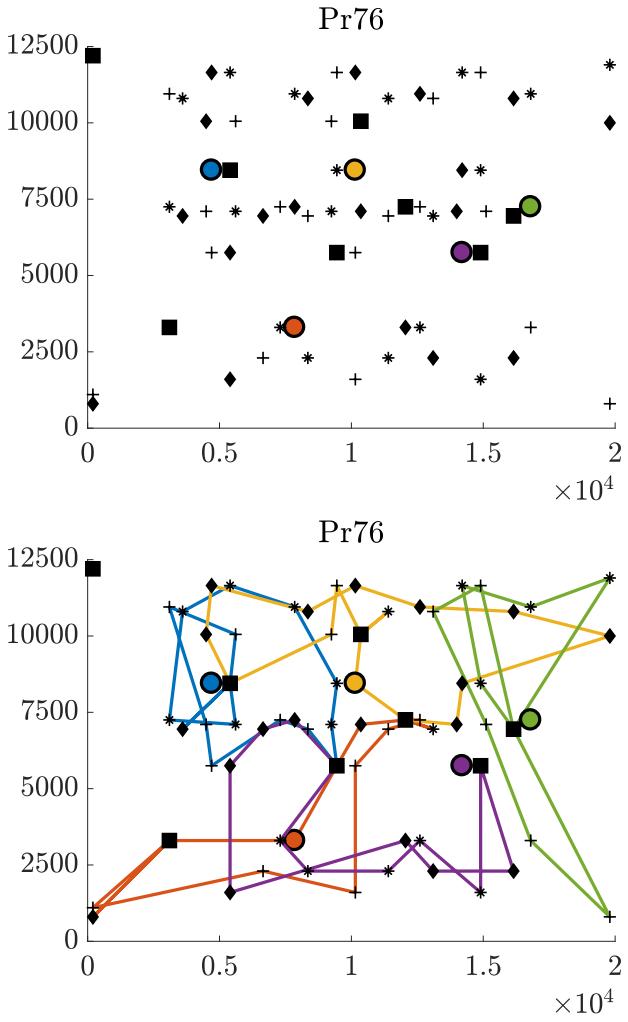


Figure 9: A multi-robot routing with tools problem from the publicly available TSPLIB. The five robots (circles) form path plans to complete tasks (+, *, ♦). They must make use of tools located at tool racks (squares), and return the tools before returning to their start location. The bottom graph shows an example solution for the MiniMax objective.

Table 9: TSPLIB to multi-robot routing for 13 points and 3 task types. We wish to compare up to 3 robots and up to 2 tool racks, so we select for a robot every N nodes and a tool rack every M nodes. This is an example of $N = 5$, $M = 10$. The effect of number of robots (R) and tool racks (TR) can then be compared without changing the location or amount of tasks.

Node	(3R,2TR)	(2R,2TR)	(1R,1TR)
1	Robot	Robot	-
2	Tool Rack	Tool Rack	Tool Rack
3	Task 1	Task 1	Task 1
4	Task 2	Task 2	Task 2
5	Task 3	Task 3	Task 3
6	Robot	-	Robot
7	Task 1	Task 1	Task 1
8	Task 2	Task 2	Task 2
9	Task 3	Task 3	Task 3
10	Task 1	Task 1	Task 1
11	Robot	Robot	-
12	Tool Rack	Tool Rack	-
13	Task 2	Task 2	Task 2

maintained when transforming. Effectively this is a problem that all known algorithms are weak at. This is a key area where future algorithms can make improvements.

From Tables 10 and 11 we can see that the use of the transformation improves the results. For the sequential auction, this is especially true with low numbers of robots for both MiniSum and MiniMax objectives.

10. Conclusion

We have provided three algorithms to solve the multi-robot routing optimisation problem with adaptive heterogeneity. Firstly, a transformation that converts the problem into a standard multi-robot routing problem was introduced, with a proven upper bound on its ability to maintain optimal solutions. Secondly, adaptations to the sequential single-item auction bidding and resolution phases were made to find fast solutions. Thirdly, adaptations were made to genetic algorithm representation, along with crossover and mutation functions. The quality and speed of these solutions were compared and empirically tested to illustrate their performance for a range of heterogeneity overlaps.

In particular, directly solving the problem with auctions and genetic algorithms provides good results when heterogeneity overlap is high, but produce poor results at low heterogeneity overlap due to the introduction of many poor solutions. Transforming the problem avoids this gap in performance, and allows other techniques to be used, such as mixed-integer linear programs. Furthermore, transforming the problem also prevents exploitation of tools with high heterogeneity overlap.

Finally, the algorithms are benchmarked on repeatable TSPLib problems such that any algorithms made in the future can be quantitatively compared.

Acknowledgment

This research was supported by the Phoenix High Performance Computing (HPC) service at the University of Adelaide, an Australian Government Research Training Program (RTP) Scholarship, and by the Commonwealth of Australia (represented by the Defence Science and Technology Group) through a Defence Science Partnerships agreement.

References

[1] J. R. Montoya-Torres, J. L. Franco, S. N. Isaza, H. F. Jimnez, N. Herazo-Padilla, A literature review on the vehicle routing problem with multiple depots, Computers & Industrial Engineering 79 (2015) 115–129.

[2] R. Kulkarni, P. R. Bhave, Integer programming formulations of vehicle routing problems, European Journal of Operational Research 20 (1) (1985) 58–67.

[3] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. J. Kleywegt, S. Koenig, C. A. Tovey, A. Meyerson, S. Jain, Auction-based multi-robot routing, in: Robotics: Science and Systems, Vol. 5, Rome, Italy, 2005, pp. 343–350.

[4] S. Koenig, C. Tovey, M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, A. Meyerson, S. Jain, The power of sequential single-item auctions for agent coordination, in: Proceedings of the National Conference on Artificial Intelligence, Vol. 21, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006, p. 1625.

[5] E. Schneider, E. I. Sklar, S. Parsons, A. T. Õzgelen, Auction-based task allocation for multi-robot teams in dynamic environments, in: Conference Towards Autonomous Robotic Systems, Springer, 2015, pp. 246–257.

[6] D. C. Parkes, Iterative combinatorial auctions, MIT Press, 2006.

[7] M. B. Dias, Traderbots: A new paradigm for robust and efficient multirobot coordination in dynamic environments, Phd thesis, Robotics Institute, Carnegie Mellon University (2004).

[8] O. Olorunda, A. P. Engelbrecht, Measuring exploration/exploitation in particle swarms using swarm diversity, in: Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on, IEEE, 2008, pp. 1128–1134.

[9] M. H. Qais, H. M. Hasanien, S. Alghuwainem, Augmented grey wolf optimizer for grid-connected pmsg-based wind energy conversion systems, Applied Soft Computing.

[10] A. Saxena, B. P. Soni, R. Kumar, V. Gupta, Intelligent grey wolf optimizerdevelopment and application for strategic bidding in uniform price spot energy market, Applied Soft Computing 69 (2018) 1–13.

[11] H. Liu, B. Xu, D. Lu, G. Zhang, A path planning approach for crowd evacuation in buildings based on improved artificial bee colony algorithm, Applied Soft Computing 68 (2018) 360–376.

[12] S. Yuan, B. Skinner, S. Huang, D. Liu, A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms, European Journal of Operational Research 228 (1) (2013) 72–82.

[13] H. Nazif, L. S. Lee, Optimised crossover genetic algorithm for capacitated vehicle routing problem, Applied Mathematical Modelling 36 (5) (2012) 2110–2117.

[14] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, W. Rei, A hybrid genetic algorithm for multidepot and periodic vehicle routing problems, Operations Research 60 (3) (2012) 611–624.

[15] A. A. Hosseinabadi, M. Kardgar, M. Shojafar, S. Shamshirband, A. Abraham, Gels-ga: hybrid metaheuristic algorithm for solving multiple travelling salesman problem, in: Intelligent Systems Design and Applications (ISDA), 2014 14th International Conference on, IEEE, 2014, pp. 76–81.

[16] M. M. Paydar, I. Mahdavi, I. Sharafuddin, M. Solimanpur, Applying simulated annealing for designing cellular manufacturing

Table 10: MiniSum Performance with TSPLIB Problems. The problem is either transformed (tf), or solved directly (no-tf).

Problem	Tools	Robots	Racks	Branch&Cut	Auction		GA(tf, seed)			GA(no-tf, seed)		
				tf	tf	no-tf	mean	std	best	mean	std	best
Pr76	3	3	4	183597	235970	256961	232588	344	232009	244490	1718	242191
			8	182222	241961	236310	229675	1823	224914	228225	943	227567
		5	4	183597	242293	284096	231427	1380	228494	276506	216	276386
			8	182222	236515	284096	229097	1318	226991	266189	4209	253506
		10	4	183597	242293	284096	234088	1482	231630	276904	812	276386
			8	182222	236515	284096	229267	1246	228464	269406	3945	259782
	5	3	4	245851	315657	333252	293973	2225	290687	315286	4663	306380
			8	243503	307923	339981	292675	2909	285557	310355	9145	294167
		5	4	245851	307014	311534	295422	2407	290952	306285	322	306238
			8	243503	305318	322855	295333	1808	290878	319620	1761	312746
		10	4	245851	307014	311534	298596	1691	295209	306777	1020	306238
			8	243503	305318	322855	298679	1635	294320	319745	2221	311348
	8	3	4	297825	354012	374538	336568	2867	332178	357666	1981	352495
			8	293368	342822	343274	319698	2043	316266	327955	2722	322332
		5	4	297825	346509	366276	331357	2326	328508	356135	942	355047
			8	293368	336597	357152	319586	2095	315404	350155	1217	341987
		10	4	297825	346509	366276	336937	2874	330662	357230	1379	353779
			8	293368	336597	357152	324853	2099	319831	350303	407	349687
Pr152	3	3	4	154796	271563	360249	260703	1459	255249	329976	11698	303745
			8	154796	257784	340880	250131	4192	235981	292227	2361	285044
		5	4	154796	263476	263476	262028	30	261960	262046	0	262046
			8	154796	253475	263476	251295	207	250347	254438	257	254059
		10	4	154796	263476	263476	262027	42	261825	262046	0	262046
			8	154796	254167	256619	252021	131	251796	255189	0	255189
	5	3	4	211525	356784	530940	336586	4065	328292	479261	8463	458537
			8	208383	334406	438471	308204	6669	299150	358480	5025	343540
		5	4	211525	362505	391686	355951	893	353758	389604	1737	386697
			8	208383	331458	357725	320476	6018	304100	323005	5282	310966
		10	4	211525	362505	391686	355761	934	353705	388806	1958	386697
			8	208383	332147	341448	323711	4266	306059	338696	3324	324907
	8	3	4	287005	486280	637924	430930	3885	421990	566985	5787	551607
			8	281483	421524	526650	389713	2398	384727	423087	5297	408723
		5	4	287005	486933	536268	445458	722	444047	533636	203	532485
			8	281483	415861	481786	390312	2322	384991	430898	5204	418114
		10	4	287005	486933	536268	446162	2115	442498	533643	166	532829
			8	281483	413104	416544	387057	3812	381710	411293	1886	403106
Pr226	3	3	4	238217	346782	789559	341315	210	340606	688990	7528	673199
			8	238217	339231	594548	325736	1239	323501	453524	9568	420727
		5	4	238217	341238	341238	335953	297	335265	336108	97	335532
			8	238217	333030	334423	325874	1663	322328	324201	532	323151
		10	4	238217	341238	341238	336044	260	334857	336129	51	336065
			8	238217	335620	339809	330051	1681	326857	328206	705	326961
	5	3	4	306242	445949	823804	435187	3366	424637	745984	6479	728945
			8	306242	433698	780193	411147	4094	405351	568848	13756	534466
		5	4	306242	449627	476466	443496	309	442706	470617	91	470272
			8	306242	432079	465542	416581	1524	413755	445275	1488	441467
		10	4	306242	449627	476466	443448	308	442473	470617	77	470554
			8	306242	430264	465181	419400	855	417399	448533	1118	446962
	8	3	4	376894	596417	952583	556797	6957	539098	852215	7826	830097
			8	376527	548764	912566	523128	3482	515520	687073	19431	643325
		5	4	376894	589463	659079	570764	1304	567860	656670	250	656531
			8	376527	552145	616739	534323	2809	530403	591721	2457	587771
		10	4	376894	589463	659079	564062	4982	553372	656695	268	656531
			8	376527	547894	613748	527262	2617	523382	595596	2634	591918

Table 11: MiniMax Performance with TSPLIB Problems. The problem is either transformed (tf), or solved directly (no-tf).

Problem	Tools	Robots	Racks	Auction		GA(tf, seed)			GA(no-tf, seed)		
				tf	no-tf	mean	std	best	mean	std	best
Pr76	3	3	4	100272	102964	90624	0	90624	98265	2117	93537
			8	96945	94320	87695	0	87695	93423	708	92780
		5	4	68931	64811	54353	0	54353	64095	130	63989
			8	62709	68775	55035	214	54189	67171	0	67171
		10	4	41789	46779	39618	17	39499	42290	1026	41075
			8	42077	39160	34075	0	34075	39160	0	39160
	5	3	4	121815	126887	113159	1050	110452	123114	1974	118795
			8	114745	119691	107305	1000	106014	115876	1562	112454
		5	4	75933	76978	70229	0	70229	75536	245	74521
			8	72495	75488	67864	1644	66432	71416	1388	69644
		10	4	53852	52931	44052	0	44052	50368	603	49279
			8	46197	47452	41596	173	41024	45207	434	43786
	8	3	4	128582	131371	123189	283	121894	130390	684	126172
			8	123558	130318	118829	954	116767	123182	2181	119321
		5	4	86458	85151	76174	692	75506	94522	174	84331
			8	81913	82261	76699	177	75967	77673	1213	75080
		10	4	56382	54584	51511	1915	48236	52238	946	50966
			8	49913	50529	43653	806	43133	47386	1186	45353
Pr152	3	3	4	105813	139857	99583	0	99583	134586	937	131438
			8	107359	107450	91339	555	90509	105451	601	104439
		5	4	65383	65409	66627	0	66627	65400	0	65400
			8	58916	63862	59021	0	59021	61875	0	61875
		10	4	45962	44604	41052	41	41044	42902	350	42584
			8	33532	33532	34914	21	34895	33522	0	33522
	5	3	4	135490	180570	125811	1070	125174	168280	662	165733
			8	134516	149327	115523	1659	113108	132421	2231	127216
		5	4	87218	89080	79976	0	79976	88356	0	88356
			8	74521	86552	72669	194	72622	82702	987	81819
		10	4	52850	54731	50593	7	50592	54442	0	54442
			8	44115	44604	44858	0	44858	41633	296	41553
	8	3	4	170773	213305	158923	798	158173	206106	24	206102
			8	151612	160248	142201	1430	139439	158259	297	158014
		5	4	105245	119531	98807	1142	96631	119341	306	118683
			8	94514	102599	89296	425	88233	93477	1332	89250
		10	4	64488	65037	60112	824	58361	65037	0	65037
			8	52144	54105	48826	211	48325	52412	372	51899
Pr226	3	3	4	129631	148539	121330	42	121274	140975	342	140473
			8	126342	193205	117089	304	116444	168768	2573	160507
		5	4	75313	80098	74459	129	74285	79674	0	79674
			8	73009	76020	72888	32	72792	75765	51	75716
		10	4	44412	43829	46962	0	46962	43405	0	43405
			8	41208	43297	44538	447	43532	41948	411	41509
	5	3	4	166496	228097	151255	130	151138	217498	520	215402
			8	155872	242431	146779	1691	144954	210036	2445	202077
		5	4	99672	105535	97544	0	97544	102357	0	102357
			8	94780	106047	92554	78	92123	103334	0	103334
		10	4	61446	61728	54066	0	54066	60055	626	59560
			8	52988	58537	55945	266	54823	58333	0	58333
	8	3	4	208471	310497	204522	761	202093	300525	1641	295246
			8	197803	295217	187934	1448	182255	253015	4684	235516
		5	4	130200	142424	126754	0	126754	141388	0	141388
			8	120205	137853	113751	777	112932	137792	136	136842
		10	4	73117	75551	69955	1191	67993	74772	284	74406
			8	70821	73687	64373	0	64373	73161	287	72926

systems using MDmTSP, Computers & Industrial Engineering 59 (4) (2010) 929–936.
[17] J. Grobelny, R. Michalski, A novel version of simulated annealing based on linguistic patterns for solving facility layout problems, Knowledge-Based Systems 124 (2017) 55–69.
[18] L. P. Behnck, D. Doering, C. E. Pereira, A. Rettberg, A modified simulated annealing algorithm for suavs path planning, IFAC-PapersOnLine 48 (10) (2015) 63–68.
[19] Y. Chen, Z. Jia, X. Ai, D. Yang, J. Yu, A modified two-part wolf pack search algorithm for the multiple traveling salesmen problem, Applied Soft Computing 61 (2017) 714–725.
[20] Y. Chen, D. Yang, J. Yu, Multi-UAV task assignment with parameter and time-sensitive uncertainty using modified two-part wolf pack search algorithm, IEEE Transactions on Aerospace and Electronic Systems.
[21] R. Wu, S. Wang, Discrete wolf pack search algorithm based transit network design, in: Software Engineering and Service Science (ICSESS), 2016 7th IEEE International Conference on, IEEE, 2016, pp. 509–512.
[22] S. Ghafurian, N. Javadian, An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems, Applied Soft Computing 11 (1) (2011) 1256–1262.
[23] H. Zhang, X. Wang, P. Memarmoshrefi, D. Hogrefe, A survey of ant colony optimization based routing protocols for mobile ad hoc networks, IEEE Access 5 (2017) 24139–24161.
[24] W. Wu, Y. Tian, T. Jin, A label based ant colony algorithm for heterogeneous vehicle routing with mixed backhaul, Applied Soft Computing 47 (2016) 224–234.
[25] X. Chen, P. Zhang, G. Du, F. Li, Ant colony optimization based memetic algorithm to solve bi-objective multiple traveling salesmen problem for multi-robot systems, IEEE Access.
[26] E. Tuci, Evolutionary swarm robotics: genetic diversity, task-allocation and task-switching, in: International Conference on Swarm Intelligence, Springer, 2014, pp. 98–109.
[27] A. Bernard, J.-B. Andr, N. Bredeche, Evolving specialisation in a population of heterogeneous robots: the challenge of bootstrapping and maintaining genotypic polymorphism, Artificial Life 15 (2016) 1–8.
[28] J. Whitacre, A. Bender, Degeneracy: a design principle for achieving robustness and evolvability, Journal of Theoretical Biology 263 (1) (2010) 143–153.
[29] c. Koç, T. Bektaş, O. Jabali, G. Laporte, Thirty years of heterogeneous vehicle routing, European Journal of Operational Research 249 (1) (2016) 1–21.
[30] K. Sundar, S. Rathinam, Algorithms for heterogeneous, multiple depot, multiple unmanned vehicle path planning problems, Journal of Intelligent & Robotic Systems 88 (2017) 513–526.
[31] G. P. Das, T. M. McGinnity, S. A. Coleman, L. Behera, A distributed task allocation algorithm for a multi-robot system in healthcare facilities, Journal of Intelligent & Robotic Systems 80 (1) (2015) 33–58.
[32] N. Sullivan, S. Grainger, B. Cazzolato, Sequential single-item auction improvements for heterogeneous multi-robot routing, Submitted to Journal of Intelligent & Robotic Systems.
[33] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman, Team IHMC's lessons learned from the DARPA robotics challenge trials, Journal of Field Robotics 32 (2) (2015) 192–208.
[34] R. A. Knepper, T. Layton, J. Romanishin, D. Rus, Ikeabot: An autonomous multi-robot coordinated furniture assembly system, in: Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE, 2013, pp. 855–862.
[35] S. Jeon, M. Jang, D. Lee, Y.-J. Cho, J. Kim, J. Lee, Multiple Robots Task Allocation for Cleaning and Delivery, Springer, 2016, book section 1, pp. 195–214.
[36] M. Bollini, S. Tellex, T. Thompson, N. Roy, D. Rus, Interpreting and executing recipes with a cooking robot, in: Experimental Robotics, Springer, 2013, pp. 481–495.
[37] J. Baca, P. Pagala, C. Rossi, M. Ferre, Modular robot systems towards the execution of cooperative tasks in large facilities, Robotics and Autonomous Systems 66 (2015) 159–174.
[38] T. M. Roehr, F. Cordes, F. Kirchner, Reconfigurable integrated multirobot exploration system (RIMRES): heterogeneous modular reconfigurable robots for space exploration, Journal of Field Robotics 31 (1) (2014) 3–34.
[39] H. Ahmadzadeh, E. Masehian, M. Asadpour, Modular robotic systems: characteristics and applications, Journal of Intelligent & Robotic Systems 81 (3-4) (2016) 317–357.
[40] M. S. Emami Taba, Solving traveling salesman problem with a non-complete graph, Masters thesis, School of Computer Science (2010).
[41] R. W. Floyd, Algorithm 97: shortest path, Communications of the ACM 5 (6) (1962) 345.
[42] N. M. Razali, J. Geraghty, Genetic algorithm performance with different selection strategies in solving TSP, in: Proceedings of the world congress on engineering, Vol. 2, 2011, pp. 1134–1139.
[43] D. S. Johnson, Local optimization and the traveling salesman problem, in: International Colloquium on Automata, Languages, and Programming, Springer, 1990, pp. 446–461.
[44] G. Reinelt, TSPLIB-a traveling salesman problem library, ORSA journal on computing 3 (4) (1991) 376–384.

Chapter 5

Analysing Collaborative Localisation Properties

This chapter analyses the conditions which make Collaborative (or Cooperative) Localisation (CL) perform effectively. Prior to this, it was unclear what conditions make CL worthwhile, and what the weaknesses are of the widely available Extended Kalman Filter (EKF). A number of sensor qualities (position accuracy, yaw accuracy, sample rate), communication rates, and number of robots are analysed for both homogeneous and heterogeneous systems. Trends were found in simulation using a popular dataset, and confirmed in hardware-in-the-loop experiments. It is found that CL is less effective in homogeneous systems, systems with very fast inter-robot detections, and systems with minimal access to exteroceptive sensors such as GPS.

It may be useful to read Appendix A before this chapter to contextualise how CL can be used on physical platforms.

Statement of Authorship

Paper Title:	Analysis of Cooperative Localisation Performance Under Varying Sensor Qualities and Communication Rates
Status:	Accepted on 29 September 2018
Details:	Published in *Robotics and Autonomous Systems*, vol 110, pp 73-84, 2018

Principal Author

Name:	Nick Sullivan
Contribution Details:	Performed literature review on algorithms for localising with multiple robots, categorising them by the types of environments where they are used. Discovered the lack of research on how system conditions affect collaborative localisation. Implemented collaborative localisation in simulation and in a hardware-in-the-loop system. Developed and performed tests to analyse how sensor quality and communication properties affects collaborative localisation performance. Parsed and analysed results. Prepared the manuscript and generated all figures.
Contribution Percentage (%):	80
Signature:	Date: 17 Mar, 2019

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

1. the candidate's stated contribution to the publication is accurate (as detailed above);
2. permission is granted for the candidate to include the publication in the thesis; and
3. the sum of all co-author contributions is equal to 100% less the candidates stated contribution.

Name:	Steven Grainger
Contribution Details:	Guided research direction. Supervised work development. Helped generate ideas for tests and edited manuscript.
Signature:	Date: 15 Mar, 2019

Name:	Ben Cazzolato
Contribution Details:	Guided research direction. Supervised work development. Helped generate ideas for tests and edited manuscript.
Signature:	Date: 13 Mar, 2019

Robotics and Autonomous Systems 110 (2018) 73–84

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Analysis of cooperative localisation performance under varying sensor qualities and communication rates

Nick Sullivan *, Steven Grainger, Ben Cazzolato

The University of Adelaide, South Australia 5005, Australia

HIGHLIGHTS

- We alter and analyse parameters affecting Cooperative Localisation (CL).
- Simulations are performed in MATLAB, and validated on hardware-in-the-loop experiments.
- Our findings will help determine the suitability of CL for a system.

ARTICLE INFO

Article history:
Received 7 March 2018
Received in revised form 26 July 2018
Accepted 29 September 2018
Available online xxxx

Keywords:
Cooperative localisation
Multi-robot
Performance analysis
Kalman filter

ABSTRACT

Cooperative Localisation (CL) is a robust technique used to improve localisation accuracy in multi-robot systems. However, there is a lack of research on how CL performs under different conditions. It is unclear *when* CL is worthwhile, and *how* CL performance is affected if the system changes. This information is particularly important for systems with robots that have limited power and processing, which cannot afford to constantly perform CL. This paper investigates CL under varying sensor qualities (position accuracy, yaw accuracy, sample rate), communication rates, and number of robots for both homogeneous and heterogeneous multi-robot systems. Trends are found in MATLAB simulations using the UTIAS dataset, and then validated on Kobuki robots using an OptiTrack-based system. We find that yaw accuracy has a substantial effect on performance, a communication rate that is too fast can be detrimental, and heterogeneous systems are greater candidates for cooperative localisation than homogeneous systems.

1. Introduction

A fundamental challenge for mobile robotics is calculating the position and orientation (pose) of robots within their environment, known as *localisation*. This is necessary for robots to accurately interact with their environment, as well as for interacting with one another. In many industries, multiple robots operate within the same environment, known as *multi-robot systems*. Industries such as agriculture [1], warehouse automation [2], search and rescue [3], environment monitoring [4], healthcare assistance [5], mining [6], transport [7], and assembly [8], are beginning to use mobile robots in everyday operations. To address the localisation problem, robots are typically equipped with two types of sensors. The internal state of robots are measured by *interoceptive* sensors, such as gyroscopes, accelerometers, and wheel encoders. Interoceptive sensors reliably provide data, but have pose errors that accumulates over time. *Exteroceptive* sensors interact with the environment, such as GPS, cameras, and LIDARs. Exteroceptive sensors do not suffer from error accumulation, but they are sensitive to environment conditions. For example, localisation using GPS requires satellite signals, LIDARs require static terrain or recognisable features, and cameras require certain lighting conditions. There exists a lot of research in creating systems that are robust to exteroceptive sensor outages [9,10]. In multi-robot systems, the environment can be measured by multiple robots. The robots can then share their information to improve localisation. This is known as *Cooperative Localisation* (CL).

There are two major areas of research for CL, one is known as *Cooperative Simultaneous Localisation and Mapping* (C-SLAM), where robots independently produce maps of the environment and then share and combine those maps. C-SLAM was a key contributor for the winning team of the MAGIC 2010 competition [11], where robots had to autonomously survey and map a 500 m $\times$ 500 m dynamic urban environment. Communication was not always available, so individual mapping and map fusion was necessary to continue surveillance during communication down-times. C-SLAM has also been used for tasks such as mapping a large area with aerial vehicles [12], localising underwater vehicles to reduce the need for surfacing [13], and to identify and track dynamic targets [14].

C-SLAM can be powerful, but it has requirements that make it unsuitable in certain systems. Firstly, each robot must have SLAM

* Corresponding author.
E-mail address: nicholas.sullivan@adelaide.edu.au (N. Sullivan).

https://doi.org/10.1016/j.robot.2018.09.010
0921-8890/

capabilities. This can inflate the cost of multi-robot systems, as effective SLAM often makes use of high quality sensors such as 3D LIDARs. Each robot must also be capable of processing data quickly, either through on-board processing or communication, and is therefore not suitable for systems with inexpensive processors or unreliable communication. Secondly, SLAM performance is dependent on the type and number of landmarks in the environment [15]. For example, SLAM does not operate well in open areas where there are few recognisable features.

The other major area for CL research involves measuring and communicating inter-robot observations. This differs from C-SLAM in that no map sharing occurs. Robots observe one another, estimate each others position, and communicate their estimates to the observed robots. There are no requirements for how robots localise and perform inter-robot measurements, allowing individual robots to have different sensors, processing capabilities, and internal representations of the environment. There is also less dependence on the environment, as it is able to operate provided robots are able to detect one another. This method is the focus of this paper.

Communicating and processing inter-robot measurements incurs a cost of bandwidth, energy, and processing time. To date, CL papers assume these costs are negligible, and therefore communicate whenever possible. However, there are scenarios where bandwidth and power are not readily supplied, and over-use of these resources can lead to mission failure. It has been shown that CL *can* be used but not *when* it should be used, or what system changes can be made to improve its effectiveness.

We analyse the efficiency and effectiveness of using CL for indoor ground vehicles under varying sensor quality, inter-robot communication rate, and the number of robots. We make use of a commonly used dataset, and then compare these results using real indoor robots.

Section 1 introduces the concept of cooperative localisation and other recent works. Section 2 discusses the CL approach, including the implementation, sensor fusion, and calibration. Section 3 contains the simulation results, showing how the system configuration affects the effectiveness and efficiency of cooperative localisation. Section 4 contains information about the physical system, including hardware information, software flow, and implementation points of note. Section 5 includes results from physical tests, which is compared to simulation. Section 7 discusses the results and their relevance to previous and future research. Section 8 concludes the paper.

1.1. Related works

Cooperative localisation (CL) was first addressed in 1994 [16], where robots periodically 'leap-frogged' past one another. The stationary robots took the role of stationary landmarks, allowing the mobile robots to more accurately measure their own movement. More recently, CL techniques involve communication between the robots. This allows cooperation to be used without constraining robot movement.

An obvious use case for CL is for localisation assistance. Robots with accurate localisation capabilities assist in the localisation of robots with less expensive or broken sensors. This was shown to be beneficial in experiments where ground vehicle leaders localised followers [17]. CL can also be used as a backup localisation method, where robots attempt to communicate when their primary localisation source is unavailable. This was shown to work in a simulation of smart cars moving through tunnels and urban canyons [18]. GPS sensors are unavailable during these areas, so CL was used until GPS signals returned.

CL is not restricted to land, it has been used in air and underwater vehicles as well. A team of simulated underwater vehicles were able to localise themselves by identifying their distance from a surface vehicle [19]. The surface vehicle was equipped with a GPS unit. This provided accurate localisation for all vehicles without the need for high quality sensors in every vehicle. Ground and air vehicles performed CL to improve performance [20], where ground vehicles were equipped with QR codes, and drones had cameras that were able to estimate distance and angles.

Inspired by the original 1994 technique, CL has been used to improve 3D mapping of large buildings [21]. Child robots periodically take the role of stationary landmarks in order to improve the localisation of a parent robot. The parent robot is equipped with a 3D LIDAR, and controls the child robots as a means to maintain a high localisation accuracy, which results in more accurate 3D maps.

There are a number of different ways to implement CL. Communication is often two-way so that communicated messages are beneficial to both robots, and it was found that using range and bearing is more useful than using either individually [22]. While many papers assume that inter-robot measurements will also identify the robot, some have dealt with anonymous detections [23]. There has also been work on network topologies for instances where communication between robots is non-trivial [24].

CL implementations most commonly use a decentralised framework [17,25–27], citing the fragility and lack of scalability of centralised systems, and often use a centralised system as a benchmark. Recent research in this area [17,24,25,28] has used a publicly available multi-robot collaboration dataset known as *Multi-robot Cooperative Localisation And Mapping* (MRCLAM) [29]. This dataset was collected from five ground vehicles in an indoor location equipped with visual markers for inter-robot observations. The dataset contains the raw inter-robot observations, as well as the ground truth position and orientation of each robot as recorded using a 10-camera Vicon motion capture system with millimetre accuracy.

Recent literature has proposed a large number of cooperative localisation algorithms for improving scalability, reliability, and accuracy for different scenarios. Wanasinghe et al. [17] used a *Cubature Kalman Filter* (Gaussian-based particle filters that can intrinsically carry covariances, [30]) to show the improvements that can be made when a subset of robots have superior sensing capabilities. Čurn et al. [27] applied a *Common Past-Invariant Ensemble Kalman Filter* to improve localisation of a number of road vehicles with regions of no GPS coverage. They argue that cooperative localisation is much cheaper to implement than introducing localisation infrastructure such as beacons.

De Silva et al. [25] developed an algorithm that tracks other robots with registration and correction stages. Each robot communicates the inter-robot observations as well as their own velocity. The robots use a standard *Kalman Filter* for fusing interoceptive sensors, and *Covariance Intersection* to incorporate the tracking information. Covariance Intersection is very similar to a Kalman Filter, with the key difference being that Kalman Filters assume all inputs are independent, whereas Covariance Intersection assumes all inputs are dependent.

The reason for these different filters is that cooperative localisation violates the assumption of independence used in a Kalman Filter. If a robot influences another robot's pose estimate, then inter-robot observations from that robot are no longer independent. This circular reasoning problem is called *data incest*. Data incest leads to improper covariance values within a filter, which in turn leads to sub-optimal fusion.

Li et al. [26] developed a method using a *Split Covariance Intersection Filter* that specifically aims to deal with data incest in cooperative localisation. Regular sensory information is fused in an EKF, while the inter-robot information is fused separately by Covariance Intersection.

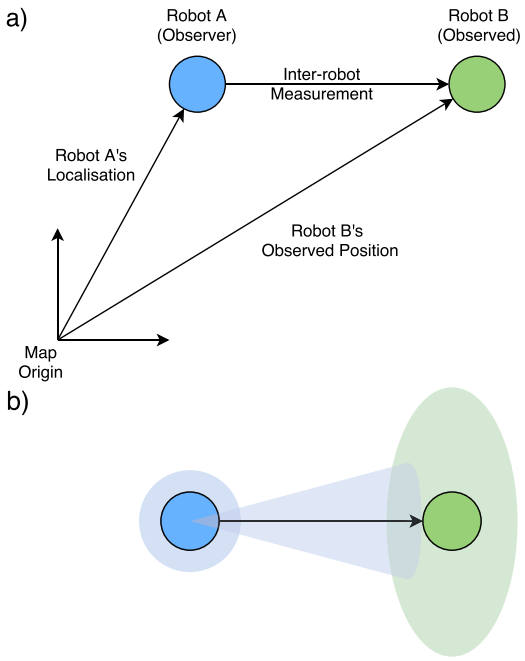


Fig. 1. A graphical depiction on how inter-robot position and uncertainty are calculated. The circles are robot positions, and the shaded regions represent the uncertainty in positions and orientations. (a) The observed robot's position is the vector addition of the observing robot's position and the inter-robot measurement. (b) The uncertainty of the observed robot's position is a function of the observing robot's uncertainty. Yaw uncertainty stretches the uncertainty region, for small yaw uncertainty this can be represented by an ellipse.

2. Approach

The cooperative localisation method used in this paper is as follows. If a robot detects another robot, it measures the range and bearing between them. The inter-robot observation is combined with the observing robot's pose to produce a position estimate of the observed robot. This can be seen graphically in Fig. 1(a). The position uncertainty of the observed robot is calculated from the pose uncertainty of the observing robot and the uncertainty of the inter-robot measurement, as shown graphically in Fig. 1(b). Uncertainty is represented as a covariance matrix. This information is then communicated to the observed robot.

We make use of the MRCLAM dataset [29], which provides the poses and inter-robot observations of five indoor robots. Exteroceptive sensor readings were generated by sampling the true positions of the robots and adding 0-mean Gaussian noise. The sampling rate and standard deviation of the exteroceptive sensors could then be adjusted to any desired value. The interoceptive sensor readings are also generated from true positions, using a noise profile generated from Kobuki robots operating at The University of Adelaide. The Kobuki's use wheel encoders (11.7 ticks/mm) and a gyroscope (1-axis, up to 110 deg/s) to measure the velocity of the robots at a rate of 50 Hz. The noise profile has been calculated to consist of a translation noise ($\mu = -2.1$ cm per metre translation, $\sigma = 3.0$ cm per metre translation) and a rotation noise ($\mu = 0.57°$ per 90° rotation, $\sigma = 0.30°$ per 90° rotation). This kind of noise profile has been used in similar experiments [31].

Inter-robot measurements are provided in the MRCLAM dataset, but have noise that is heavily dependent on the range and bearing between the observed and observing robots. This noise distribution can be seen in Fig. 2. A corrective calibration was applied to produce inter-robot measurement noise that is less dependent on bearing and range between robots. The noise calibration equations were calculated using regression modelling in MATLAB, as shown

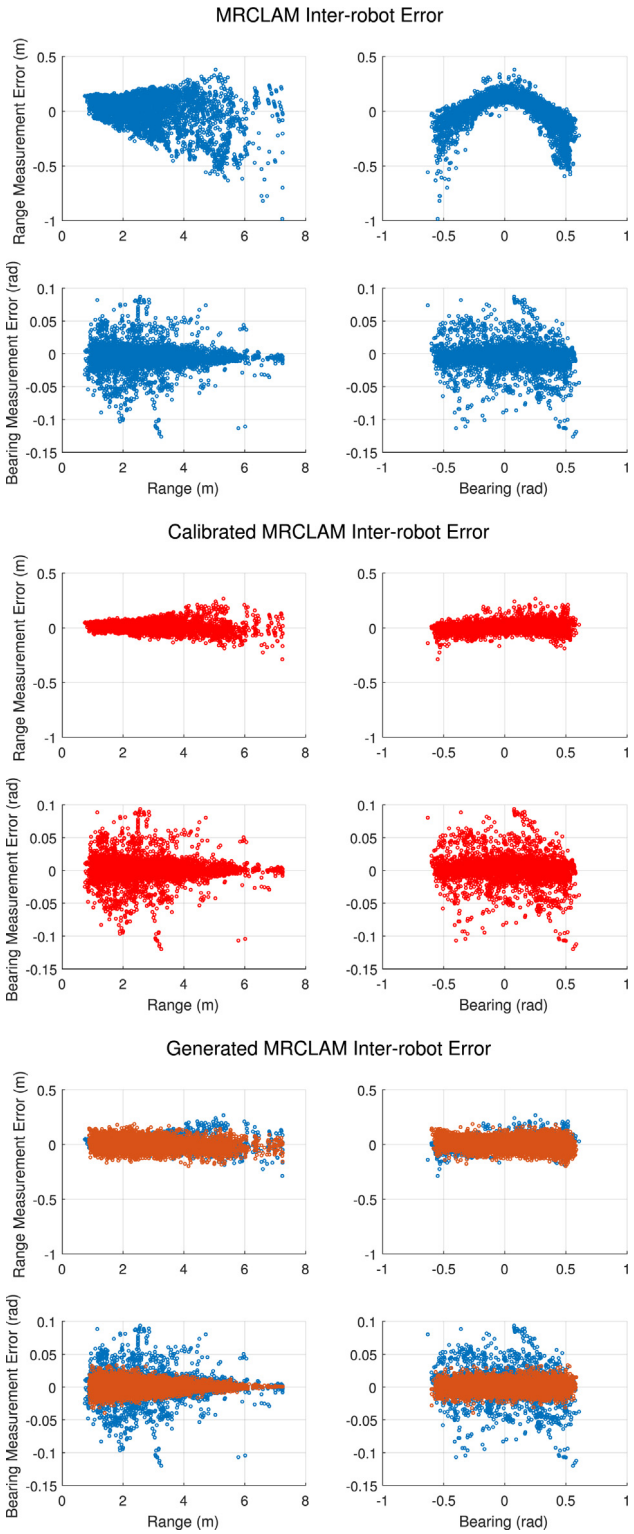


Fig. 2. Inter-robot measurement noise from the MRCLAM dataset. The raw error (top) has noise that is dependent on the range and bearing from the observing robot to the observed robot. Regression modelling was used in MATLAB to reduce the dependence on range and bearing to produce calibrated errors (middle). This error profile was then approximated for use at different rates, overlaying the calibrated distribution from the MRCLAM dataset (bottom).

in Eqs. (1) and (2). The values of the parameters in these equations are listed in Table 1.

$$bearing_{calibrated} = y + 0.0064 \quad (1)$$

Table 1
Parameter values for the Eqs. (1) and (2).

Parameter	Value
x	*range*
y	*bearing*
p_{00}	0.1003
p_{01}	0.0164
p_{02}	0.0960
p_{10}	−0.0191
p_{11}	−0.0014
p_{12}	−0.4977
p_{20}	0.0149
p_{21}	−0.0024
p_{22}	0.0000
p_{30}	−0.0017

$$\begin{aligned} range_{calibrated} &= p_{00} + p_{01} * y + p_{02} * y^2 + p_{10} * x \\ &+ p_{11} * x * y + p_{12} * x * y^2 + p_{20} * x^2 \\ &+ p_{21} * x^2 * y + p_{22} * x^2 * y^2 + p_{30} * x^3 \end{aligned} \quad (2)$$

The calibration produces close to 0-mean error, as shown in Fig. 2. It can be seen that the error is mostly independent of bearing, but that the range has an effect on measurement noise. Detections of nearby robots have more accurate range estimations, but less accurate bearing estimations. Detections of far away robots are the opposite.

For most experiments, this calibrated inter-robot data was used. In other experiments, faster inter-robot data was required, so an error approximation profile was created. The approximated noise profile is shown at the bottom of Fig. 2. This noise profile is able to generate inter-robot measurement noise from true inter-robot information.

We use Extended Kalman Filters (EKF) for non-holonomic vehicles, which are widely used for mobile vehicles. The implementation was taken from a popular package in the Robot Operating System [32] called *robot_localization*. It is well known that cooperative localisation with filters that treat inputs as independent, such as EKF, will suffer from data incest [27]. Nevertheless, we use EKFs for the following reasons:

1. CL can successfully be done with EKFs [18,19,26–28]
2. To explore when data incest becomes problematic
3. To support roboticists wanting to use CL but also want to use readily available EKF implementations
4. To provide a benchmark for comparison with more innovative fusion algorithms

In each test, simulated robots are equipped with an exteroceptive sensor and an interoceptive sensor. The interoceptive sensor consists of wheel encoders that estimate the pose of the robots relative to their start position. The exteroceptive sensor is a GPS that provides position. The data from these sensors are fused into two Extended Kalman Filters per robot. One of the EKFs also fuses inter-robot measurement data. It is therefore performing cooperative localisation, and is called the CL EKF. The other EKF does not fuse inter-robot measurements. It is performing singular localisation, and is called the SL EKF.

At time k, each robot stores a state estimate $\hat{\mathbf{x}}_{k|k}$ and a state covariance $\mathbf{P}_{k|k}$, which contain the estimated pose and velocity, as well as the uncertainty of these estimations. The EKF performs two phases; *prediction* and *update*. The prediction phase calculates the robot state based on its prior state:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} \quad (3)$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k \quad (4)$$

The update phase uses sensor measurements $\mathbf{z}_k$ with covariance $\mathbf{R}_k$ and observation matrix $\mathbf{H}_k$ to improve the state:

$$\text{Innovation:} \quad \tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \quad (5)$$

$$\text{Innovation Covariance:} \quad \mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \quad (6)$$

$$\text{Kalman Gain:} \quad \mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1} \quad (7)$$

$$\text{Updated State:} \quad \hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \quad (8)$$

$$\text{Updated Covariance:} \quad \mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \quad (9)$$

The values of the matrices $\mathbf{F}_k$ and $\mathbf{Q}_k$ can be found in the appendix.

Two scenarios were used for experiments, homogeneous and heterogeneous. In the homogeneous case, every robot was equipped with sensors of the same accuracy and period. In the heterogeneous case, one of the robots had superior sensors. The exact details of the sensors depend on the test, and are discussed in the results section.

3. Results

Cooperative localisation simulations have been performed in MATLAB. Each test is in one of two scenarios: homogeneous, where each robot has the same sensor quality, and heterogeneous, where one of the robots has superior sensing capabilities. The superior robot is known as the parent, and the other robots are known as the children. In every test the robots have the same interoceptive sensors, and only the exteroceptive sensors are altered. The exact exteroceptive sensor specifications are detailed in each section. In every test the robots maintain two filters each, one which fuses local and inter-robot information (CL), and one which only fuses local information (SL).

The positions of the robots, as output by the filters, were compared at each timestamp to the ground truth. From this, a mean position error could be generated for CL and SL cases. All communication between robots is logged in order to calculate the improvement per message sent. The tests involved altering exteroceptive sensor accuracy, exteroceptive sensor rate, inter-robot communication rate, and number of robots.

3.1. Exteroceptive sensor accuracy

Fig. 3 shows the performance improvement by using cooperative localisation under two scenarios. In the homogeneous scenario, every robot is equipped with a GPS sensor providing measurements once per second, with accuracy that was varied. The heterogeneous scenario differs in that the parent robot has high accuracy differential GPS with a constant standard deviation of 10 cm. The localisation errors were averaged over the children robots and plotted in Fig. 3. The CL improvement was calculated by dividing the mean SL position error by the mean CL position error.

It can be seen that it is more effective to use CL when robots have poor localisation. The inter-robot measurement noise becomes less significant when the robots have large localisation errors. Robots with very poor pose estimates benefit strongly from more information, whereas robots with good pose estimates do not benefit as much.

Considering the heterogeneous robots, CL performs better when the children have poor localisation. This is as expected, cooperative localisation can propagate the high accuracy position information to the children. The parent had a SL position error of 0.013 m, which is 3–30 times more accurate than the children. As the level of heterogeneity increases, the overall improvement increases, as seen by the widening gap between the homogeneous and heterogeneous trend-lines.

For the parent, however, the use of CL is less effective as heterogeneity increases. For the rightmost data points, we see that the four children received 40% localisation improvement, and the parent was worsened by 2%. This is a substantial improvement on average, but the improvement is not equally distributed. It is more effective to use CL when robots have different localisation accuracy, but this benefit is one-sided.

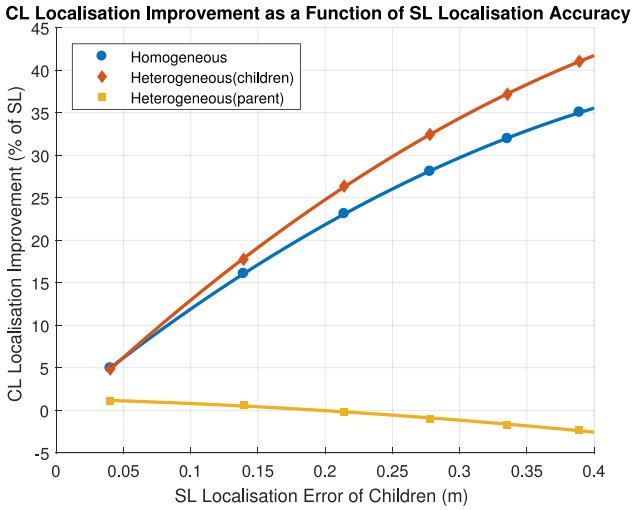


Fig. 3. One parent robot and four children robots localise using cooperative localisation (CL) and singular localisation (SL) in simulation. The child robots had GPSs with a period of one second, with an accuracy that was adjusted from 0.1 to 5 m. The parent robot had the same GPS quality in the homogeneous scenario, and higher accuracy GPS in the heterogeneous scenario.

With perfect information fusion the parent should never worsen their localisation through CL, and is clearly an artefact of an EKF. If using an EKF, it may be beneficial to not fuse CL information on the parent robot at all.

3.2. Exteroceptive sensor period

Fig. 4 illustrates the average localisation improvement when using CL. In these tests, each child robot has a GPS providing position with a standard deviation of 1 m. The period of the sensor is varied between 1 s and the simulation time of 1500 s. In the homogeneous scenario, the parent robot had the same GPS sensor quality as the children. In the heterogeneous scenario, the parent had a differential GPS providing position with a standard deviation of 0.1 m and a period of 1 s.

As the exteroceptive sensor period changes, so too does the singular localisation accuracy. The change in singular localisation accuracy will have an effect on the performance of CL. Therefore the change in CL performance is caused by two factors: the change in sensor period (the independent variable), and the change in accuracy. It is possible to separate these factors using the trendlines in Fig. 3. These are shown in each graph of Fig. 4 as a red line. The difference between the total localisation CL improvement (blue) and the CL improvement due to localisation accuracy (red) is the CL improvement due to exteroceptive period (purple). Through this process, the purple line has been normalised to a period of 1 s.

In Fig. 4, we can see some key differences between the homogeneous and heterogeneous scenarios. For the heterogeneous scenario, CL is more effective for the children robots, i.e. the children are able to leverage the superior sensing capabilities of the parent. The CL improvement due to exteroceptive period (purple) shows a clear improvement as the children's sensors become slower. This same trend is shown for the parent robot. For the homogeneous scenario, however, there is a peak. This indicates that the use of exteroceptive sensors that are very fast or very slow may be inferior candidates for CL using an EKF.

While the heterogeneous system performs better, a system with fewer exteroceptive sensors, could theoretically be more affected by data incest. With fewer sources of independent information, the same information is propagated throughout the system, causing more circular reasoning. We expect that the use of filters

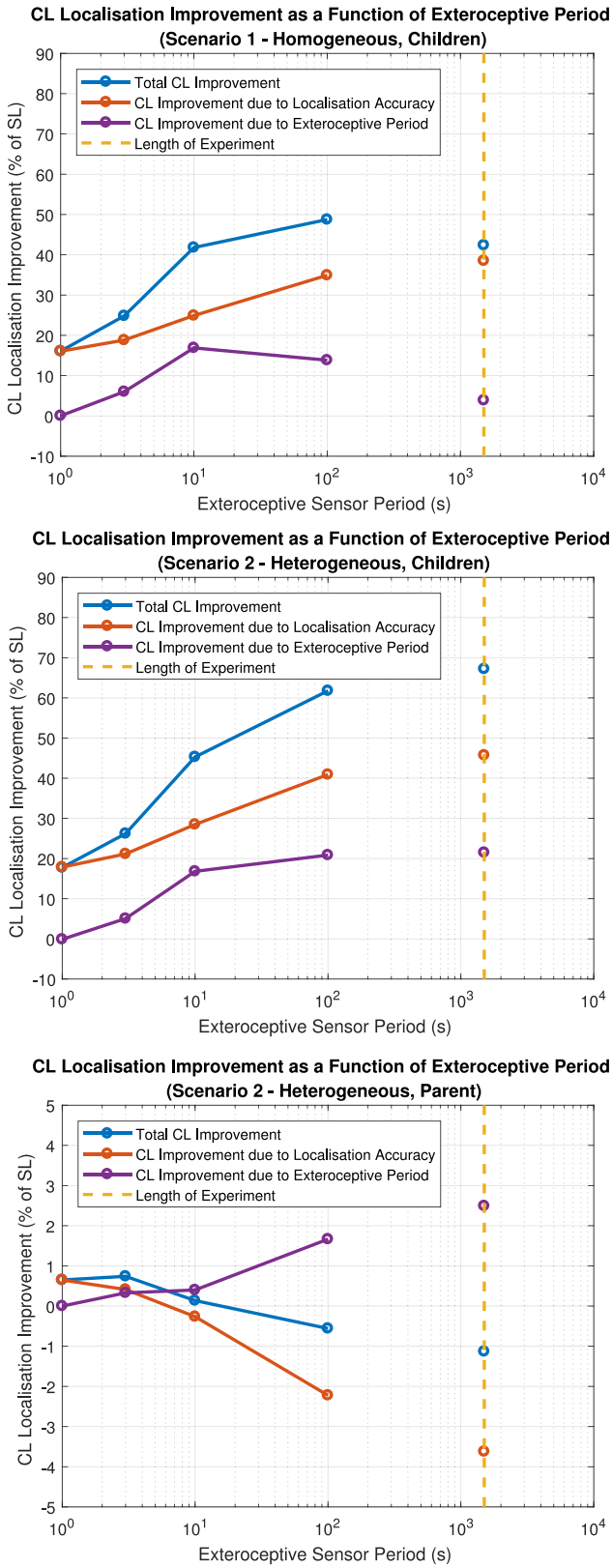


Fig. 4. Four robots (children) had their GPS sensor periods varied from 1 to 1500 s. One robot (parent) either had the same sensor as the children (homogeneous) or a differential GPS with 0.1 m accuracy and 1 s period (heterogeneous). The mean localisation errors for cooperative localisation (CL) and singular localisation (SL) cases were measured. Each graph plots the total localisation improvement from using CL, the CL improvement solely due to localisation accuracy, and the CL improvement due to GPS sensor period. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

that avoid data incest will have greater applicability in systems with few exteroceptive sensors.

We can also see that localisation accuracy has a greater effect than exteroceptive period. Note that there is a gap between 100 and 1000 s because there are insufficient numbers of exteroceptive sensor readings in this region to indicate a clear trend.

3.3. Communication rate

Communication rates have significant effect on CL performance. Fig. 5 shows the localisation improvement when using CL while varying communications rate from 0.5 Hz to 50 Hz. Each child robot is equipped with a GPS with standard deviation of 1 metre and period of 1 s. In the homogeneous scenario, the parent is equipped with the same GPS as the children robots. In the heterogeneous scenario, the parent is equipped with a differential GPS with 0.1 metre accuracy. The results for a homogeneous system are as one might expect, the faster the robots communicate, the more accurate the localisation becomes. This occurs with diminishing returns, so in general it is more effective to send more messages, but more efficient to send fewer messages.

For the children robots in the heterogeneous system, the localisation improves as expected from 0.5 to 10 Hz. But as communications rate increases beyond 10 Hz the position improvement gets worse. This is due to a violated assumption in the Extended Kalman Filter. It expects zero-mean information, but inter-robot messages are biased by the observer. Looking back at the vector addition in Fig. 1, it can be seen that observer localisation error will be added to the inter-robot message. If the inter-robot messages are generated before the observer is able to significantly update its pose estimate, then the inter-robot messages will not have zero-mean error. In addition, the messages are received very quickly, leading to the receiving robot becoming overconfident in the inter-robot messages. The filter will then prioritise interrobot messages over GPS. Effectively this means that for the EKF there is an optimal communication rate that is dependent on the observer's localisation accuracy and update rate. This is an area where state of the art filters would be beneficial, as it is expected that this falloff would not occur if the filter tracked information dependency. For the parent we see that small amounts of communication are beneficial, but quickly become detrimental after 2 Hz. This supports earlier results in this paper that it is advisable to ignore CL information for robots with good localisation in an EKF.

3.4. Yaw accuracy

When using inter-robot measurements that include range and bearing, the bearing accuracy is dependent on the observing robots yaw accuracy, as seen in Fig. 1. Robots had exteroceptive sensors providing both position and orientation, such as from a camera performing Simultaneous Localisation and Mapping (SLAM). The yaw accuracy was varied to find its effect on localisation accuracy. In Fig. 6, we can see that it is substantially more effective to use CL when robots have accurate yaw.

3.5. Number of robots

The number of robots participating in the simulation was varied to determine its effect on localisation performance. Fig. 7 shows the results of up to five homogeneous robots, equipped with GPS units with 1 m standard deviation with 1 s period. The data points involving two, three and four robots were averages of all subsets of the five available robots. Predictably, having more robots improves localisation accuracy. However, the efficiency of CL decreases slowly the more robots are involved. This is because the use of CL becomes less efficient as localisation becomes more accurate, as found earlier.

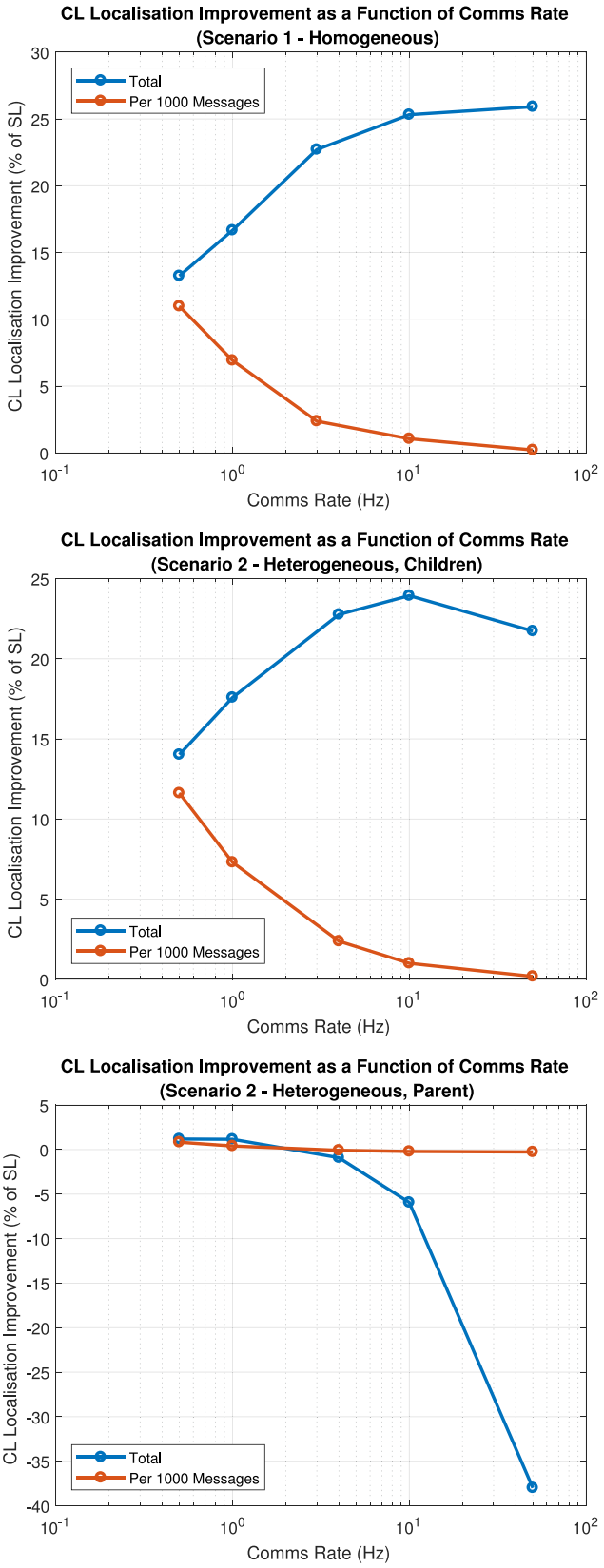


Fig. 5. The communication rate between robots was varied to determine its effect on cooperative localisation (CL) performance. In the homogeneous case, the parent robot had equal GPS quality to four child robots. In the heterogeneous case, the parent had superior GPS. Each graph illustrates the total performance improvement by using CL, as well as the improvement per 1000 messages sent between robots.

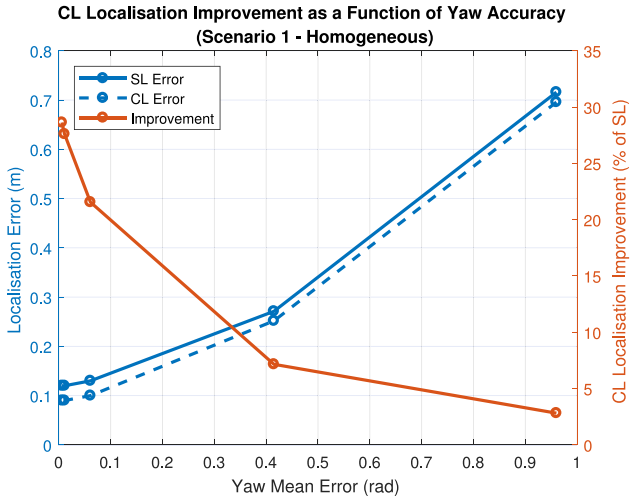


Fig. 6. Yaw accuracy was varied to determine its effect on cooperative localisation (CL) performance over singular localisation (SL).

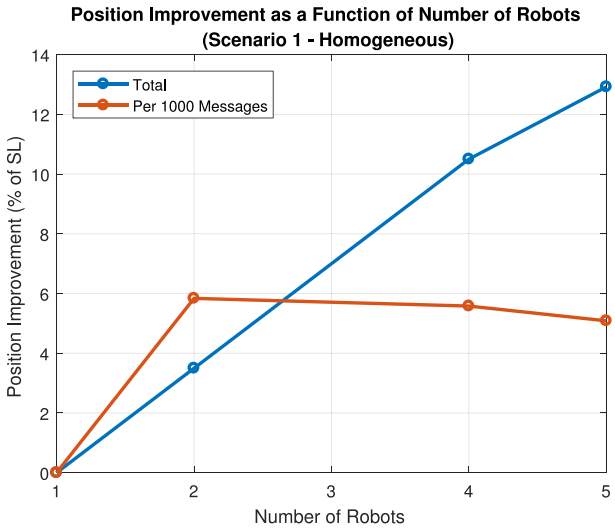


Fig. 7. The number of robots was varied to determine its effect on cooperative localisation (CL) performance.

4. Experimental system

A hardware-in-the-loop system was used to demonstrate that the results found in simulation have relevance to the real world. The physical system uses YujinRobot Kobukis (Fig. 8) equipped with controllers in the form of Raspberry Pi 3 Model B's running Robot Operating System (ROS). This system is an example of low-cost hardware that would not be able to perform cooperative localisation all the time, as its limited processing power would be needed for other duties.

ROS is a widely used robotics repository and middleware. It provides transparent message transport between and within robots. ROS uses a series of nodes that perform specific functions. These packages can be downloaded from the ROS repository, or custom made. Many robots, including Kobukis, provide ROS interoperability as standard. The software flow and ROS package usage is detailed in Section 4.1.

The Kobukis are mounted with a series of reflective markers in unique orientations. The poses and ID's of the Kobuki's are tracked at 100 Hz in real-time using a 6-camera OptiTrack camera system. The system can be seen in Fig. 9. The Kobuki's move in a set pattern over 300 s. Each sensor configuration was tested five times to produce a distribution of results.

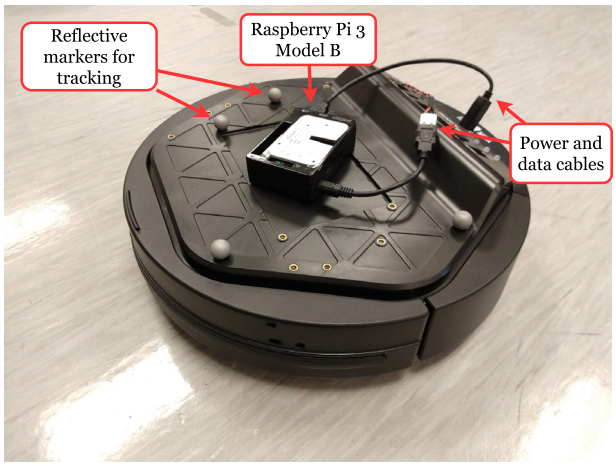


Fig. 8. YujinRobot Kobuki's were controlled by Raspberry Pi's running ROS. Each Kobuki is equipped with wheel encoders (11.7 ticks/mm) and a 1-axis gyroscope.

4.1. Software

A number of components were used in conjunction to successfully perform cooperative localisation, as shown in Fig. 10. The robots have a set of reflective markers mounted on the top to determine their poses. These markers are uniquely distributed to identify each robot. This information is streamed via the Virtual Reality Networking Protocol (VRPN) using Motive software for robot pose tracking. This protocol is used to communicate the poses of all tracked objects. The poses are then read by the central computer, which converts the VRPN pose (Y-up convention, standard in computer vision) into the ROS system (Z-up convention). The central computer then performs the following tasks:

1. Logs the true poses of the robots
2. Sends exteroceptive sensor messages to robots that contain:
   - Robot pose such as from GPS or SLAM, produced by adding Gaussian noise to the true pose
   - Distance and bearings to other robots (if within viewing range), with noise added using the function outlined in Section 2
3. Sends control messages to drive the robots in a pre-determ-ined path for repeatable tests.

The robot controllers (Raspberry Pi 3 Model B's) receive the exteroceptive sensor information, and parse it into the individual global and interrobot pose information. Global pose information represents data as from GPS or SLAM, as opposed to local information from the wheel encoders and gyroscope. The global poses are fused into the two Extended Kalman Filters, implemented in the publicly available *robot_localization* package in ROS. One fuses all exteroceptive and interoceptive sensor information (Cooperative Localisation EKF), and the other fuses the same but without interrobot measurements (Single Localisation EKF). The logged outputs of the two filters can then be compared to determine how cooperative localisation affects localisation. The local poses fused into the filters come from the Kobuki's on-board encoders and gyroscope, using a standard Kobuki interface node *kobuki_node*.

The output of the CL EKF is combined using vector addition with interrobot measurements. This process happens in the Interrobot Client node. The resulting tracked position is sent via Wi-fi to the

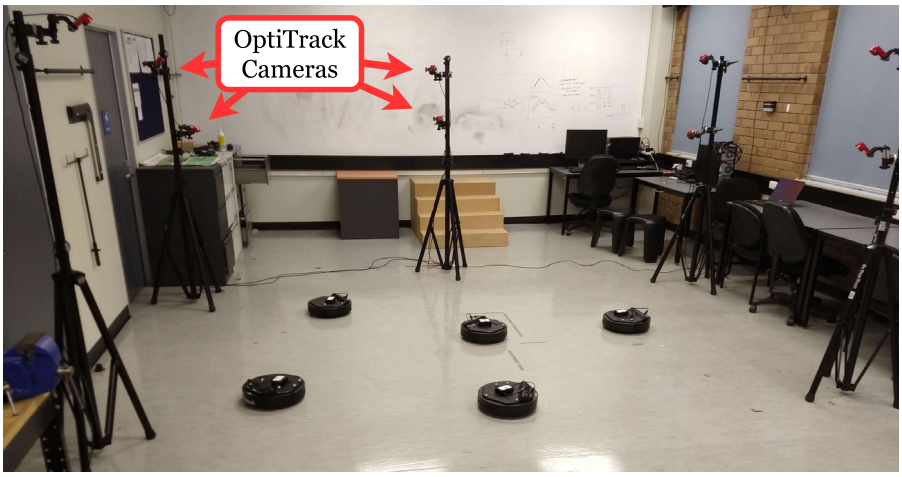


Fig. 9. The experimental setup. OptiTrack cameras track reflective markers to determine the position and orientation of the Kobuki robots.

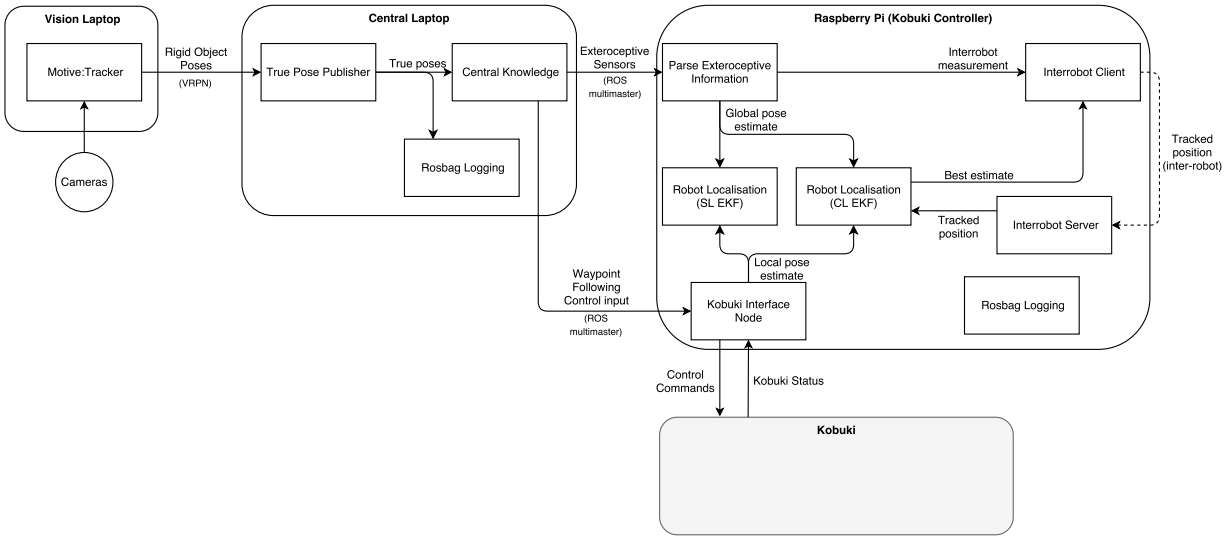


Fig. 10. The software flow for the real system of Kobuki robots. Each Kobuki is equipped with visual markers to be detected by vision tracking cameras. The robot poses are streamed to the central laptop. The central laptop logs the position, adds noise, and sends this information to the robots. Robot controllers fuse sensory information and communicate with one another. They also interact with the Kobuki for on-board sensory information and movement.

target robot, which receives and fuses the tracked position into their CL EKF. When a robot receives the tracked position, it responds with its own position estimate, allowing both robots to gain new position information based on each inter-robot measurement.

4.2. Implementation points of note

The standard operating procedure for ROS is to have a single ROS Master. All nodes communicate to the Master to form topic-based links to other nodes. Once the link is established, communication occurs directly between the linked nodes using TCP. It was found, however, that each robot used a large amount of bandwidth, causing excessive delays for critical information. This was due to the standard ROS topic *tf*, which contains high frequency transformation information between hardware components (e.g. the position of the left wheel relative to the base). This topic is shared by all parties, meaning every robot sent high frequency unnecessary information to all the other robots. To alleviate this, we made use of the *multimaster_fkie* package. This package allows the central computer and every robot controller to use their own ROS master, with specified topics being dynamically synchronised between the ROS environments. The transformation topic *tf* could then be excluded from synchronisation, significantly reducing bandwidth usage.

The robots were on a local network, so they could not use the internet as a source of timing synchronisation. In lieu of this, made use of *chrony*. Each robot then used the central laptop as the time server, periodically polling and adjusting clock times as necessary. Without this, the Raspberry Pi robot controllers would default to times that were days out of sync.

There was also an issue of interrobot message time delay. In simulation, the delay from observing a robot to fusing the estimate is 1 timestep (10 ms). In the real system, however, the time it takes to perform an interrobot measurement has an observable impact. Not taking this into account led to all interrobot messages lagging slightly behind the true position. This effect compounded upon itself: robot pose estimates are affected by a lagged interrobot measurement, so future interrobot position estimates become even more lagged. This significantly negatively impacted the performance of cooperative localisation. For our system, the time delay was measured to be ∼70 ms. All interrobot measurements were past-dated by this time, and the problems were no longer an issue.

5. Experimental results

The tests in simulation, detailed in Section 3, were repeated for the hardware-in-the-loop system. This was done to determine if physical implementation had significant effects on the results found in simulation. Elements such as time delay, data loss, and lack of synchronisation could have pertinent effects on the performance on cooperative localisation. While the experimental results are only for one physical system, they are not intended as a rigorous investigation in how these elements affect cooperative localisation. Rather, they are used to validate the simulation data.

Four Kobukis were moved in a set path five times for each experiment. The experiments lasted for 300 s. As outlined in Section 3, exteroceptive sensing was simulated so that it could be altered to custom rates and accuracies. Interoceptive sensors, however, were performed by the physical robot, and were never adjusted.

The experiment details are listed in each section, and will be compared with the equivalent simulation experiments.

Note that the experimental robots are not traversing the same paths as the simulated robots due to insufficient space, so the numbers should not be directly compared. However, we believe the trends remain valid.

5.1. Exteroceptive sensor accuracy

The graph in Fig. 11 shows the change in position error by using cooperative localisation. Two scenarios are explored. Three child robots were equipped with GPS sensors that provide measurements once per second with an accuracy that was varied for each test. In the homogeneous scenario, the parent robot had identical sensing capabilities. In the heterogeneous scenario, the parent had a superior GPS with standard deviation of 0.1 m.

Data has been fitted with a line, as there is not enough precision to determine the exact relationship. It can clearly be seen that CL is more efficient when position error is larger. We can also see that the children robots are benefited by a parent with high accuracy sensing, but the parent is affected to a much lower extent. One difference between these results and those found in simulation is that the parent robot always has, on average, a non-negative localisation improvement with CL. The difference in slope between simulation and experiment is quite small, and may disappear with more samples.

5.2. Exteroceptive sensor period

Exteroceptive sensor accuracy was held constant, and the period was adjusted. The localisation error and improvement is shown in Fig. 12. The children robots each have a GPS providing position with a standard deviation of 0.1 m. The period of the sensors is varied between 1 s and the run time of 300 s. Each period was performed five times. The total CL improvement (blue) improves as the period is larger, then drops significantly when the period is as long as the experiment for the homogeneous scenario. Whereas for the heterogeneous scenario, the same drop-off does not occur. This closely matches the results for simulation found in Section 3.2.

Considering the CL improvement due to exteroceptive period (purple), we can see that it decreases as the exteroceptive period increases. For the homogeneous scenario, there is a large dip when the exteroceptive period is as large as the length of the experiment. This dip does not occur for the heterogeneous scenario. These results are similar to those found in simulation in Section 3.2. A difference is that the hardware-in-the-loop experiments indicate that an increased exteroceptive period may have negative or negligible effect on CL performance, whereas in simulation it had a distinct positive improvement. This could be due to the effect of time delays, EKF tuning values, or non-linearities not accounted for by the simulation.

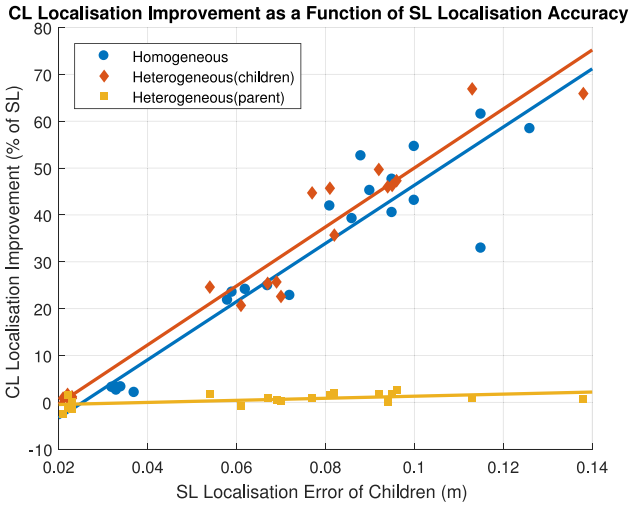


Fig. 11. One parent robot and four children robots localise using cooperative localisation (CL) and singular localisation (SL) using hardware-in-the-loop simulation. The child robots had GPSs with a period of one second, with an accuracy that was adjusted from 0.1 to 4 m. The parent robot had the same GPS quality in the homogeneous scenario, and higher accuracy GPS in the heterogeneous scenario.

Limited analysis can be done of the parent robot, as the uncertainty ranges are larger than the localisation improvement. What we can say is that the impact on the parent is far smaller than that of the children, which is also the case for the simulation.

5.3. Communication rate

The rate of communication between robots was adjusted between 0.1 and 15 Hz. At frequencies above 15 Hz, the robots were unable to process all information in real-time. This caused sensor readings to be dropped, producing increased localisation error for CL and SL, as they were both executed on the same processor for each robot. The results are shown in Fig. 13. Both the homogeneous and heterogeneous cases show CL improvement as communication rate increases, along with decreased CL efficiency. They both reach a limit, at which point increasing communications rate provides no benefit. This matches the results found in simulation, detailed in Section 3.3. It is suspected that higher frequencies for the heterogeneous case would display a fall-off as shown in simulation, but the system was unable to reach the required frequencies.

5.4. Yaw accuracy

Robots were equipped with exteroceptive sensors providing 1 m position error and orientation error adjusted between 1 and 40 degrees. The robots also had orientation information from onboard encoders and gyroscopes, with an error profile of approximately $\mu = 0.57°$ per 90° rotation, $\sigma = 0.30°$ per 90° rotation. The CL localisation improvement can be seen in Fig. 14. As yaw error increases, the use of CL becomes less effective. This matches results seen in simulation in Section 3.4.

5.5. Number of robots

Experiments were conducted with a differing number of robots. In the experimental system, the number of robots was adjusted between 1 and 4. The robots were given exteroceptive sensor information with 1 m accuracy at a period of 1 s Fig. 15 shows the localisation improvement when using CL. The improvement increases as more robots are used, but the efficiency decreases. This closely matches the results found in simulation.

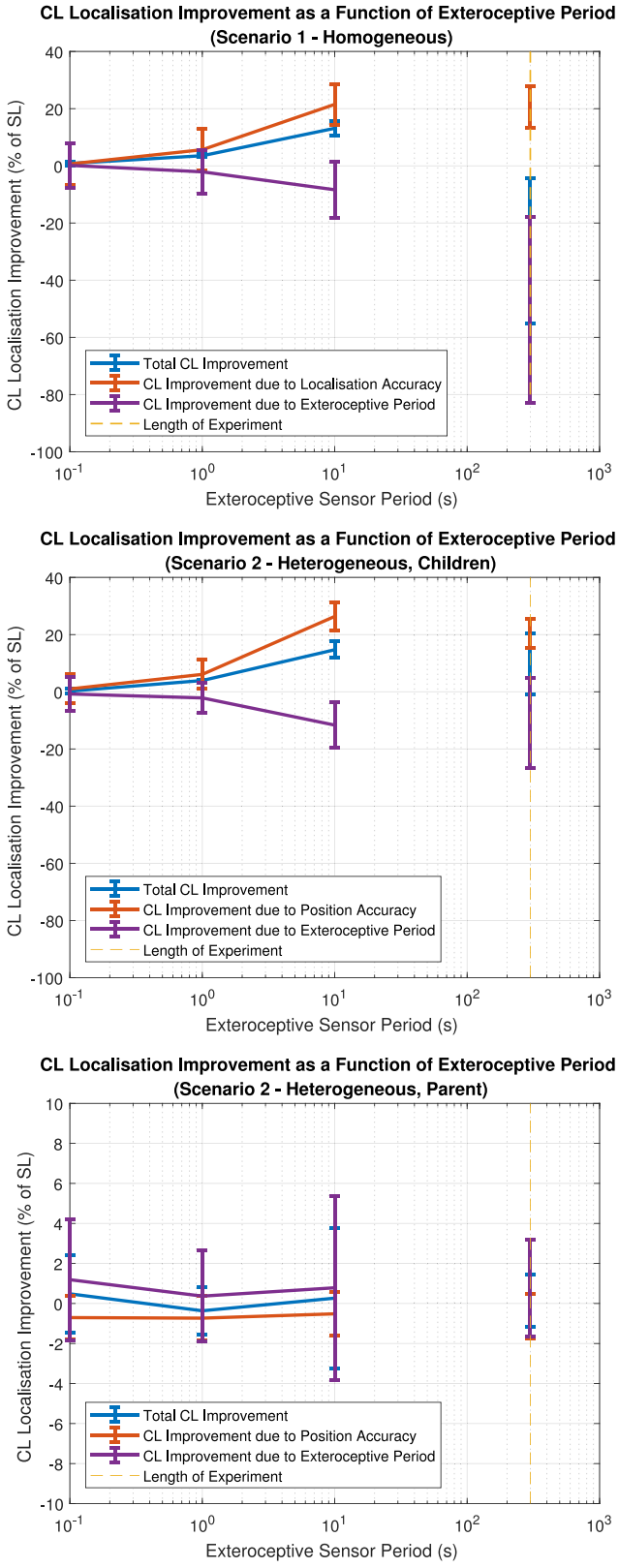


Fig. 12. Three robots (children) had their GPS sensor periods varied from 1 to 300 s. One robot (parent) either had the same sensor as the children (homogeneous) or a superior GPS (heterogeneous). The mean position errors for cooperative localisation (CL) and singular localisation (SL) cases were measured. Each graph plots the total localisation improvement from using CL, the improvement solely due to localisation accuracy, and the CL improvement due to GPS period.

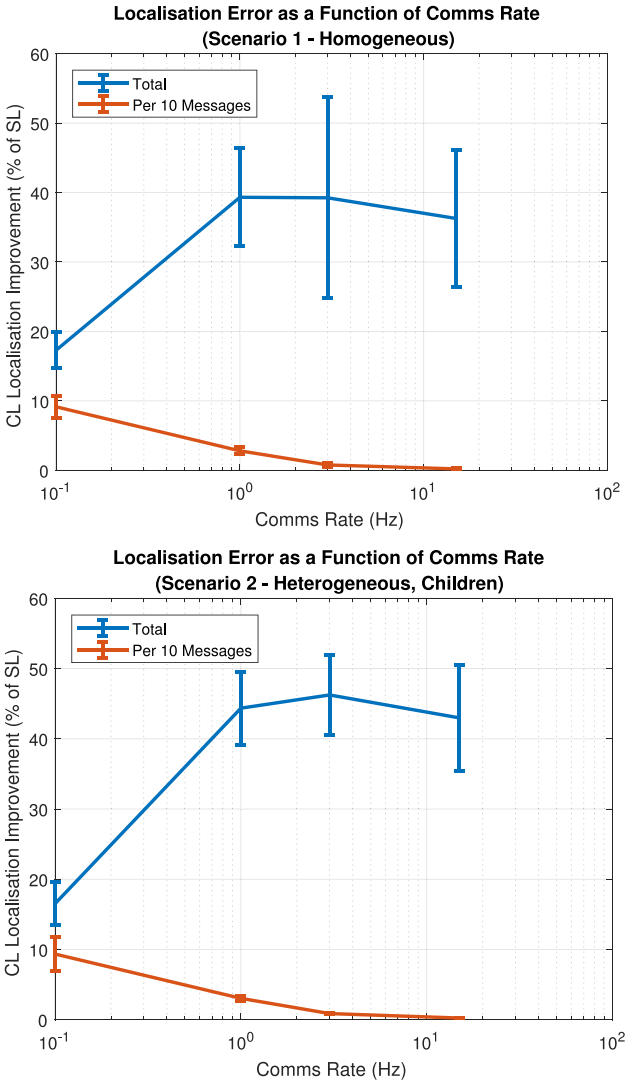


Fig. 13. The communication rate between robots was varied to determine its effect on cooperative localisation (CL) performance for a hardware-in-the-loop simulation. In the homogeneous case, the parent robot had equal GPS quality to three children robots. In the heterogeneous case, the parent had superior GPS. Each graph illustrates the total performance improvement by using CL, as well as the improvement per 10 messages sent between robots.

6. Multivariate performance

A function has been created to estimate the predicted cooperative localisation improvement based on the input parameters. This was performed using MATLAB's curve fitting toolbox.

$$y_{sa} = 1.03x_{sa}^{0.4889} \tag{10}$$

$$y_{sp} = -0.11\log(x_{sp})^2 + 0.992\log(x_{sp}) \tag{11}$$

$$y_{cr} = -9.51x_{cr}^{-0.01594} + 10.59 \tag{12}$$

$$y_{ya} = -1.614x_{ya}^{0.259} + 2.25 \tag{13}$$

$$y_{nr} = 0.364x_{nr}^{0.7634} - 0.37 \tag{14}$$

$$y_{total} = y_{sa}y_{sp}y_{cr}y_{ya}y_{nr} \tag{15}$$

where x_{sa} is the exteroceptive sensor accuracy (m), x_{sp} is the exteroceptive sensor period (s), x_{cr} is the communication rate (Hz), x_{ya} is the yaw accuracy (rad), and x_{nr} is the number of robots. The resulting value y_{total} is the average relative improvement in position accuracy when using cooperative localisation, normalised to 15%,

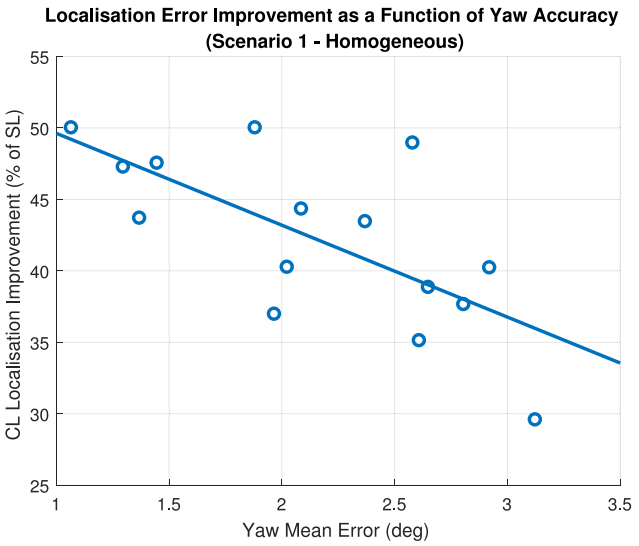


Fig. 14. Yaw accuracy was varied to determine its effect on cooperative localisation (CL) performance over singular localisation (SL) using hardware-in-the-loop simulation.

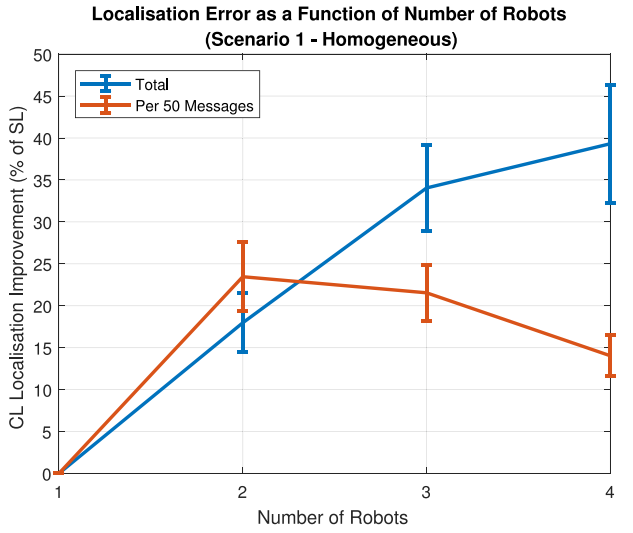


Fig. 15. Experimental result for 300 s using 1–4 robots. The mean position errors for cooperative localisation (CL) and singular localisation (SL) cases were measured, from which the CL improvement percentages were calculated.

i.e., a y_{total} of 1 would have an expected localisation improvement of 15%, and a value of 2 would have an expected localisation improvement of 30%.

Similarly, the further improvement obtained by using a single robot with far greater localisation capabilities, h, is as follows:

$$h_{sa} = -0.088x_{sa}^{-0.4011} + 1.041 \tag{16}$$

$$h_{sp} = 0.0219\log(x_{sp})^2 - 0.071\log(x_{sp}) + 0.9296 \tag{17}$$

$$h_{cr} = -0.08x_{cr}^{0.336} + 0.9755 \tag{18}$$

$$h_{ya} = 0.6981x_{ya} + 1.0475 \tag{19}$$

$$h_{total} = h_{sa}h_{sp}h_{cr}h_{ya} \tag{20}$$

7. Discussion

The simulation and hardware-in-the-loop results match what has been observed (and sometimes assumed) in literature, while also providing new data which have not previously been analysed. We highlight results and discuss their relevance to previous literature and future research.

The use of cooperative localisation is more effective when the robots have poor localisation. While most literature on this subject assumes this is the case (the authors could not find research on CL in systems where localisation is already very accurate), our results match the available analysis of this trend [17]. We also demonstrate that accurate yaw is of particular importance for effective use of CL.

CL is more effective in heterogeneous systems, but the benefit is greater for the robots with poor localisation. CL algorithms have previously been designed and implemented for very heterogeneous cases, where children robots have poor (or non-existent) localisation and parents have very precise localisation [17,19,28]. In these cases, CL has been implemented as one-way, assuming that CL will have little benefit to the parents. Our results validate these assumptions, illustrating that extending CL to be two-way offers little benefit in very heterogeneous circumstances, and can even be detrimental when using an EKF.

Using more robots improves the effectiveness of CL, but at diminishing efficiency. Our results match data found in other studies [18,22,23,31], and suggests that CL in large swarms may not show the level of improvement that some might expect [31].

EKF can be used to perform CL, but will suffer from data incest. Previous works have used EKF to perform CL successfully [18,19,26–28], and have compared EKF performance with newer state-of-the-art filters [26–28]. Based on our results, we expect that these filters will show particularly good improvements in systems with fast inter-robot communication, and in systems with slow exteroceptive sensors.

8. Conclusion

This paper was written to help future roboticists design systems using cooperative localisation. We have profiled its properties in simulation and through hardware-in-the-loop experiments in order to better understand the conditions where CL is effective and efficient. We have indicated where the commonly available EKF is limited, and hence where state-of-the-art filters may be required.

These results provide information for systems where the power and processing costs for performing CL need to be considered. Rather than constantly performing CL, which drains power and processor availability, CL could be activated during opportunities of high efficiency.

Acknowledgements

The Commonwealth of Australia (represented by the Defence Science and Technology Group) supported this research through a Defence Science Partnerships agreement.

Appendix A

The EKF state contains the robot's pose (x, y, yaw) in metres and radians, and its velocity (forward, sideways, angular) in metres per second and radians per second. The filter is discrete, and uses the time between iterations dt. For simulations, $dt = 0.01$ s. For shorthand, we let $c_y = \cos(yaw)$ and $s_y = \sin(yaw)$.

$$\hat{\mathbf{x}} = \begin{bmatrix} pos_x \\ pos_y \\ pos_{yaw} \\ vel_x \\ vel_y \\ vel_{yaw} \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} 0.05 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.05 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.06 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.025 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.025 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.02 \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & dt(-vel_x s_y - vel_y c_y) & dt(c_y) & dt(-s_y) & 0 \\ 0 & 1 & dt(-vel_y s_y + vel_x c_y) & dt(s_y) & dt(c_y) & 0 \\ 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2018.09.010.

References

[1] Luis Emmi, Mariano Gonzalez-de Soto, Gonzalo Pajares, Pablo Gonzalez-de Santos, New trends in robotics for agriculture: integration and assessment of a real fleet of robots, Sci. World J. 2014 (2014) 1–21.

[2] Peter R. Wurman, Raffaello. D'Andrea, Mick Mountz, Coordinating hundreds of cooperative, autonomous vehicles in warehouses, AI Mag. 29 (1) (2008) 9.

[3] Alfredo Martins, Bruno Ferreira, Hugo Ferreira, Guilherme Amaral, Rui Almeida, Filipe Silva, Multiple robot operations for maritime search and rescue in euRathlon 2015 competition, in: OCEANS, IEEE, 2016, pp. 1–7.

[4] Katsuaki Tanaka, Hiroyuki Ishii, Shinichi Kinoshita, Qing Shi, Hikaru Sugita, Satoshi Okabayashi, Yusuke Sugahara, Atsuo Takanishi, Design of operating software and electrical system of mobile robot for environmental monitoring, in: Robotics and Biomimetics, ROBIO, IEEE, 2014, pp. 1763–1768.

[5] Min Chen, Yujun Ma, Sana Ullah, Wei Cai, Enmin Song, Rochas: Robotics and cloud-assisted healthcare system for empty nester, in: Proceedings of the 8th International Conference on Body Area Networks, ICST, Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2013, pp. 217–220.

[6] Chris Brown, Autonomous vehicle technology in mining, Eng. Min. J. 213 (1) (2012) 30.

[7] Kichun Jo, Junsoo Kim, Dongchul Kim, Chulhoon Jang, Myoungho Sunwoo, Development of autonomous car-part I: distributed system architecture and development process, IEEE Trans. Ind. Electron. 61 (12) (2014) 7131–7140.

[8] Ross A. Knepper, Todd Layton, John Romanishin, Daniela Rus, Ikeabot: An autonomous multi-robot coordinated furniture assembly system, in: Robotics and Automation, ICRA, IEEE, 2013, pp. 855–862.

[9] Xiyuan Chen, Chong Shen, Wei-bin Zhang, Masayoshi Tomizuka, Yuan Xu, Kuanlin Chiu, Novel hybrid of strong tracking Kalman filter and wavelet neural network for GPS/INS during gps outages, Measurement 46 (10) (2013) 3847–3854.

[10] Xi-Yuan Chen, Jing Yu, Xue-Fen Zhu, Theoretical analysis and application of Kalman filters for ultra-tight global position system/inertial navigation system integration, Trans. Inst. Meas. Control 34 (5) (2012) 648–662.

[11] Edwin Olson, Johannes Strom, Ryan Morton, Andrew Richardson, Pradeep Ranganathan, Robert Goeddel, Mihai Bulic, Jacob Crossman, Bob Marinier, Progress toward multirobot reconnaissance and the MAGIC 2010 competition, J. Field Robot. 29 (5) (2012) 762–792.

[12] Christian Forster, Simon Lynen, Laurent Kneip, Davide Scaramuzza, Collaborative monocular SLAM with multiple micro aerial vehicles, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 3962–3970.

[13] Liam Paull, Guoquan Huang, Mae Seto, John J. Leonard, Communication-constrained multi-AUV cooperative SLAM, in: Robotics and Automation, ICRA, 2015 IEEE International Conference on, IEEE, 2015, pp. 509–516.

[14] Diluka Moratuwage, Ba-Ngu Vo, Danwei Wang, Collaborative multi-vehicle slam with moving object tracking, in: Robotics and Automation, ICRA, 2013 IEEE International Conference on, IEEE, 2013, pp. 5702–5708.

[15] Gregory Dudek, Michael Jenkin, Computational Principles of Mobile Robotics, Cambridge University Press, 2010.

[16] Ryo Kurazume, Shigemi Nagata, Shigeo Hirose, Cooperative positioning with multiple robots, in: IEEE International Conference on Robotics and Automation, IEEE, 1994, pp. 1250–1257.

[17] Thumeera R. Wanasinghe, George K.I. Mann, Raymond G. Gosine, Distributed leader-assistive localization method for a heterogeneous multirobotic system, IEEE Trans. Autom. Sci. Eng. 12 (3) (2015) 795–809.

[18] Mariam Elazab, Aboelmagd Noureldin, Hossam S. Hassanein, Integrated cooperative localization for connected vehicles in urban canyons, in: 2015 IEEE Global Communications Conference, GLOBECOM, IEEE, 2015, pp. 1–6.

[19] Benedetto Allotta, Riccardo Costanzi, Enrico Meli, L. Pugi, Alessandro Ridolfi, Gregorio Vettori, Cooperative localization of a team of AUVs by a tetrahedral configuration, Robot. Auton. Syst. 62 (8) (2014) 1228–1237.

[20] Louis G. Clift, Adrian F. Clark, Determining positions and distances using collaborative robots, in: Computer Science and Electronic Engineering Conference, CEEC, IEEE, 2015, pp. 189–194.

[21] Ryo Kurazume, Souichiro Oshima, Shingo Nagakura, Yongjin Jeong, Yumi Iwashita, Automatic large-scale three dimensional modeling using cooperative multiple robots, Comput. Vis. Image Understand. 0 (2016) 1–18.

[22] Ioannis M. Rekleitis, Gregory Dudek, Evangelos E. Milios, Multi-robot cooperative localization: a study of trade-offs between efficiency and accuracy, in: Intelligent Robots and Systems, 2002 IEEE/RSJ International Conference on, vol. 3, IEEE, 2002, pp. 2690–2695.

[23] Abdulmuttalib T. Rashid, Mattia Frasca, Abduladhem A. Ali, Alessandro Rizzo, Luigi Fortuna, Multi-robot localization and orientation estimation using robotic cluster matching algorithm, Robot. Auton. Syst. 63 (2015) 108–121.

[24] Leigang Wang, Tao Zhang, Feifei Gao, Distributed cooperative localization with lower communication path requirements, Robot. Auton. Syst. 79 (2016) 26–39.

[25] Oscar De Silva, George K.I. Mann, Raymond G. Gosine, Efficient distributed multi-robot localization: A target tracking inspired design, in: 2015 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2015, pp. 434–439.

[26] Hao Li, Fawzi Nashashibi, Ming Yang, Split covariance intersection filter: Theory and its application to vehicle localization, IEEE Trans. Intell. Transp. Syst. 14 (4) (2013) 1860–1871.

[27] Jan Curn, Dan Marinescu, Niall O'Hara, Vinny Cahill, Data incest in cooperative localisation with the common past-invariant ensemble Kalman filter, in: 16th International Conference on Information Fusion, FUSION, IEEE, 2013, pp. 68–76.

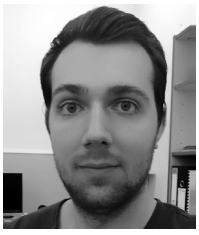
[28] Mohamed W. Mehrez, George K.I. Mann, Raymond G. Gosine, An optimization based approach for relative localization and relative tracking control in multi-robot systems, J. Intell. Robot. Syst. 83 (2016) 1–24.

[29] Keith Y.K. Leung, Yoni Halpern, Timothy D. Barfoot, Hugh H.T. Liu, The UTIAS multi-robot cooperative localization and mapping dataset, Int. J. Robot. Res. 30 (8) (2011) 969–974.

[30] Bingbo Cui, Xiyuan Chen, Xinhua Tang, Improved cubature Kalman filter for GNSS/INS based on transformation of posterior sigma-points error, IEEE Trans. Signal Process. 65 (11) (2017) 2975–2987.

[31] Frank E. Schneider, Dennis Wildermuth, Influences of the robot group size on cooperative multi-robot localisation analysis and experimental validation, Robot. Auton. Syst. 60 (11) (2012) 1421–1428.

[32] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y. Ng, ROS: an open-source robot operating system, in: ICRA Workshop on Open Source Software, vol. 3, 2009, p. 5.



Nick Sullivan received his bachelor degrees in mechatronics engineering and computer science from the University of Adelaide, Australia, in 2016. He is currently working toward the Ph.D. degree in robotics at the University of Adelaide. His research interests include multi-robot task allocation, unmanned ground vehicles, and machine learning.

Steven Grainger obtained his Ph.D. on the control of electric drives from Glasgow Caledonian University, Scotland and holds B.E. degrees in computing and electronic engineering. He is a lecturer in Control and Embedded Systems at the University of Adelaides School of Mechanical Engineering. His current research interests include nanopositioning systems and autonomous vehicles.

Ben Cazzolato received his B.E. in mechanical engineering at the University of Adelaide, Australia, in 1990. At the same university, he received his Ph.D. in the field of active control for sound transmission in 1998. He is currently a professor at the University of Adelaide, teaching and researching in the fields of dynamics and control. Current research interests include modelling of complex electro-mechanical systems, control of unstable vehicles, active control and nano-positioning.

Chapter 6

Task Allocation with Collaborative Localisation

This chapter presents and analyses a new algorithm that allocates tasks to multiple robots while ensuring superior localisation using collaborative localisation techniques. The task allocation constraint requires robots remain within a sufficient distance of one another for visual connectivity, but can equally be used to guarantee network connectivity for communication. Previous algorithms do not guarantee connectivity in environments with obstacles, and are less scalable than the presented algorithm.

A hardware platform was used to experimentally validate the results. This is discussed in detail in Appendix A, presented as a conference paper presented to the Australasian Conference on Robotics and Automation (ACRA 2018). The results are compared to another algorithm that solves a very similar problem, also written by the author of this thesis, found in Appendix B, and was presented to the International Conference of Control, Automation, Robotics and Vision (ICARCV 2018).

Statement of Authorship

Paper Title:	Formation-Based Multi-Robot Routing with Inter-Robot Distance Constraints
Status:	Submitted for publication
Details:	Submitted to *IEEE Transactions on Automation Science and Engineering*, 9 Nov 2018

Principal Author

Name:	Nick Sullivan
Contribution Details:	Performed literature review on algorithms for allocating tasks to robots while maintaining distance between the robots, separating them by their consideration of multiple tasks, speed, and connectivity guarantees. Implemented algorithms from literature, discovering the violation of connectivity when collision avoidance is included. Developed a new algorithm that uses target clustering to provide guarantees even when obstacles are included. Created tests to illustrate the performance of the new algorithm relative to those in literature, then wrote the code for these tests. Implemented the algorithms on real robots. Parsed and analysed results. Prepared the manuscript and generated all figures.
Contribution Percentage (%):	80
Signature:	Date: 17 Mar, 2019

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

1. the candidate's stated contribution to the publication is accurate (as detailed above);
2. permission is granted for the candidate to include the publication in the thesis; and
3. the sum of all co-author contributions is equal to 100% less the candidates stated contribution.

Name:	Steven Grainger
Contribution Details:	Guided research direction. Supervised work development. Helped generate ideas for tests and edited manuscript.
Signature:	Date: 15 Mar, 2019

Name:	Ben Cazzolato
Contribution Details:	Guided research direction. Supervised work development. Helped generate ideas for tests and edited manuscript.
Signature:	Date: 13 Mar, 2019

Formation-Based Multi-Robot Routing with Inter-Robot Distance Constraints

Nick Sullivan, *Member, IEEE,* Steven Grainger, *Member, IEEE,* and Ben Cazzolato

***Abstract*—We provide a formation-based algorithm to solve the problem where robots are required to visit multiple targets, with the added constraint that they must remain within a certain distance of one another. We show that this algorithm is more scalable than existing algorithms, and is able to guarantee distance constraints even in the presence of static obstacles. We empirically compare its performance to other algorithms, altering the number of robots, targets, and distance constraint. We then compare the results to the implementation on real vehicles, illustrating near identical performance. Finally, we illustrate how our algorithm can be applied to an example system where multiple vehicles visually track one another to determine their locations.**

***Index Terms*—Multi-robot routing, multiple traveling salesman problem, task allocation**

I. Note to Practitioners

The motivation behind this work is problems with multiple robots that must move to a number of locations to perform certain tasks, e.g. manufacture components, transport people, monitor the environment, or rearrange goods. The innovation is in the added requirement that they stay near one another, e.g. to observe one another in order to detect failure; to be prepared for new tasks that require multiple robots to work together; to maintain network connectivity over large distances; or to fuse sensory information from multiple robots detecting the same area. This paper provides and analyses a planning algorithm, generating specific paths to be followed by each robot. The primary objective is to complete all tasks as quickly as possible, while ensuring robots do not go beyond the distance limit, and guarantees no robot to robot collisions, and no robot to static-obstacle collisions. It does not consider dynamic obstacles, assumes robots are able to stop (i.e., not suitable for gliders), and assumes all robots have a relatively small turning circle.

II. Introduction

Multi-robot routing is the act of planning how robots should move between target locations. In this paper, we consider targets that can be visited in any order by any robot. Problems of this type occur in industries such as transportation [1], environmental monitoring [2], container loading and unloading [3], and manufacturing [4]. Inefficient routing can cause delays and excess energy usage, reducing the amount of useful work robots can perform. However, this problem is not trivial to solve. The number of possible routes scales factorially with the number of targets, requiring enormous amounts of computing power to consider every route, even for small problems.

School of Mechanical Engineering, The University of Adelaide, Australia, 5005 e-mail: nicholas.sullivan@adelaide.edu.au

The algorithms that have been developed for this problem can be broadly categorised by solution quality versus processing time. The fastest algorithms to compute are known as heuristics, and produce routes iteratively. The best performing heuristics for multi-robot routing are sequential-single item auctions, where robots bid on targets based on their ability to complete them. Each round, a target is sold to a robot, which then updates its bids accordingly. This is repeated until all targets have been allocated to robots. Different objectives can be met using different bidding rules [5]. Heuristics are often used to handle dynamic scenarios as they can be recalculated quickly [6], [7].

A category of algorithms known as metaheuristics can produce better solutions than heuristics, but require additional processing time. Metaheuristics are problem-independent search algorithms that are tuned to solve particular problems. In this problem, metaheuristics search for good routes amongst many bad routes. They operate through *exploration*, finding new solutions (valid routes) that are not similar to those found before, and *exploitation*, improving upon discovered solutions. Many metaheuristics exist, and it is not yet clear which (if any) is most suitable for the multi-robot routing problem. Examples of metaheuristics used to solve the multi-robot routing problem include genetic algorithms (GAs) [8], simulated annealing [9], ant colony optimisation [10], and tabu search [11].

In some cases, we can solve the multi-robot routing problem optimally. Solvers for Integer Linear Programs (ILPs) are able to ignore a large portion of sub-optimal routes using techniques such as branch and cut [12]. This requires that the problem can be represented as an ILP, which limits the types of objectives and constraints that can be solved. Notably, the objective to visit all targets in the fastest time is non-linear, and cannot be represented as an ILP. For applicable objectives and constraints, ILPs can solve problems of up to 100 targets in the order of seconds, but scales poorly with more targets [13].

Literature has previously addressed the additional constraints that robots must cooperate for certain tasks to be completed. Transportation vehicles may need to meet at the same location to transfer goods [14], re-fuelling stations may only be able to service one robot at a time [15], and drones may be used for short-range delivery while returning to a truck for long-range movement [16]. These are applications considered as a Vehicle Routing Problem (VRP), using constraints such as carrying capacity, time windows, and driver hours [17].

However, these constraints only need to consider the time of

arrival at each target, and do not need to consider the physical movement between targets. As such, algorithms designed for these constraints cannot be applied to cases where constraints are applicable for the entirety of a robots' journey, not just their time of arrival at specified targets. One such constraint is where robots remain near each other during operation in order to maintain communication capabilities, or to visually observe one another.

If robots can maintain communication, they can dynamically re-plan when faced with new targets, obstacles, or hardware failures. Maintaining distance to other robots also enables the use of *cooperative localisation*, where robots work together to estimate their location and orientation. Cooperative localisation provides greater performance over singular localisation [18], and enables the use of robots with poor sensing by leveraging robots with greater sensing [19].

Recent research on inter-robot distance constraints has addressed the optimal placement of robots to provide the best possible network connectivity [20], as well as searching strategies while maintaining sufficient connectivity to relay video [21]. Tuning rules can be used to adjust how much information can be collected before reporting it back to a base station [22]. A comparison of four communication-based exploration techniques is available, along with a short taxonomy of communication-constrained exploration types [23]. When applied to real systems, it is common for an outer control layer to make a plan and for an inner control layer to measure and ensure connectivity remains strong [24], [25]. However, these techniques consider one target at a time, and do not consider multiple targets that can be completed in any order.

A bounded solution has been found for a two-robot problem where robots alternate movement on a small grid [26]. All targets can be visited using 9/2 of optimal energy usage while maintaining distance, but under the limiting conditions of a small grid with alternating movements. A sequential-auction based heuristic has been created to constrain distance between robots, known as Connected Nearest Neighbour [27], which alternates between target allocation and network maintenance phases. However, it assumes that robots move in straight lines between targets, which would potentially violate the distance constraints if the vehicles use obstacle avoidance to move around obstacles.

Genetic algorithms have been used to search for solutions that constrain inter-robot distances. One uses a customised improvement mechanism to turn solutions that break this constraint into ones that do not [28]. This approach measures the distance between robots whenever one reaches a target, which does not necessarily guarantee robots remain within range at all times. For example, two robots can move different ways around an obstacle. Another algorithm, known as Dual-GA, periodically introduces waypoints, and discretises paths to measure distance for the entire journey [29]. Both these approaches use randomness to search for valid solutions, and therefore do not guarantee that a valid result will be found.

We introduce a new algorithm for routing multiple robots to multiple target locations. The objective is to visit all targets and return to base within the fastest time. We use a clustering and formation based approach to ensure robots remain within a required distance at all times. It is guaranteed to avoid robot-robot collisions, as well as collisions with static obstacles. We compare the performance of our algorithm to a sequential-auction based algorithm (Connected Nearest Neighbour [27]), a GA based algorithm (Dual-GA [29]), and an integer linear program for a single robot (MATLAB TSP solver [30]). The performance is compared using MATLAB, and our algorithm is verified on a real system.

In this paper, we formalise the problem in Section III. We introduce our new algorithm in Section IV. Results from MATLAB simulations are presented in Section V. Computation time of the algorithm is analysed in Section VI. Verification of these results were performed on real robots, discussed in Section VII. We conclude the paper in Section VIII.

III. PROBLEM DEFINITION

Consider a set of robots R that start at positions S. These robots must visit a set of targets T then return to their start positions. They do so by moving in a path p_r, which is the robot position as a function of time. They are not allowed to occupy the same position at the same time, nor may they occupy the same position as a set of obstacles O. The robots must also stay within range c of one another, which is assumed to be given as part of the problem. To define the problem, we first define some functions. The first function, $\delta_1()$, calculates the time a path will take to traverse. $\delta_2()$ returns the distance between two paths as a function of time. $\alpha_1()$ takes two paths, and returns how many times the paths collide (same point at the same time, within a specified tolerance). $\alpha_2()$ takes a path and a point, and returns the number of times the path moves through that point (with a specified tolerance).

The non-linear program formulation for the mTSP is as follows:

$$\text{minimise: } \max_{r \in R} \delta_1(p_r) \tag{1}$$

subject to:

$$\sum_{r \in R} (\alpha_2(t, p_r)) \geq 1 \qquad \forall t \in T \tag{2}$$

$$\sum_{r \in R} (\alpha_2(o, p_r)) = 0 \qquad \forall o \in O \tag{3}$$

$$\alpha_1(p_{r_1}, p_{r_2}) = 0 \qquad \forall r_1, r_2 \in R, r_1 \neq r_2 \tag{4}$$

$$\max(\delta_2(p_{r_1}, p_{r_2})) \leq c \qquad \forall r_1, r_2 \in R, r_1 \neq r_2 \tag{5}$$

$$p_{r,start} = s_r \qquad \forall r \in R \tag{6}$$

$$p_{r,end} \in S \qquad \forall r \in R \tag{7}$$

Objective (1) is to minimise the maximum path length, i.e., visit all targets in the fastest amount of time. Constraint (2) specifies that all targets must be included in at least one robot's path. Robots may move through the target more than once if desired. Constraint (3) specifies that no robot may move through an obstacle. Constraint (4) specifies that no two robots may be at the same point at the same time. Constraint (5) specifies that all robots must be within a certain distance with one another. If robots can act as relays then this constraint could be relaxed to one that specifies all robots belong in the same network. Constraint (6) specifies that each robot's path

TABLE I: Average performance improvement using different robot zone generation algorithms across maximum allowed distances between them (5 to 50 m) and number of targets (10 to 100). Measured relative to single-robot solutions (%).

Num Robots	2	3	4	5	6
K-means	6.9	**15.0**	**17.0**	**17.8**	**18.2**
Grid-based	6.4	12.4	14.4	15.4	15.9
Hierarchical	**8.9**	12.7	13.5	13.8	13.8

must begin at the robot's start position. Constraint (7) specifies that each robot's path must end at a robot's start position, not necessarily the same one that they started at.

We consider a robot that moves at a constant speed, is able to stop, and can turn on the spot. This representation is also valid if the distance between targets is large relative to the robots' turning circles.

IV. ALGORITHM

Firstly, we define a circle called the *robot zone*. The robot zone has diameter equal to the maximum allowed distance between the robots. If all robots are within the robot zone, they are guaranteed to be within range of one another.

Our formation-routing algorithm can be briefly summarised as a series of steps:

- Create robot zones that encapsulate all robot and target locations
- Create a single path through all robot zones
- For each robot zone:
  - Move robots through all targets within the zone (intra-zone)
  - Move robots in formation to the next zone (inter-zone)

Examples of robot zones can be seen in Figures 1, 2, 3 and 4. A number of methods can be used for generating robot zones, including K-means clustering [31], agglomerative hierarchical clustering [32], and grid-based set cover [33]. The algorithms are listed in Algorithms 1, 2, and 3, respectively. An average comparison of multi-robot performance is shown in Table I. We make use of K-means clustering, as it produced better results.

Once robot zones have been generated, we plan a path through the centre of each zone, referred to as the *robot zone path*. This is a single path, so it is an example of the Travelling Salesman Problem (TSP). The TSP has been researched for many years [34], so we do not repeat the details here. We represent the TSP as a integer linear program and solve it in MATLAB to generate optimal robot zone paths [30], examples of which are shown in Figures 1, 3, and 4.

At first, all robots are located within the starting robot zone. They must move through all target locations within the zone and end in a specified formation, ready for inter-zone movement. The robots must remain within the robot zone so that their distance constraint is guaranteed. If the robots move in a straight line between targets, they are guaranteed to remain within the robot zone. The intra-zones allocation problem is therefore a problem formulation identical to Equations (1)

Algorithm 1 K-means Clustering

// Clusters move to encapsulate as many targets as
// possible.
procedure
  $n \leftarrow size(T)$ ▷ num targets
  $m \leftarrow 0$ ▷ num clusters
  $a \leftarrow 0$ ▷ num targets in clusters
  while $a < n$ **do**
    $m \leftarrow m + 1$
    for attempt $\in \{1, \ldots, 100\}$ **do**
      $KL_i =$ (rand,rand) $\quad \forall i = \{1, \ldots, m\}$
      for iteration $\in \{1, \ldots, 10\}$ **do**
        $a \leftarrow 0$
        $K_i \leftarrow \{\}$ $\quad \forall i = \{1, \ldots, m\}$
        for all $t \in T$ **do**
          $k = \underset{\forall i \in \{1,\ldots,m\}}{\text{argmin}} \ (\text{dist}(KL_i, TL_t))$
          $K_k \leftarrow K_k + \{t\}$
          if $\text{dist}(KL_k, TL_t) < d_{max}$ **then**
            $a \leftarrow a + 1$
        if $a = n$ **then**
          return
        $L_i = \text{mean}(K_i)$ $\quad \forall i = \{1, \ldots, m\}$

Algorithm 2 Agglomerative Hierarchical Clustering

// Clusters are placed on targets, and redundant
// clusters are removed.
procedure
  $n \leftarrow size(T)$ ▷ num targets
  $m \leftarrow size(T)$ ▷ num clusters
    ▷ Cluster locations are on target locations
  $K_i \leftarrow T_i$ $\quad \forall i = \{1, \ldots, n\}$
  $KL_i \leftarrow TL_i$ $\quad \forall i = \{1, \ldots, n\}$
  $merged \leftarrow true$
  while $merged$ **do**
    $merged \leftarrow false$
    $K \leftarrow sort(K, size())$ ▷ descending
    $c \leftarrow 0$
    while !$merged$ **do**
      $k_1 \leftarrow K_c$
      for all $k_2 \in K, k_2 \neq k_1$ **do**
        $K_{merge} \leftarrow K_{k1} \cup K_{k2}$
        if $\underset{\forall t \in K_{merge}}{\max} \ (dist(KL_{k1}, TL_t)) <= d_{max}$ **then**
          $K_{k1} \leftarrow K_{merge}$
          $K_{k2} \leftarrow \{\}$
          $merged \leftarrow true$
      $c \leftarrow c + 1$

Algorithm 3 Grid-based Set Cover Clustering

// A grid of clusters is created, those that encapsulate
// the most targets are picked.
procedure
    $n \leftarrow size(T)$ ▷ num targets
    $m \leftarrow size(T)$ ▷ num clusters
    $a \leftarrow 0$ ▷ num targets in clusters
    $K_i \leftarrow \{\}$ $\forall i \in \{1, \ldots, m\}$
    $P_i \leftarrow \{\}$ $\forall i \in \{1, \ldots, n\}$
    $k \leftarrow 0$
    for all $x \in xrange$ **do** ▷ set cluster locations
        for all $y \in yrange$ **do**
            $KL_k \leftarrow (x, y)$
            $k \leftarrow k + 1$
    for all $i \in \{1, \ldots, m\}$ **do** ▷ link clusters to targets
        for all $j \in \{1, \ldots, n\}$ **do**
            if $dist(KL_i, TL_j) <= d_{max}$ **then**
                $K_i \leftarrow K_i + \{j\}$
                $P_j \leftarrow P_j + \{i\}$
    $K_{final} \leftarrow \{\}$
    while $a < n$ **do** ▷ pick cluster with most targets
        $k \leftarrow \underset{\forall i \in \{1,\ldots,m\}}{\text{argmax}} \ (size(K_i))$
        $K_{final} \leftarrow K_{final} + \{k\}$ ▷ lock-in selection
        for all $j \in K_k$ **do** ▷ targets in this cluster
            for all $i \in P_j$ **do** ▷ clusters in range
                $K_i \leftarrow K_i - \{j\}$ ▷ remove this target
            $a \leftarrow a + size(P_j)$
            $P_j \leftarrow \{\}$

through (7), except without the inter-robot distance constraint (5), and with different end positions.

To solve this problem, we use an iterative process known as a sequential single-item auction. Targets are sold to the robots using an auction process. Robots submit bids equal to the total path distance if they were to include a given target in their path [5]. An auctioneer then calculates the best bid, and allocates the target to the winner. Traditionally, the best bid is the lowest one, but our previous work has shown that it is better to use the least contested bid for the fastest completion objective [35]. Robots are not allowed to bid on targets that result in paths intersecting. If paths initially intersect, their tails are swapped. This auction process is given in Algorithm 4. Any other task allocation process could also be used, as this does not need to meet distance constraints.

Once all targets within a robot zone are complete, we move the robots between robot zones while maintaining the distance between them. We do this by locking robots in formation. A robot will wait in its formation location until all robots have assembled. They will then move together to the next robot zone, as can be seen in Figure 2. We assume they are able to maintain formation using a basic formation control algorithm [36].

There are a couple of points of note here. Firstly, robots do not rotate their formation once it has been fixed (otherwise, robots will be required to move at different speeds). They

Algorithm 4 Completing targets within a robot zone

procedure SEQUENTIAL SINGLE-ITEM AUCTION
    $n \leftarrow size(T)$ ▷ num targets
    $A \leftarrow T$ ▷ to be auctioned
    $p_r \leftarrow (s_r, e_r)$ $\forall r \in R$ ▷ path start and end
    $p \leftarrow$ checkForCollisionsAndSwap(p)
    for $\{1, \ldots, n\}$ **do**
        for all $t \in A$ **do**
            // Calculate new paths if target is sold
            for all $r \in R$ **do**
                $pn_r \leftarrow$ addTargetToPath(p_r, t)
                $bid_r \leftarrow$ calculateCost(pn_r)
                if checkForCollisions(pn) **then**
                $bid_r \leftarrow \infty$
            // Calculate the least contested bid
            $rn_1 \leftarrow \underset{\forall r \in R}{\text{argmin}}(bid_r)$
            $rn_2 \leftarrow \underset{\forall r \in R, r \neq r_1}{\text{argmin}} \ (bid_r)$
            $lcb_t \leftarrow bid_{r_2} - bid_{r_1}$
            // Store the best robot and path
            $rn_t \leftarrow r_1$
            $pn_t \leftarrow pn_{r_1}$
        // Choose the winner, update the path
        $t_{win} \leftarrow \underset{\forall t \in T}{\text{argmax}}(lcb_t)$
        $r_{win} \leftarrow rn_{t_{win}}$
        $p_{r_{win}} \leftarrow pn_{t_{win}}$
        $A \leftarrow \{A\} - \{t\}$
// We insert the target into the path
procedure ADDTARGETTOPATH(p,t)
    $m \leftarrow size(p)$
    $c_{min} \leftarrow inf$
    $pn_{min} \leftarrow \{\}$
    for $i = \{1, \ldots, m\}$ **do**
        // Add target t to path p at index i
        $pn \leftarrow$ insertIntoPath(p,t,i)
        $c \leftarrow$ calculateCost(pn)
        if $c < c_{min}$ **then**
            $c_{min} \leftarrow c$
            $pn_{min} \leftarrow pn$
    return pn_{min}
// We swap path tails if they collide
procedure CHECKFORCOLLISIONS(p)
    for $r_1 \in R$ **do**
        $p_{1_{len}} \leftarrow size(p_{r_1})$
        for $i = \{1, \ldots, p_{1_{len}}\}$ **do**
            $seg_1 \leftarrow p_{r_1, i \text{ to } i+1}$
            for $r_2 \in R$ **do**
                $p_{2_{len}} \leftarrow size(p_{r_2})$
                for $j = \{1, \ldots, p_{2_{len}}\}$ **do**
                $seg_2 \leftarrow p_{r_2, j \text{ to } j+1}$
                if checkCollision(seg_1, seg_2) **then**
                $p \leftarrow swapTails(pr_1, pr_2, i+1, j+1)$

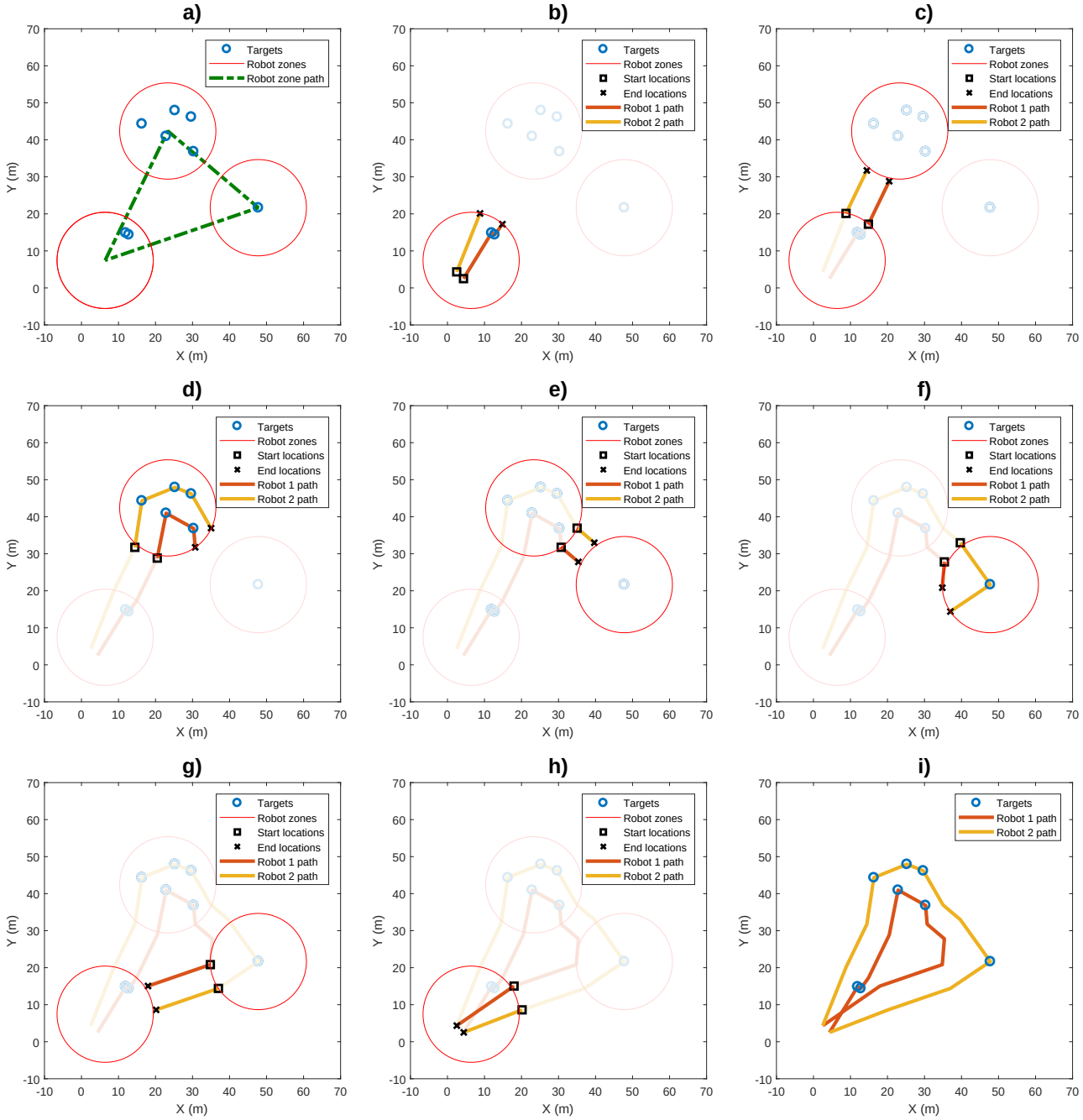


Fig. 1: An example problem being solved. Two robots begin in formation near (0,0), and must visit all targets while staying within required range of one another. a) Targets are clustered into robot zones with diameters equal to the maximum allowed distance between robots. A path through these zones is generated. b-h) Robots complete targets within zones, then wait at locations marked by x's until formation is restored. Robots then move between robot zones in formation. i) The final path plan for the two robots.

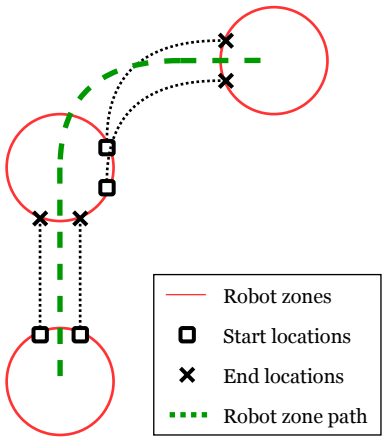


Fig. 2: The inter-robot zone process. When robots have completed all targets within a robot zone, they establish and hold a formation from one robot zone to another. The path they follow matches the robot zone path.

therefore produce the formation required for the *next* robot zone. An example of this is shown in Figure 2, where the robots pre-rotate the formation to accommodate the rotation of the robot zone path. Secondly, the robots follow an identical path as the robot zone path, ending once all robots have entered the next robot zone.

A step-by-step example of the routes can be seen in Figure 1. Note that some solutions have paths that cross over one another. Collisions will not occur, because the crossing paths happen at different time segments. If we consider the example shown in Figure 1, a series of paths that do not cross may look like they cross if we do not consider time. Even if a robot were to drive faster or slower than planned, it will be forced to wait until formation is restored before moving onwards.

A. Static Obstacles

In addition to planning paths that prevent robots colliding with one another, our method supports avoidance of static obstacles. In an obstacle-free environment such as shown in Figure 3, movement between vertices is done with a straight line. In environments with obstacles, such as in Figure 4, straight lines could lead to robots crashing into the obstacles. Instead, we perform the following alterations to allow for obstacle avoidance.

We separate obstacles into two categories: those that are small enough for robots to stay in range while on opposite sides (small), and those that are too large for this (large). For the large obstacles, we ensure robots stay within range by forcing all robots to go around them the same way. This is done by altering the path between robot zones. We inflate the large obstacles by the required inter-robot distance, and plan a robot zone path around these inflated obstacles using MATLAB's probabilistic roadmap (although any path planner could be used, such as A* [37]).

For the small obstacles, robot zones are placed so that small obstacles are located entirely within them (i.e. not touching the edge of a robot zone). We then plan paths around the obstacles during the intra-zone phase.

The updated algorithm is as follows:

- Separate obstacles into small and large obstacles
- Inflate the large obstacles by the required inter-robot distance
- Create robot zones that encapsulate all robot and target locations, whose locations must not coincide with an inflated large obstacle. The edges of a robot zone must not overlap with a small obstacle.
- Create a single path (robot zone path) through all robot zones, the path must not go through an inflated large obstacle
- If there are any small obstacles within the required inter-robot distance of the robot zone path, and if they are not already located within a robot zone, add a new robot zone to include it and create a new robot zone path
- For each robot zone:
  - Move robots through all targets within the zone, without intersecting with small obstacles (intra-zone)
  - Move robots in formation to the next zone (inter-zone)

This process can be seen in Figure 4. Firstly, robot zones are selected to contain all targets and small obstacles. A robot zone path is generated, giving sufficient space between the path and the large obstacles. The final solution for one and two robots are given.

In short, robots avoid large obstacles because they follow the robot zone path between zones. Robots avoid small obstacles using path planning within a zone.

For robot zone clustering with obstacles, we make use of grid-based set cover clustering (Algorithm 3) for robot zone selection (instead of K-means used earlier). It can quickly find a valid set of robot zones, or return that none could be found that meet all the criteria, whereas the K-means clustering algorithm will loop infinitely if it is unable to find a valid set.

V. RESULTS

In our analysis, we place the robots in formation around (0,0) and randomly generate a number of targets within a 100 m x 100 m area. An Integer Linear Program (ILP) [30] is used to solve for the single-robot solution (a standard TSP), which acts as a solution benchmark. We then apply our algorithm for comparison. We alter the maximum allowable distance between robots (5 to 50 m), the number of targets (10 to 100), and the number of robots (2 to 10). Each configuration is repeated 100 times to generate an ensemble average. The computation is performed using the University of Adelaide's high-performance computing cluster, Phoenix. The robot formation was in an arc, evenly spaced between $-45°$ and $+45°$.

We can see the performance of the formation routing algorithm in Figure 5. The first thing to notice is that it does not always perform better than the single-robot solution. A green line indicates this crossover, below which the single-robot solution is better. In particular, it performs poorly when

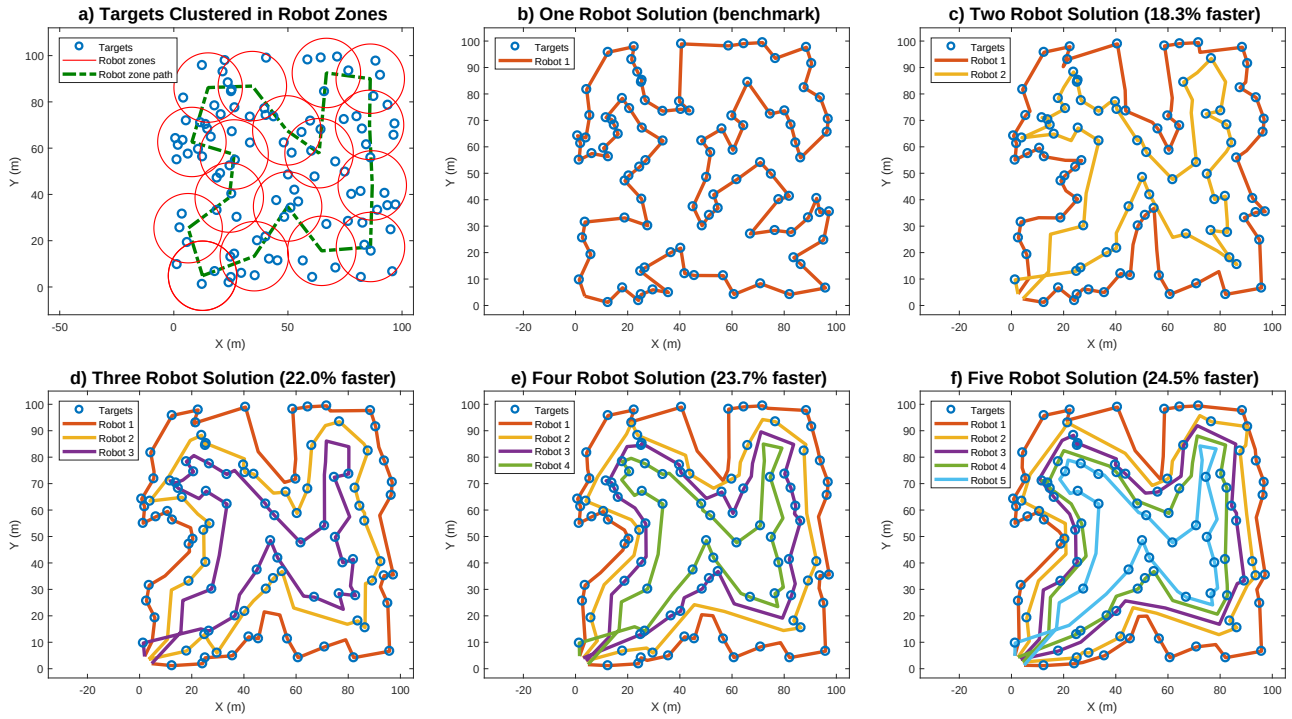


Fig. 3: Optimisation without obstacles. Robots begin in formation near (0,0), and must visit all targets while staying within required range of one another. a) Targets are clustered into robot zones with diameters equal to the maximum allowed distance between robots. b) An optimal solution for a single robot is found using an Integer Linear Program (ILP). This is used as a benchmark. c-f) Multi-robot solutions are generated through our formation-routing algorithm that guarantee no robot collisions and all robots stay within a required range of one another.

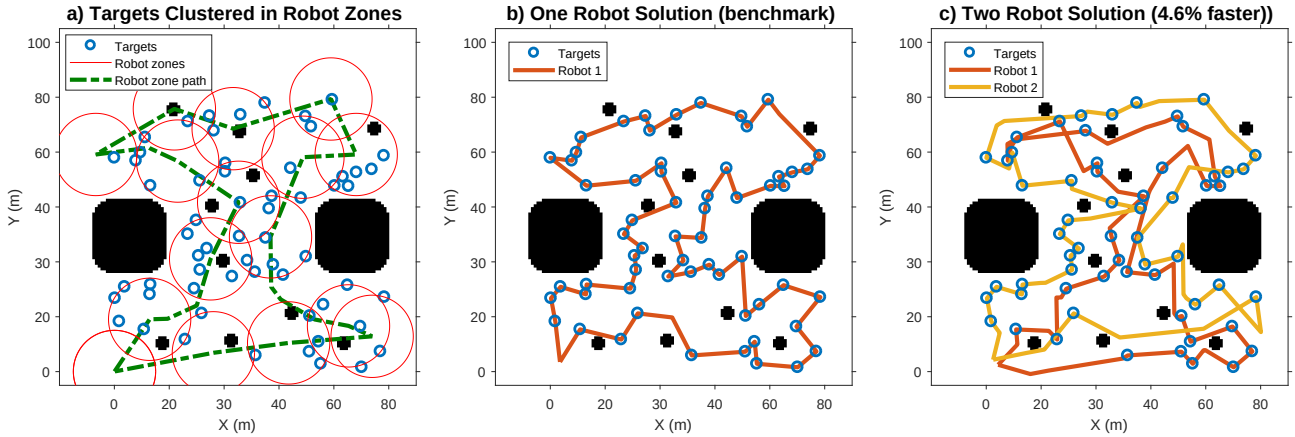


Fig. 4: Optimisation with obstacles. Robots begin in formation near (0,0), and must visit all targets while staying within a specified distance of one another. They cannot move through obstacles (black regions). a) Targets are clustered into robot zones with diameters equal to maximum allowed distance. The path between robot zones avoids larger obstacles, and encapsulate smaller ones. b) An optimal solution for a single robot is found using an Integer Linear Program (ILP). This is used as a benchmark. c) A two-robot solution is generated through our formation-routing algorithm that avoids larger obstacles by following the robot zone path, and avoids smaller obstacles using a path finding algorithm such as MATLAB's probabilistic roadmap. It guarantees no robot collisions with obstacles or each other and ensures robots stay within required distance of one another.

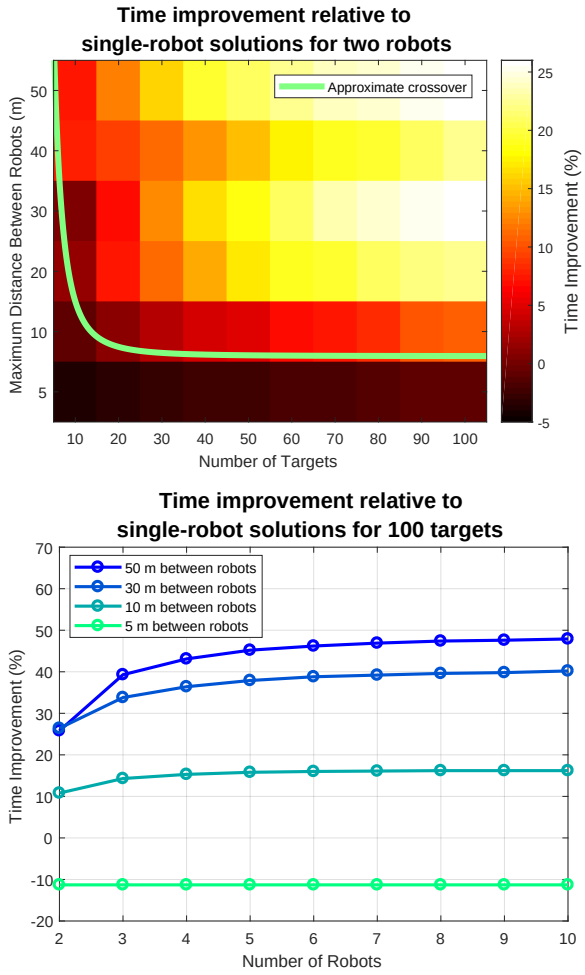


Fig. 5: The time required to reach all targets and return home while staying within required distance using our algorithm. Performance is measured relative to optimal solutions for a single robot.

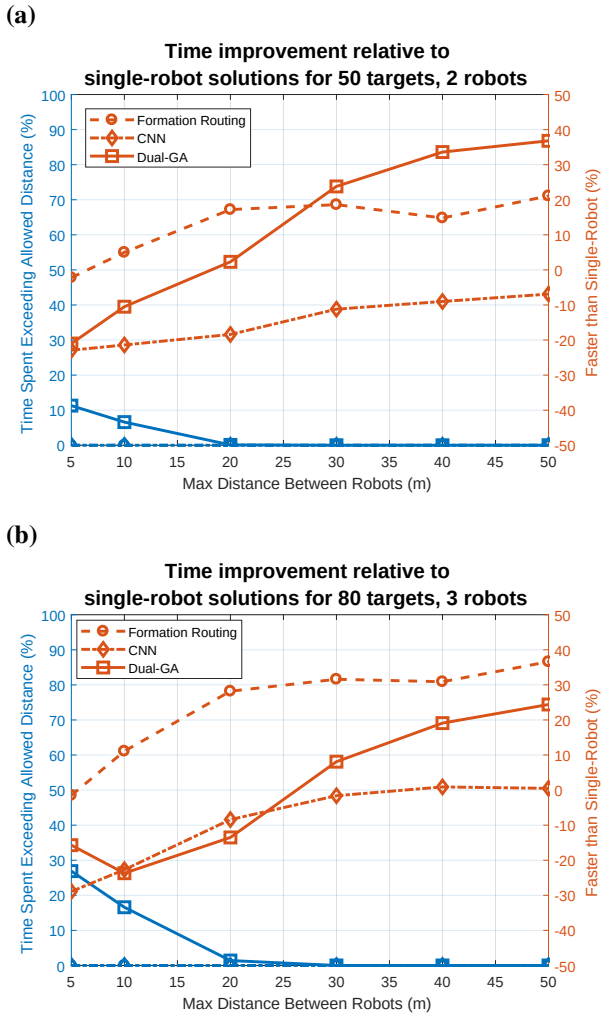


Fig. 6: The performance of formation routing algorithm, compared to Connected Nearest Neighbour (CNN) [27], Dual-GA [29], and a single-robot solution equivalent to the Travelling Salesman Problem.

the required distance between robots is very low (less than 5 m in a 100 m x 100 m area). A small inter-robot distance results in very few targets per robot zone, which means only one robot performs useful work. The other robots move solely to stay in range. When this is the case, the single-robot solution produces a more efficient path, resulting in faster completion.

Beyond this edge case, our algorithm performs significantly better than a single robot. As we increase the number of robots, the targets are completed much faster (in our tests, up to 26% for two robots, 39% for three robots, 45% for five robots). These improvements are biggest when the allowable inter-robot distance is large, as there is more room within robot zones for robots to reach targets in parallel. There are diminishing returns as we saturate the system with more robots, where the time gained from reaching targets in parallel is close to the time spent keeping them within range of one another.

We compare our formation routing algorithm to two other algorithms in literature in Figure 6. We can see that formation routing performs best for the more complex scenarios (80 targets, 3 robots), but is outperformed by Dual-GA for the smaller problem (50 targets, 2 robots) with large inter-robot distances ($>$ 30 m in a 100 m x 100 m area).

To summarise the performance of the compared algorithms:

Single-Robot Solution: This is a standard Travelling Salesman Problem (TSP). This method produced the fastest paths when the inter-robot distance is very low (5 m in 100 m x 100 m area), because the distance is not large enough to allow targets to be visited in parallel.

Formation Routing: This method (introduced in this paper), clusters targets and uses a combination of formation and sequential auction to create paths. It produces the fastest paths for small to medium inter-robot distances (10 m to 30 m). At larger communication distances, it is easier for other techniques to find paths that do not spend time maintaining formation. This method scales well for number of targets and robots.

Genetic Algorithm (Dual-GA [29]): This method searches

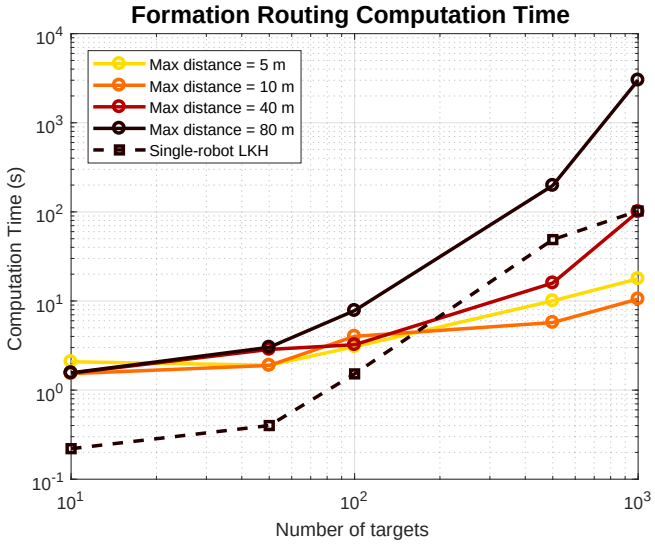


Fig. 7: The computation time for the formation routing algorithm. Targets were randomly placed in a 100 m x 100 m area, then solved by the formation routing algorithm. The Lin-Kernhigan-Helsgaun algorithm [38] was used to generate single-robot solutions as a benchmark.

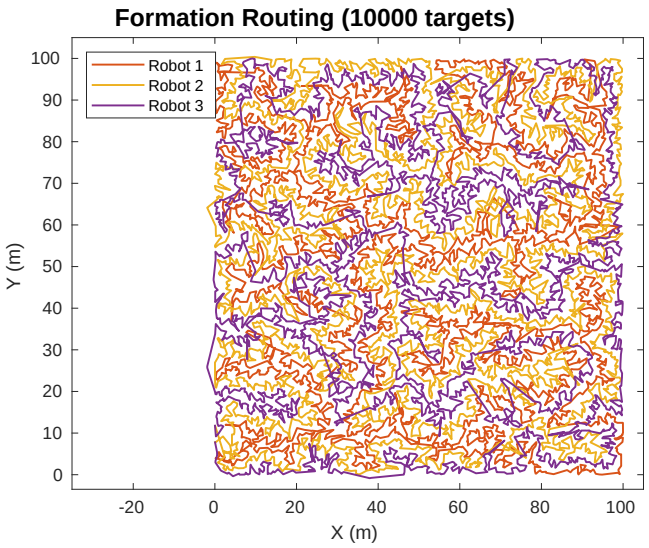


Fig. 8: An example solution to a problem with ten thousand targets. Maximum allowed distance between robots is 10 m. It took 13.5 seconds to cluster into 217 robot zones, 5.9 seconds to plan a path through the robot zones, then 278.4 seconds to solve sequentially. If performed with full parallelism (217 parallel computers), this could be completed in 20.7 seconds. In comparison, the Lin-Kernhigan-Helsgaun algorithm took 3 hours and 44 minutes.

for paths that meet the constraints. It produces the fastest paths for large inter-robot distances (> 40 m) in small problems (< 50 targets, < 4 robots). However, it may not find solutions that meet the inter-robot distance constraint when distances are small (< 30 m). When problems are large (> 50 targets, > 4 robots), the enormous number of possible solutions makes it difficult to find fast paths.

Sequential Auction (CNN [27]): This method incrementally builds paths. While this never produced the fastest paths, it is able to be calculated more quickly than any other tested algorithm.

VI. Computation Time

The formation-routing algorithm was tested with thousands of targets to determine its scalability. To accomodate this, we did not use an exact TSP solver to form a single path through robot zones, as it can take hours to solve beyond 100 tasks [13]. We instead use LKH [38], an efficient implementation of the Lin-Kernighan heuristic, with added improvements by Helsgaun. It produces near-optimal results for TSP problems while finding solutions orders of magnitude faster than exact algorithms.

The computation time for a single-core single-node computer in The University of Adelaide's High Performance Computing Cluster (Intel Xeon Gold 6148, 2.40 GHz, 4 GB RAM) to perform the formation routing algorithm is shown in Figure 7. The computation time is largely dependent on the maximum allowed distance between robots. If the distance is small, the majority of time is spent on finding a path between all robot zones. If the distance is large, the majority of time is spent performing auction allocation. The latter is slower, as the collision check can take a long time when robot paths are long. The formation routing algorithm is sometimes faster than the LKH algorithm, which can allow large problems to be solved efficiently, such as in Figure 8. A ten-thousand target problem is solved within 5 minutes, with the possibility of reducing this time to 21 seconds if parallelism is used. Once a path between robot zones is formed, all movement within and between robot zones can be performed in parallel. The smaller the maximum distance is, the more zones will be used, which increases the level of parallelism possible. It should also be noted that the computation time of the formation routing algorithm is largely independent of the number of robots.

VII. Physical Implementation

Our algorithm was implemented on Clearpath Jackals, shown in Figure 9. These robots use the Robot Operating System (ROS), and are equipped with an accelerometer, gyroscope, magnetometer, wheel encoders, and GPS. They use these sensors to localise themselves using an Extended Kalman Filter (EKF), implemented in a popular ROS package *robot_localization*. The robots communicate using long-range Ubuiquity Bullet radios, with a base station that can operate as a wireless access point. The robots operate using a multi-master extension called *multimaster_fkie*, so they can operate independently when apart.

We consider a scenario where one robot has suffered hardware failure of their GPS. The environment is a large open field, making other methods of localisation difficult (no landmarks are available for SLAM loop-closure [39]), i.e. this robots position error will increase over time. The goal is to successfully visit all targets in the fastest time, even with limited localisation capabilities.

Each Jackal is also equipped with four Pointgrey Grasshopper3 cameras mounted on each side, with a resolution of 2048x2048 pixels. The images of these cameras are then processed to detect AprilTag fiducial markers [40], which provide an estimate of the relative location of other robots. We ran the cameras at 1 Hz so that the robots were not overloaded with visual processing. With this relative location information, we can improve position and orientation estimates using Collaborative Localisation (CL) [41]. The robot with GPS will watch and guide the robot without GPS.

We randomly placed 30 targets in a 50 m x 50 m area on a grass field. Our objective is to visit all targets in the fastest time, while ensuring the robots stay within 20 m of one another.

Using our formation-based algorithm, we generated a series of waypoints for the robots to drive to. A simple waypoint following script was written for the robots, where robots drive forward at constant speed (0.5 m/s), and rotate until they are facing the next waypoint (up to 1 rad/s). When they need to establish formation, they will wait for all robots to arrive at the required location and report that they are in formation. The robots send periodic information to one another at 1 Hz.

Firstly, we give robots access to GPS to compare real results with our calculations. This can be seen in Figure 9. In part (b), a single robot visits all targets in 487 s, where our calculations predicted 489 s. In part (c), two robots visited all targets in 441 s, resulting in an improvement of 9.6%. Our calculations were expecting a time of 426 s, and an improvement of 12.8%. The discrepancy can be due to the extra time the robots spent waiting for formation to be established, where a 2 s delay was included to ensure both robots were in agreement that a formation had been formed. Excluding this, the robots would have visited all targets in 423 s, resulting in an improvement of 13.1%.

These experimental results closely match our calculations, with a difference of 0.6%. The real robots completed all tasks slightly faster than what our calculations predicted, even without driving in perfectly straight lines (due to GPS noise). This is because the robots consider a target as visited when they are within 1 m of it, so their paths are slightly shorter.

For the experiment shown in Figure 10, we give only one robot access to GPS, so CL is required to accurately localise both robots. Visual detections occur best when robots are within 20 m of one another, so the robots must remain within this distance during operation.

In part (a), the robot without GPS operates purely using dead-reckoning (integrating velocity), and eventually loses its bearing due to wheel slippage on the grass, resulting in many of the targets being missed. In part (b), inter-robot detections are used to vastly improve the localisation of the robot without GPS. We found that inter-robot detections were particularly effective when the observing robot takes the shorter path, as camera detections were most accurate when the observing robot was stopped and waiting for the other robot to establish formation. Part (c) shows that the robots always stay within their 20 m distance limit.

This scenario is an example of where inter-robot distance constraints are fundamental for successful task completion,

(a) Robots

(b) One Robot Solution (benchmark)

(c) Two Robot Solution (9.6% faster)

Fig. 9: The objective is for robots to visit all target locations and return to the start in the fastest time. a) The robots used in the experiment. b) A single robot visited all targets in 487 s. c) Two robots visited all targets in 441 s using our formation-routing algorithm.

(a) Singular Localisation

(b) Collaborative Localisation

(c) Distance between robots. Must be < 20 m.

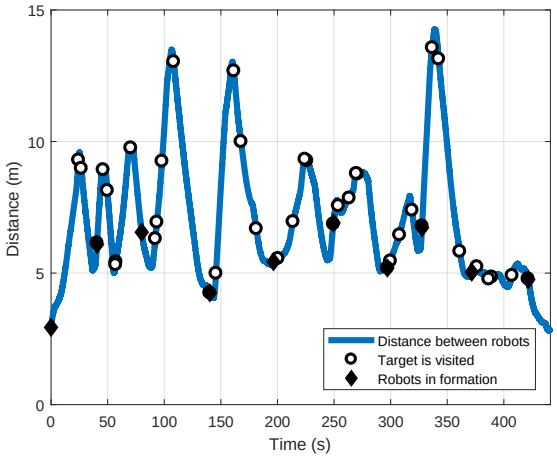


Fig. 10: The objective is for robots to visit all target locations and return to the start in the fastest time. One robot has GPS, while the other does not. Our formation-based routing algorithm was used to generate paths that stay within 20 m of one another, so that collaborative localisation can be used. a) Without GPS, one of the robots drifts. b) Inter-robot detections allow both robots to visit their targets. c) The distance between the robots is within 20 m at all times.

and illustrates how the formation-based routing algorithm can be used to generate robot routes that meet the distance constraints.

VIII. Conclusion

We have developed a formation-based routing algorithm that ensures a number of targets are visited by robots, while constraining that the robots do not exceed a given distance from one another. This algorithm is more scalable than existing methods, while offering stronger distance guarantees in the presence of static obstacles. Calculations closely match the results from a real robot trial, and illustrate how this algorithm can be applied to systems where multiple vehicles share sensory information. It could equally be applied to systems where vehicles must maintain communication with one another, allowing them to immediately respond to new information and hardware failure.

Acknowledgment

This research was supported by the Phoenix High Performance Computing service at the University of Adelaide, an Australian Government Research Training Program (RTP) Scholarship, and by the Commonwealth of Australia (represented by the Defence Science and Technology Group) through a Defence Science Partnerships agreement.

References

[1] G. Daugherty, S. Reveliotis, and G. Mohler, "Optimized multiagent routing for a class of guidepath-based transport systems," *IEEE Transactions on Automation Science and Engineering*, 2018.

[2] K. Woiceshyn, Z. Kashino, G. Nejat, and B. Benhabib, "Vehicle routing for resource management in time-phased deployment of sensor networks," *IEEE Transactions on Automation Science and Engineering*, no. 99, pp. 1–13, 2018.

[3] D. Stavrou, S. Timotheou, C. G. Panayiotou, and M. M. Polycarpou, "Optimizing container loading with autonomous robots," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 717–731, 2018.

[4] D. Spensieri, J. S. Carlson, F. Ekstedt, and R. Bohlin, "An iterative approach for collision free routing and scheduling in multirobot stations," *IEEE Transactions on Automation science and Engineering*, vol. 13, no. 2, pp. 950–962, 2016.

[5] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. J. Kleywegt, S. Koenig, C. A. Tovey, A. Meyerson, and S. Jain, "Auction-based multi-robot routing," in *Robotics: Science and Systems*, vol. 5. Rome, Italy, 2005, Conference Proceedings, pp. 343–350.

[6] E. Schneider, O. Balas, A. T. Ozgelen, E. I. Sklar, and S. Parsons, "Evaluating auction-based task allocation in multi-robot teams," in *AAMAS Workshop: ARMS*, 2014, Conference Proceedings.

[7] G. P. Das, T. M. McGinnity, S. A. Coleman, and L. Behera, "A distributed task allocation algorithm for a multi-robot system in healthcare facilities," *Journal of Intelligent & Robotic Systems*, vol. 80, no. 1, pp. 33–58, 2015.

[8] H. Nazif and L. S. Lee, "Optimised crossover genetic algorithm for capacitated vehicle routing problem," *Applied Mathematical Modelling*, vol. 36, no. 5, pp. 2110–2117, 2012.

[9] L. P. Behnck, D. Doering, C. E. Pereira, and A. Rettberg, "A modified simulated annealing algorithm for SUAVs path planning," *IFAC-PapersOnLine*, vol. 48, no. 10, pp. 63–68, 2015.

[10] W. Wu, Y. Tian, and T. Jin, "A label based ant colony algorithm for heterogeneous vehicle routing with mixed backhaul," *Applied Soft Computing*, vol. 47, pp. 224–234, 2016.

[11] J. W. Escobar, R. Linfati, P. Toth, and M. G. Baldoquin, "A hybrid granular tabu search algorithm for the multi-depot vehicle routing problem," *Journal of Heuristics*, vol. 20, no. 5, pp. 483–509, 2014.

[12] H. A. Taha, *Integer programming: theory, applications, and computations*. Academic Press, 2014.

[13] K. Sundar and S. Rathinam, "Algorithms for heterogeneous, multiple depot, multiple unmanned vehicle path planning problems," *Journal of Intelligent & Robotic Systems*, vol. 88, pp. 513–526, 2016.
[14] M. Fink, G. Desaulniers, M. Frey, F. Kiermaier, R. Kolisch, and F. Soumis, "Column generation for vehicle routing problems with multiple synchronization constraints," *European Journal of Operational Research*, vol. 272, pp. 699–711, 2018.
[15] G. D'Urso, S. L. Smith, R. Mettu, T. Oksanen, and R. Fitch, "Multi-vehicle refill scheduling with queueing," *Computers and Electronics in Agriculture*, vol. 144, pp. 44–57, 2018.
[16] N. Agatz, P. Bouman, and M. Schmidt, "Optimization approaches for the traveling salesman problem with drone," *Transportation Science*, vol. 52, pp. 965–981, 2018.
[17] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, "The vehicle routing problem: State of the art classification and review," *Computers & Industrial Engineering*, vol. 99, pp. 300–313, 2016.
[18] H. Li, F. Nashashibi, and M. Yang, "Split covariance intersection filter: Theory and its application to vehicle localization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1860–1871, 2013.
[19] O. De Silva, G. K. Mann, and R. G. Gosine, "Efficient distributed multi-robot localization: A target tracking inspired design," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, Conference Proceedings, pp. 434–439.
[20] Y. Yan and Y. Mostofi, "Robotic router formation in realistic communication environments," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 810–827, 2012.
[21] Y. Pei, M. W. Mutka, and N. Xi, "Connectivity and bandwidth-aware real-time exploration in mobile robot networks," *Wireless Communications and Mobile Computing*, vol. 13, no. 9, pp. 847–863, 2013.
[22] V. Spirin, S. Cameron, and J. De Hoog, "Time preference for information in multi-agent exploration with limited communication," in *Conference Towards Autonomous Robotic Systems*. Springer, 2013, Conference Proceedings, pp. 34–45.
[23] J. Banfi, A. Q. Li, N. Basilico, and F. Amigoni, "Communication-constrained multirobot exploration: Short taxonomy and comparative results," in *Proceedings of the IROS workshop on on-line decision-making in multi-robot coordination (DEMUR2015)*, 2015, Conference Proceedings, pp. 1–8.
[24] J. Stephan, J. Fink, V. Kumar, and A. Ribeiro, "Concurrent control of mobility and communication in multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1248–1254, 2017.
[25] Y. Kantaros and M. M. Zavlanos, "Global planning for multi-robot communication networks in complex environments," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1045–1061, 2016.
[26] S. G. Manyam, S. Rathinam, S. Darbha, D. Casbeer, Y. Cao, and P. Chandler, "GPS denied UAV routing with communication constraints," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, pp. 691–703, 2016.
[27] Y. Wang and C. Hu, "Moving as a whole: multirobot traveling problem constrained by connectivity," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 23, no. 3, pp. 769–788, 2015.
[28] G. Dhein, A. F. K. Neto, and O. C. B. de Araújo, "The multiple traveling salesman problem with backup coverage," *Electronic Notes in Discrete Mathematics*, vol. 66, pp. 135–142, 2018.
[29] N. Sullivan, S. Grainger, and B. Cazzolato, "A dual genetic algorithm for multi-robot routing with network connectivity and energy efficiency," in *Control, Automation, Robotics and Vision (ICARCV) 2018. Proceedings. 2018 15th International Conference on*. IEEE, 2018, Conference Proceedings.
[30] MathWorks, "Traveling salesman problem: Solver-based," 2018, accessed: 2018-07-01. [Online]. Available: http://au.mathworks.com/help/optim/ug/travelling-salesman-problem.html
[31] M. Elango, S. Nachiappan, and M. K. Tiwari, "Balancing task allocation in multi-robot systems using K-means clustering and auction based mechanisms," *Expert Systems with Applications*, vol. 38, no. 6, pp. 6486–6491, 2011.
[32] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012.
[33] M. Stolpe and A. Bechmann, "A greedy algorithm for a special class of geometric set covering problems," *Departmen of Wind Energy Report*, 2012.
[34] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 231–247, 1992.
[35] N. Sullivan, S. Grainger, and B. Cazzolato, "Sequential single-item auction improvements for heterogeneous multi-robot routing," *Submitted to Journal of Robots and Autonomous Systems*, Submitted.
[36] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.
[37] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, *Path planning and trajectory planning algorithms: A general overview*. Springer, 2015, book section 1, pp. 3–27.
[38] T. Renato, K. Helsgaun, and D. Whitley, "Efficient recombination in the lin-kernighan-helsgaun traveling salesman heuristic," in *Parallel Problem Solving from Nature, 2018 International Conference on*. Springer, 2018, Conference Proceedings, pp. 95–107.
[39] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.
[40] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, Conference Proceedings, pp. 4193–4198.
[41] N. Sullivan, S. Grainger, and B. Cazzolato, "Analysis of cooperative localisation performance under varying sensor qualities and communication rates," *Robotics and Autonomous Systems*, 2018.

Nick Sullivan Nick Sullivan received his bachelor degrees in mechatronics engineering and computer science from the University of Adelaide, Australia, in 2016. He is currently working toward the Ph.D. degree in robotics at the University of Adelaide. His research interests include multi-robot task allocation, unmanned ground vehicles, and machine learning.

Steven Grainger Steven Grainger obtained his Ph.D. on the control of electric drives from Glasgow Caledonian University, Scotland and holds B.E. degrees in computing and electronic engineering. He is a lecturer in Control and Embedded Systems at the University of Adelaides School of Mechanical Engineering. His current research interests include nanopositioning systems and autonomous vehicles.

Ben Cazzolato Ben Cazzolato received his B.E. in mechanical engineering at the University of Adelaide, Australia, in 1990. At the same university, he received his Ph.D. in the field of active control for sound transmission in 1998. He is currently a professor at the University of Adelaide, teaching and researching in the fields of dynamics and control. Current research interests include modelling of complex electro-mechanical systems, control of unstable vehicles, active control and nano-positioning.

Chapter 7

Summary and Conclusion

Multiple robots can complete tasks faster and more robustly than a single robot. However, it is non-trivial to decide how robots should operate to complete tasks quickly, energy efficiently, and robustly. Algorithms for this problem are differentiated by the time it takes for them to allocate tasks, where fast allocations are best handled by sequential single-item auctions, optimal allocations are provided by integer linear programs, and a trade-off between the two are solved by metaheuristics.

In Chapter 3, adaptations to sequential single-item auctions have been provided, improving their performance. These adaptations use game theory to resolve auctions, resulting in reduced energy usage in heterogeneous systems, and reduced completion time for both heterogeneous and homogeneous systems. This has clear practical significance in that multi-robot systems are able to complete tasks with lower operational cost via reduced energy costs. It also raises questions about the application of algorithms designed for heterogeneous systems. It was found that the new algorithm designed to operate on heterogeneous systems also showed significant improvement for homogeneous systems, because their different locations makes them temporarily

heterogeneous. Applying this knowledge could improve other fields such as collision avoidance, formation control, and network routing.

In Chapter 4, support for robot tool changes has been developed for auctions and genetic algorithms, which allow these techniques to directly solve the problem. They have been empirically compared to an introduced transformation, which converts the tool-swap problem into a standard allocation problem. Guidance and mathematical formulation has been provided to illustrate when each technique performs best. The practical significance is that systems with robots that use tools now have more algorithms at their disposal, as well as guarantees on the solution quality. It was found that the type of capabilities (specific to a single task or useful for many tasks) have significant impact on the ease of solving the problem, with the former being easily transformed while maintaining the optimal solution, and the latter being far more difficult due to the number of new solutions. This prompts the need for further research on task allocation involving general-purpose tools. In particular, overcoming the local optima which limit heuristic and meta-heuristic algorithms.

Robots can improve their localisation capabilities by detecting one another in a process called collaborative localisation (CL). This provides a robust mechanism to help localise robots with broken sensors, and improve position estimates of groups of robots without GPS. In Chapter 5, the properties that improve CL performance have been analysed. It was found that yaw accuracy has a substantial effect, communication rate can be detrimental if it is too slow or fast, and heterogeneous systems are greater candidates for CL than homogeneous systems. These insights are useful for determining whether a given robotic system would be suitably improved by using CL, or how to design a robotic system such that CL can be effective. This analysis was performed using a standard CL algorithm, but it would be useful to perform similar analysis on new algorithms that operate differently. In particular, it is encouraged that researchers apply new CL algorithms on real systems to display both effectiveness and sensitivity.

To enable CL, it is necessary to consider inter-robot distance during task allocation. While some algorithms existed in the literature, they were unable to guarantee performance when applied to real systems. In Chapter 6, a new formation-based algorithm was presented, which provides inter-robot distance guarantees even in the presence of obstacles. It also scales far better with the number of robots and tasks than other available algorithms. This was used on a real system running CL, where one robot without GPS was able to be successfully localised thanks to another robot, and all tasks were completed successfully. This algorithm is also useful in systems which need robots to stay near each other to maintain network communication, or to collaborate to quickly complete a complex task. This algorithm has its limitations, and future research in this area is encouraged to help overcome them. In particular, research is encouraged on decentralised and distributed planning, dynamic environments (inner and outer control loops), 3D environments, and other motion models.

There are a number of open problems which would have considerable impact on robotic task allocation research. Allocating tasks for the most energy efficient completion is a linear problem, allowing researchers to draw on mathematical theory to develop provably good (and sometimes optimal) solutions. Allocating tasks for the fastest completion, however, is non-linear, and very little mathematical development is available for this problem. Research in this area relies on empirical testing to compare algorithms, and would benefit from any mathematical rigour that could be introduced. Collaborative localisation research is in its infancy, and very few algorithms have been implemented on real robots. The sensitivity of these algorithms to real world conditions (lost messages, delays, non-Gaussian noise) have not been considered. In addition, there has not been a comparison of the available algorithms, so it is unclear what their strengths and weaknesses are. A platform for this purpose has been developed and is described in Appendix A. Multiple metaheuristics are available for optimisation problems, but it is unclear which one is most suited for task allocation. Comparison is

difficult, as there are a lot of options for representation, initialisation, selection, and exploration/exploitation functions. Any empirical comparison between them would be useful, albeit difficult to generalise. It would be very useful if one could explain *why* one metaheueristic is more suitable for a given problem than another.

Appendix A

Multi-Robot Hardware Platform

The following conference paper details the creation of a robotic platform for collaborative localisation. This platform was developed to experimentally validate collaborative localisation algorithms. It is referenced in Chapter 6, where a task allocation algorithm was developed for systems with cooperative localisation capabilities.

Statement of Authorship

Paper Title:	An Outdoor Multi-Vehicle Platform for Collaborative Localisation Research
Status:	Presented at the Australasian Conference on Robotics and Automation (ACRA 2018)
Details:	Published in *ACRA 2018 Proceedings*, vol 20, 2018

Principal Author

Name:	Nick Sullivan
Contribution Details:	Programmed the robots to a working baseline state, where they could be driven manually. Added the functionality of autonomous waypoint following, camera calibration, marker detection, network communication, and data storage. Performed experiments. Post processed data to generate localisation estimates. Parsed and analysed results. Prepared the manuscript and generated all figures.
Contribution Percentage (%):	60
Signature:	Date: 17 Mar, 2019

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

1. the candidate's stated contribution to the publication is accurate (as detailed above);
2. permission is granted for the candidate to include the publication in the thesis; and
3. the sum of all co-author contributions is equal to 100% less the candidates stated contribution.

Name:	Glen Pearce
Contribution Details:	Programmed the robots to a working baseline state, where they could be driven manually. Created and installed the fiducial marker boxes and cameras. Performed experiments. Edited manuscript.
Signature:	Date: 13 Mar, 2019

Name:	Steven Grainger
Contribution Details:	Supervised work development and edited manuscript.
Signature:	Date: 15 Mar, 2019

Name:	Ben Cazzolato
Contribution Details:	Supervised work development and edited manuscript.
Signature:	Date: 13 Mar, 2019

An Outdoor Multi-Vehicle Platform for Collaborative Localisation Research

Nick Sullivan[1*], Glen Pearce[2], Steven Grainger[1], Ben Cazzolato[1]
[1]School of Mechanical Engineering, The University of Adelaide
[2]Defence Science and Technology Group
*nicholas.sullivan@adelaide.edu.au

Abstract

Autonomous vehicles rely on GPS to determine their location, but GPS may not always be available. A number of methods are being researched to localise vehicles in GPS-denied environments, such as collaborative localisation (CL), where robots measure their position relative to one another. However, CL algorithms are not often tested on real hardware. To accommodate this, we have upgraded commercial off-the-shelf Clearpath Jackals to have inter-robot detection capabilities. We present the hardware and software design of the vehicles, and our work flow to transition from simulations to live operation. We provide extensions to the available Clearpath Jackal simulation (in Gazebo), as well as contributions to Robot Operating System (ROS) packages. Preliminary results are presented for cases where a single vehicle with access to GPS uses inter-robot detections to localise other vehicles.

1 Introduction

Many industries are starting to use autonomous vehicles, such as transportation, warehousing, and assisted healthcare. As such, they are drawing significant research attention. One essential component for successful control of these vehicles is localisation. Vehicles must have some idea of where they are within their environment in order to effectively navigate to where they need to go. Multi-vehicle systems have the additional complexity that vehicles must evade and coordinate with one another. To perform localisation, outdoor vehicles typically rely on GPS availability or recognisable features for Simultaneous Localisation and Mapping (SLAM), with dead-reckoning being used to interpolate between readings as necessary. However, these are not always available. In particular, GPS is not available in tunnels, areas covered in large trees or buildings, or in warzones.

Figure 1: The physical robots equipped with cameras and fiducial markers for inter-robot detections, as well as a number of on-board sensors for localisation.

SLAM approaches use recognisable landmarks to generate and localise from a map. Ideally, available landmarks will be recognisable, precise, and invariant to rotation, translation and scale [Fuentes-Pacheco et al., 2015]. It is therefore a challenge to successfully localise robots in unpredictable, GPS-degraded environments [Ma et al., 2017].

One method for localising multiple vehicles is to use Collaborative Localisation (CL), which involves the communication of feature detections to improve the localisation of the group. Key advantages of CL is its ability to reduce dead-reckoning drift by averaging over a group [Rekleitis et al., 2002], and using robots as features, permitting the use of SLAM techniques when no other features are available. It can also be used to improve localisation of robots with limited sensing by leveraging the localisation of robots with superior sensing [Wanasinghe et al., 2015]. It has been used to localise smart cars in tunnels and urban canyons [Elazab et al., 2015], under-

water vehicles localising from a surface unit with GPS [Allotta et al., 2014], and improved 3D mapping of large buildings [Kurazume et al., 2016].

Both DST Group's Advanced Vehicle Systems team and the University of Adelaide are interested in developing and analysing multi-vehicle CL techniques to provide superior localisation over long distances in challenging environments. A first step in this endeavour is to provide simulated and real-life test beds to apply and test new algorithms.

A number of experimental robot platforms exist, and have been surveyed and categorised up to 2013 [Jimnez-Gonzlez et al., 2013; Parker, 2008]. More recently, test-beds have been developed for distributed monitoring of underwater environments [Schill et al., 2018], welding [Mendes et al., 2016], gait rehabilitation [Bayon et al., 2016], and multi-vehicle control [Labrado et al., 2016]. Of particular interest is the Robotarium [Pickem et al., 2016], which provides publicly available remote access to several small indoor robots. This lowers the barrier of entry for robotics research, allowing more people to contribute to the global pool of knowledge. Many useful tools would not exist if it were not for the sharing of software, such as OpenCV [Bradski and Kaehler, 2000], the Robot Operating System (ROS) [Quigley et al., 2009], and Gazebo [Koenig and Howard, 2004]. Our test-bed makes use of all these softwares, and we have contributed our improvements to the ROS repositories.

This paper details the physical and software design of our research platform. It includes the challenges that were overcome, as well as preliminary results showing the capabilities of the platform.

2 Applications

The primary objective of the presented test bed is for prototyping and validating collaborative localisation techniques. However, it will also be used for many other research applications.

The test-bed can be used to validate control algorithms where robots have limited vision angles, localisation inaccuracies, and inter-robot detection errors. In particular, simulations often simulate errors as zero-mean Gaussian noise. We would like to know if these techniques still perform in real-world conditions. Additionally, we are interested in control techniques that improve CL performance, i.e., moving vehicles in a way that increases the quantity and quality of inter-robot detections.

CL relies on detection of nearby robots. While we have elected to use visual detection providing both angle and distance, many other techniques are available, such as using acoustic signals [Allotta et al., 2014], radio signal direction [Russell et al., 2017], or infra-red [Rashid et al., 2015]. With accurate position information, we can test the performance of inter-robot detection techniques and produce more realistic error profiles for use in simulations.

In real conditions, network connectivity may become degraded. We wish to produce and analyse techniques that can operate even when communication is restricted or unavailable.

2.1 Objectives

A number of objectives have been designed for the test-bed in order for it to be used for the desired applications.

- Enable a quick transition from simulation to real robots.
- Measure localisation error by comparing vehicle position with centimetre accurate differential GPS (D-GPS).
- Provide customisable access to information. Vehicles may be given access to the D-GPS, a standard GPS, or no GPS at all. They may or may not be able to identify the ID of other vehicles during inter-robot detection. Detected landmarks may or may not have a known position.
- Provide independent operation, without reliance on full connectivity between robots.

To achieve these, we have elected to use ROS, which allows the same software to be used for simulation and on the real robots. In particular, initial work is implemented in Gazebo. We then collect live sensor data from the real vehicles using rosbags (message logging in ROS), and play back the data for simulation with real sensor data. Finally, the software is uploaded to the robots to operate in real-time on the real system.

3 Physical Design

The test-bed uses Clearpath Jackals [Clearpath, 2014], which support ROS and provide a readily available Gazebo plug-in for a single Jackal. Adaptations had to be made for a multi-Jackal simulation, which are described in Section 4. All components operate using ROS in both the real robot and Gazebo simulation.

The flow diagram for the physical system is shown in Figure 2. In the first stage, sensors are processed into a usable form. The sensor data can optionally be logged to develop algorithms in the lab using real data. When executing on real robots, the experimental software will use live sensor measurements to produce an output. In the figure, we have an example CL algorithm, which communicates inter-robot detections to produce new pose estimations. The output is then stored in a rosbag, which is converted into CSV files for analysis.

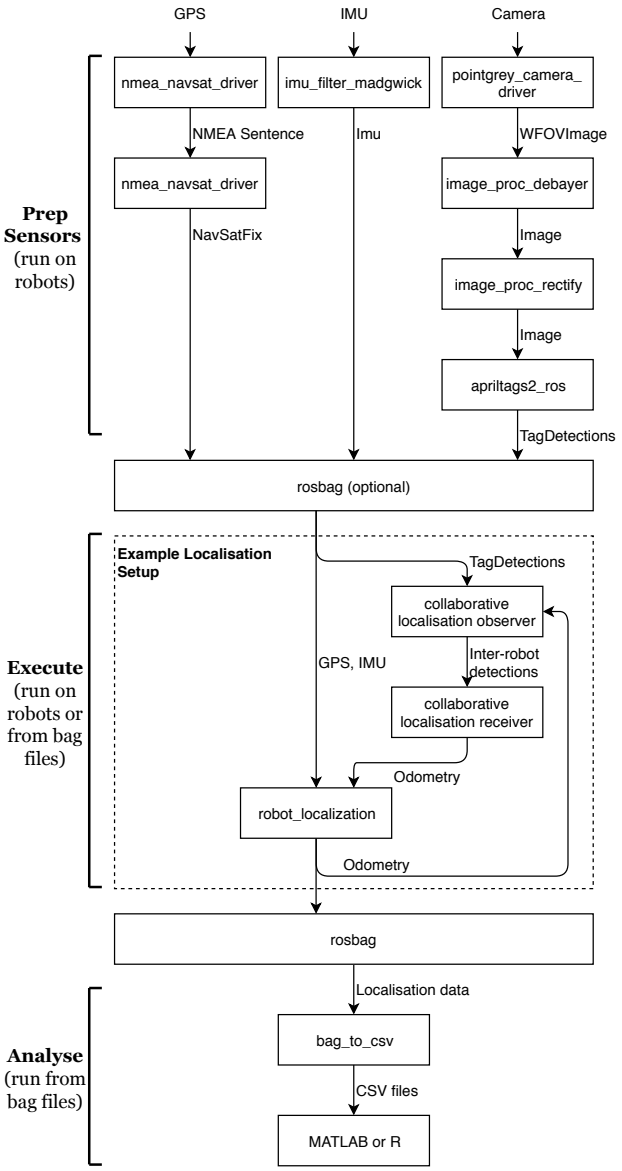


Figure 2: The software flow for the vehicles, run in the Robot Operating System (ROS). Sensors are pre-processed into a usable form, then optionally logged in a rosbag (not all are listed). The robots execute the currently uploaded software, and store the results in a rosbag. The results are then analysed using MATLAB.

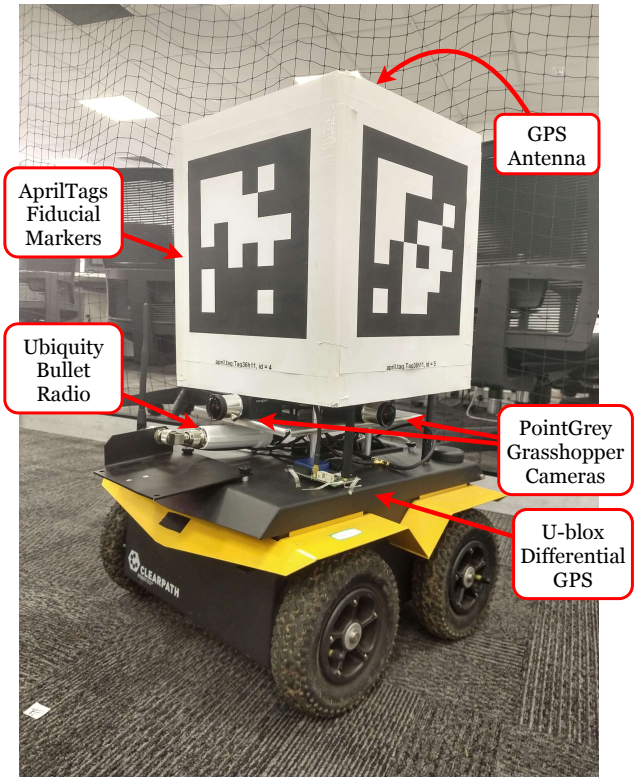


Figure 3: The physical robot.

3.1 Sensing

Many of the sensors can be seen in Figure 3. The Jackals contain an IMU consisting of an accelerometer, gyroscope, and magnetometer. The IMU is calibrated to calculate magnetic disturbances of the vehicle.

Each robot has four Pointgrey Grasshopper3 cameras separated by 90°. Their focal length was set to infinity. These cameras have a maximum resolution of 2048 x 2048 pixels, with a maximum of 90 Hz. The cameras were calibrated using the standard *camera_calibration* node in ROS. By default, they alter their exposure based on lighting conditions. We found that increasing the exposure lessened the affects of different lighting conditions, particularly shadows.

Each robot was also mounted with a large box containing fiducial markers on each side. The markers are AprilTags markers [Wang and Olson, 2016]. The robots use the *apriltags2_ros* package to identify robots and measure the distance and angle between the robots, as well as the orientation of the observed robot. The detection range of AprilTags was tested on a sunny day by holding a marker and moving backwards until detections stopped, then moving a step forward to get the range. The average error distance errors are listed in Table 1.

The GPS and differential GPS share an antenna, with the signals being split using a splitter cable. We do this

Table 1: Apriltags2 detection ranges in sunny weather.

Resolution (pix)	Size (cm)	Range (m)	Error (%)
1280x1024	11.1	7.5	1.3
1280x1024	16.6	13.0	0.8
2048x2048	11.1	9.0	1.1
2048x2048	16.6	15.0	3.3

because the on-board GPS antenna has difficulty finding signal after the Jackal has been mounted with other components. The U-blox differential GPS receives position correction information from a base station to produce centimetre level accuracy. These sensors output three strings, providing position information and GPS fix quality, velocity information, and covariance estimations. These strings are fed into ROS using the *nmea_navsat_driver* package. This driver originally used covariance approximation that does not correctly represent correction signals e.g. centimetre accurate positions were creating ROS messages with covariance values hundreds of times larger than they should be. We made some corrections to the driver and have merged our changes into the maintained ROS repository.

3.2 Control

Each Jackal comes with a bluetooth connection to a Playstation 4 controller, providing manual movement with a dead-mans switch. When no movement commands are received the Jackal stops.

Alternatively, the vehicles can travel to GPS waypoints using *robot_localization*'s *navsat_transform_node*. The *navsat_transform_node* converts GPS coordinates into map coordinates, which the robots move to using a simple fixed-speed waypoint follower.

3.3 Networking

The robots communicate to one another using long Ubiquity bullet radios, with a base station that operates as a wireless access point. For long-distance communications, they are capable of supporting point-to-point communication and forwarding messages in a decentralised network. A communication graph can be manually specified by blacklisting certain connections.

By default, ROS uses a single master to organise communication between machines. This would produce a reliance on connectivity between robots, which violates our objective of testing decentralised algorithms. Instead, the robots use a multi-master ROS setup with *multimaster_fkie*, where topics are synchronised across ROS masters. ROS masters are dynamically discovered, and will only send messages that robots have announced interest in receiving. In this setup, no synchronised ROS nodes can have the same name, otherwise there will be a conflict and cause one of the nodes to crash. As such, we uniquely namespace (i.e. prefixed with their ID) all nodes that are involved with inter-robot communication.

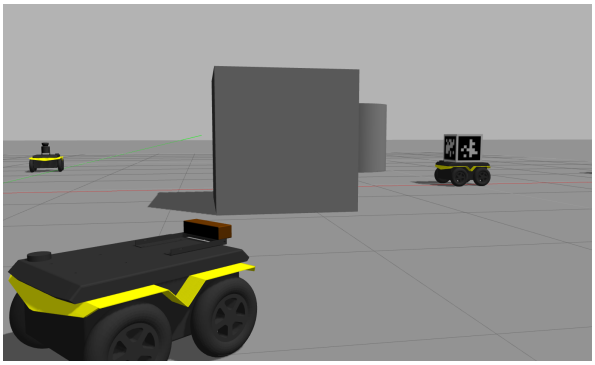


Figure 4: A screenshot from the multi-Jackal simulation in Gazebo. The robots can have cameras, LIDARs, and fiducial markers mounted on them.

Inter-robot detections provide the distance, angle, and orientation of an observed robot's marker frame relative to an observing robots camera frame. For this information to be used in localisation algorithms, it should be between each robot's centre of mass, known in ROS as the robot's *base_link*. The inter-robot detection uncertainty is manipulated using the *pose_cov_ops* package, which uses the Mobile Robot Programming Toolkit (MRPT) [Claraco, 2008].

4 Simulation Design

For simulation, Gazebo is used [Koenig and Howard, 2015]. Simulation of a single Jackal in Gazebo was already available in ROS, but it could not be used to simulate multiple Jackals, due to topic and frame naming collisions. We have developed new packages for simulating multiple Jackals that overcome these pitfalls. This is publicly available on ROS Kinetic in the package *multi_jackal_tutorials*. The option to add a fiducial marker box has also been made available. An image of the simulation can be seen in Figure 4. The launching process matches that of a single Jackal. It is separated into control, navigation, and description nodes. A base combines these components, and tutorials show how to use the base for a variety of scenarios.

5 Localisation

The robots are able to detect their relative positions from one another. This information can be used to improve localisation of the group. To localise, we use an Extended Kalman Filter (EKF). In particular, we make use of the package *robot_localization*, which provides an EKF in ROS.

At time k, each robot stores a state estimate $\hat{\mathbf{x}}_{k|k}$ and a state covariance $\mathbf{P}_{k|k}$, which contain the estimated pose

and velocity, as well as the uncertainty of these estimations. The EKF performs two phases; *prediction* and *update*. The prediction phase calculates the robot state based on its prior state:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} \tag{1}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k \tag{2}$$

The update phase uses sensor measurements $\mathbf{z}_k$ with covariance $\mathbf{R}_k$ and observation matrix $\mathbf{H}_k$ to improve the state:

$$\text{Innovation:} \quad \tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \tag{3}$$

$$\text{Innovation Covariance:} \quad \mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \tag{4}$$

$$\text{Kalman Gain:} \quad \mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1} \tag{5}$$

$$\text{Updated State:} \quad \hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \tag{6}$$

$$\text{Updated Covariance:} \quad \mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \tag{7}$$

The EKF state contains the robot's pose (x, y, ψ) relative to a static origin, and its velocity on the body-centred frame (x is forward, y is left). The filter is discrete, and uses the time between iterations dt. For shorthand, we let $c_\psi = \cos(\psi)$ and $s_\psi = \sin(\psi)$.

$$\hat{\mathbf{x}} = \begin{bmatrix} \text{pos}_x \\ \text{pos}_y \\ \text{pos}_\psi \\ \text{vel}_x \\ \text{vel}_y \\ \text{vel}_\psi \end{bmatrix}$$

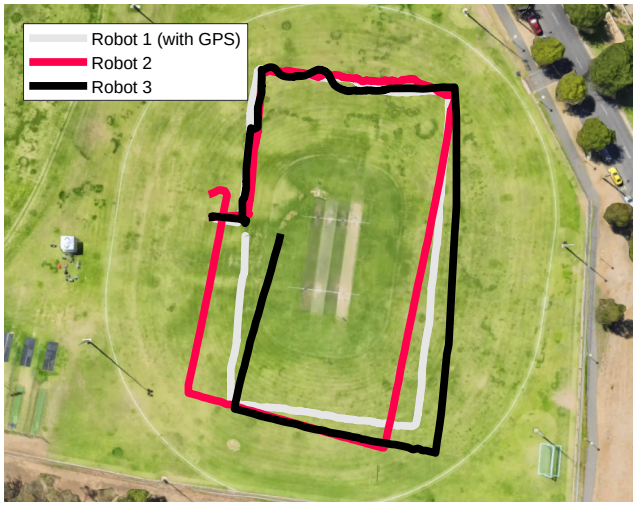
$$\mathbf{Q} = \begin{bmatrix} 0.05 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.05 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.06 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.025 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.025 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.02 \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & dt(-\text{vel}_x s_\psi - \text{vel}_y c_\psi) & dt(c_\psi) & dt(-s_\psi) & 0 \\ 0 & 1 & dt(-\text{vel}_y s_\psi + \text{vel}_x c_\psi) & dt(s_\psi) & dt(c_\psi) & 0 \\ 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The matrices used in *robot_localization* include accelerations, pitch, roll, and height. These are not utilised in our implementation so have been omitted here.

Three jackals were driven in a rectangle on a sports pitch, as shown in Figure 5, with an objective to work out their location. One robot had access to GPS for the entire trial, and the two others were denied GPS after convergence. Two methods were compared. Singular

(a) Singular Localisation



(b) Collaborative Localisation

Figure 5: Three robots were driven along the same path, in a rectangle. Two robots were denied access to GPS during operation. a) The position estimates of the robots using dead-reckoning from wheel encoders, gyroscope and accelerometers. b) The position estimates of the robots using all the sensor data from (a) as well as inter-robot detections, where the single robot with GPS is able to assist the other two robots.

localisation used encoders, gyroscope, and control input. We can see that this leads to localisation drift over time. Collaborative localisation also used encoders, gyroscope, and control input, but with inter-robot measurements where available. We can see that the robot with GPS is able to correct the localisation drift of the other robots.

We have fused inter-robot measurements in a method known as Naive Collaborative Localisation, because an EKF treats all inputs as independent, and this is not the case when robots use communicated information. When one robot affects another robot's location estimate, the two are no longer completely independent, resulting in the same source of information being fused more than once in the EKF. While this can be somewhat accounted for by tuning communication rate and process noise covariance [Sullivan et al., 2018a], this platform enables us to compare the performance of collaborative localisation algorithms that specifically deal with this problem, such as Covariance Intersection [De Silva et al., 2015], Split Covariance Intersection [Li et al., 2013], the Common Past-Invariant Ensemble Kalman Filter [Curn et al., 2013], the Cubature Kalman Filter [Wanasinghe et al., 2015], and Posterior Linearization Belief Propagation [Garcia-Fernandez et al., 2018].

6 Vehicle Routing

Localisation is not the primary objective of a multi-robot system. Rather, it is part of the means to complete certain tasks. It is therefore not practical to expect robots to drive behind one another in order to maintain collaborative localisation. Instead, we wish to use routing algorithms that aim to complete primary objectives (tasks at given locations), while enabling collaborative localisation. Such algorithms include market-based auction [Wang and Hu, 2015], genetic algorithms [Dhein et al., 2018; Sullivan et al., 2018b], and formation routing [Sullivan et al., 2018c]. Note that this section provides an example on how the platform is being utilised for testing routing algorithms. Further development and validation of vehicle routing algorithms will be discussed in future work.

We placed 30 targets in a 50 m x 50 m area, and ran a formation routing algorithm to visit all target locations while maintaining distance of under 20 m. Beyond 20 m, robots receive fewer visual detections, which would prevent the use of collaborative localisation. The formation routing steps are as follows:

1. Group targets into circles of 20 m diameters using K-means clustering
2. Order the circles using a Travelling Salesman Problem solver [MathWorks, 2018]
3. Within a circle, allocate targets using sequential auction

(a) Singular Localisation

(b) Collaborative Localisation

Figure 6: The objective is for robots to visit all target locations and return to the start in the fastest time. One robot has GPS, while the other does not. a) Without GPS, one of the robots drifts. b) Inter-robot detections allow both robots to visit their targets.

4. Between circles, move in formation
5. Repeat steps 3 and 4 until vehicles are back to their starting points

The resulting vehicle paths can be seen in Figure 6. One robot is given access to its GPS, while the other is not. The objective is for all targets to be visited by a robot. If they both use their own sensing, the robot without GPS is unable to properly move to each target location, as seen in Figure 6a. If the robot with GPS shares position information using collaborative localisation, both robots are able to navigate to the target locations, as seen in Figure 6b. The formation routing technique ensures the robots stay within visual detection distance, allowing CL to be used throughout the entire trial.

7 Conclusion

The upgraded Clearpath Jackals provide a suitable research platform to develop and test localisation algorithms. It provides a means to transfer algorithms from simulation to reality, with options to operate in a Gazebo environment; a rosbag of real sensor data; and finally live operation. We presented successful cases where collaborative localisation has been used to localise vehicles without GPS by leveraging information from those that do. In the development process, contributions to ROS GPS drivers have been made to support D-GPS, and multi-Jackal simulations have been made available in ROS. This platform will provide an important role in verifying the performance of localisation algorithms, and will help transition them from simulation to reality.

Acknowledgment

This research was supported by an Australian Government Research Training Program (RTP) Scholarship, and by the Commonwealth of Australia (represented by the Defence Science and Technology Group) through a Defence Science Partnerships agreement.

Special thanks to Anthony Perry and Daniel Nesbitt for their assistance in upgrading the Clearpath Jackals.

References

[Allotta et al., 2014] Allotta, B., Costanzi, R., Meli, E., Pugi, L., Ridolfi, A., and Vettori, G. (2014). Cooperative localization of a team of AUVs by a tetrahedral configuration. *Robotics and Autonomous Systems*, 62(8):1228–1237.

[Bayon et al., 2016] Bayon, C., Ramirez, O., Velasco, M., Serrano, J., Lara, S. L., Martinez-Caballero, I., and Rocon, E. (2016). Pilot study of a novel robotic platform for gait rehabilitation in children with cerebral palsy. In *Biomedical Robotics and Biomechatronics (BioRob), 2016 6th IEEE International Conference on*, pages 882–887. IEEE.

[Bradski and Kaehler, 2000] Bradski, G. and Kaehler, A. (2000). OpenCV. *Dr. Dobbs Journal of Software Tools*.

[Claraco, 2008] Claraco, J. L. B. (2008). Development of scientific applications with the mobile robot programming toolkit. *The MRPT reference book. Machine Perception and Intelligent Robotics Laboratory, University of Malaga, Malaga, Spain*.

[Clearpath, 2014] Clearpath (2014). Jackal, unmanned ground vehicle. [Online; accessed 16-April-2018].

[Curn et al., 2013] Curn, J., Marinescu, D., O'Hara, N., and Cahill, V. (2013). Data incest in cooperative localisation with the common past-invariant ensemble Kalman filter. In *16th International Conference on Information Fusion (FUSION)*, pages 68–76. IEEE.

[De Silva et al., 2015] De Silva, O., Mann, G. K., and Gosine, R. G. (2015). Efficient distributed multi-robot localization: A target tracking inspired design. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 434–439. IEEE.

[Dhein et al., 2018] Dhein, G., Neto, A. F. K., and de Araújo, O. C. B. (2018). The multiple traveling salesman problem with backup coverage. *Electronic Notes in Discrete Mathematics*, 66:135–142.

[Elazab et al., 2015] Elazab, M., Noureldin, A., and Hassanein, H. S. (2015). Integrated cooperative localization for connected vehicles in urban canyons. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE.

[Fuentes-Pacheco et al., 2015] Fuentes-Pacheco, J., Ruiz-Ascencio, J., and Rendón-Mancha, J. M. (2015). Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 43(1):55–81.

[Garcia-Fernandez et al., 2018] Garcia-Fernandez, A. F., Svensson, L., and Sarkka, S. (2018). Cooperative localization using posterior linearization belief propagation. *IEEE Transactions on Vehicular Technology*, 67(1):832–836.

[Jimnez-Gonzlez et al., 2013] Jimnez-Gonzlez, A., Martinez-de Dios, J. R., and Ollero, A. (2013). Testbeds for ubiquitous robotics: A survey. *Robotics and Autonomous Systems*, 61(12):1487–1501.

[Koenig and Howard, 2004] Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE.

[Koenig and Howard, 2015] Koenig, N. and Howard, A. (2015). Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE.

[Kurazume et al., 2016] Kurazume, R., Oshima, S., Nagakura, S., Jeong, Y., and Iwashita, Y. (2016). Automatic large-scale three dimensional modeling using cooperative multiple robots. *Computer Vision and Image Understanding*, 0:1–18.

[Labrado et al., 2016] Labrado, J. D., Erol, B. A., Ortiz, J., Benavidez, P., Jamshidi, M., and Champion, B. (2016). Proposed testbed for the modeling and control of a system of autonomous vehicles. In *System of Systems Engineering Conference (SoSE), 2016 11th*, pages 1–6. IEEE.

[Li et al., 2013] Li, H., Nashashibi, F., and Yang, M. (2013). Split covariance intersection filter: Theory and its application to vehicle localization. *IEEE Transactions on Intelligent Transportation Systems*, 14(4):1860–1871.

[Ma et al., 2017] Ma, K., Schirru, M., Zahraee, A. H., Dwyer-Joyce, R., Boxall, J., Dodd, T. J., Collins, R., and Anderson, S. R. (2017). Pipeslam: Simultaneous localisation and mapping in feature sparse water pipes using the rao-blackwellised particle filter. In *Advanced Intelligent Mechatronics (AIM), 2017 IEEE International Conference on*, pages 1459–1464. IEEE.

[MathWorks, 2018] MathWorks (2018). Traveling salesman problem: Solver-based. Accessed: 2018-07-01.

[Mendes et al., 2016] Mendes, N., Neto, P., Simao, M., Loureiro, A., and Pires, J. (2016). A novel friction stir welding robotic platform: welding polymeric materials. *The International Journal of Advanced Manufacturing Technology*, 85(1-4):37–46.

[Parker, 2008] Parker, L. E. (2008). *Multiple mobile robot systems*, book section 40, pages 921–941. Springer.

[Pickem et al., 2016] Pickem, D., Glotfelter, P., Wang, L., Mote, M., Ames, A., Feron, E., and Egerstedt, M. (2016). The robotarium: A remotely accessible swarm robotics research testbed. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1699–1706. IEEE.

[Quigley et al., 2009] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, volume 3, page 5.

[Rashid et al., 2015] Rashid, A. T., Frasca, M., Ali, A. A., Rizzo, A., and Fortuna, L. (2015). Multi-robot localization and orientation estimation using robotic cluster matching algorithm. *Robotics and Autonomous Systems*, 63:108–121.

[Rekleitis et al., 2002] Rekleitis, I. M., Dudek, G., and Milios, E. E. (2002). Multi-robot cooperative localization: a study of trade-offs between efficiency and accuracy. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2690–2695. IEEE.

[Russell et al., 2017] Russell, J. S., Ye, M., Anderson, B. D., Hmam, H., and Sarunic, P. (2017). Cooperative localisation of a GPS-denied UAV in 3-dimensional space using direction of arrival measurements. *IFAC-PapersOnLine*, 50(1):8019–8024.

[Schill et al., 2018] Schill, F., Bahr, A., and Martinoli, A. (2018). *Vertex: A New Distributed Underwater Robotic Platform for Environmental Monitoring*, book section 7, pages 679–693. Springer.

[Sullivan et al., 2018a] Sullivan, N., Grainger, S., and Cazzolato, B. (2018a). Analysis of cooperative localisation performance under varying sensor qualities and communication rates. *Journal of Robotics and Autonomous Systems*, 110:73–84.

[Sullivan et al., 2018b] Sullivan, N., Grainger, S., and Cazzolato, B. (2018b). A dual genetic algorithm for multi-robot routing with network connectivity and energy efficiency. In *Control, Automation, Robotics and Vision (ICARCV) 2018. Proceedings. 2018 15th International Conference on*. IEEE.

[Sullivan et al., 2018c] Sullivan, N., Grainger, S., and Cazzolato, B. (2018c). Multirobot routing with connectivity maintainence. *Submitted to the European Journal of Operational Research*.

[Wanasinghe et al., 2015] Wanasinghe, T. R., Mann, G. K., and Gosine, R. G. (2015). Distributed leader-assistive localization method for a heterogeneous multirobotic system. *IEEE Transactions on Automation Science and Engineering*, 12(3):795–809.

[Wang and Olson, 2016] Wang, J. and Olson, E. (2016). Apriltag 2: Efficient and robust fiducial detection. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4193–4198. IEEE.

[Wang and Hu, 2015] Wang, Y. and Hu, C. (2015). Moving as a whole: multirobot traveling problem constrained by connectivity. *Turkish Journal of Electrical Engineering & Computer Sciences*, 23(3):769–788.

Appendix B

Task Allocation with Network Connectivity

The following conference paper introduces a new algorithm that allocates tasks with consideration of network connectivity between robots. A genetic algorithm is used to search for solutions to the problem, then waypoints are iteratively added to improve performance. This algorithm solves the same problem as the algorithm introduced in Chapter 6, with an application of network connectivity instead of visual connectivity. As discussed in that chapter, this algorithm performs better for small problem sizes, but does not scale well. The uniqueness of this algorithm lies in its ability to provide a range of solutions for two opposing objectives, leaving a decision-maker with choices about what they consider a balance between the two objectives.

Statement of Authorship

Paper Title:	A Dual Genetic Algorithm for Multi-Robot Routing with Network Connectivity and Energy Efficiency
Status:	Presented at the International Conference on Control, Automation, Robotics and Vision (ICARCV 2018)
Details:	Published in *ICARCV 2018 Proceedings*, vol 15, 2018

Principal Author

Name:	Nick Sullivan
Contribution Details:	Performed literature review on algorithms for allocating tasks to robots while maintaining distance between the robots, separating them by their consideration of multiple tasks, speed, and connectivity guarantees. Implemented algorithms from literature, discovering the violation of connectivity when collision avoidance is included. Developed a new algorithm that uses genetic algorithms to provide guarantees even when obstacles are included. Created tests to illustrate the performance of the new algorithm relative to those in literature, then wrote the code for these tests. Parsed and analysed results. Prepared the manuscript and generated all figures.
Contribution Percentage (%):	80
Signature:	Date: 17 Mar, 2019

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

1. the candidate's stated contribution to the publication is accurate (as detailed above);
2. permission is granted for the candidate to include the publication in the thesis; and
3. the sum of all co-author contributions is equal to 100% less the candidates stated contribution.

Name:	Steven Grainger
Contribution Details:	Supervised work development and edited manuscript.
Signature:	Date: 15 Mar, 2019

Name:	Ben Cazzolato
Contribution Details:	Supervised work development and edited manuscript.
Signature:	Date: 13 Mar, 2019

A Dual Genetic Algorithm for Multi-Robot Routing with Network Connectivity and Energy Efficiency

Nick Sullivan, Steven Grainger, Ben Cazzolato

Abstract— **We provide a Dual-GA technique for solving the Multiple Travelling Salesman Problem (mTSP) while constraining distance between robots. Other techniques primarily solve for full network connectivity, with energy efficiency as a secondary objective. Our technique makes no assumptions about the desired balance between connectivity and energy efficiency. Instead, it produces a range of solutions for the decision-maker to select from. It uses NSGA-II for the primary GA, with a secondary GA periodically adding waypoints for greater connectivity. We introduce the Dual-GA and analyse its performance compared to other algorithms.**

I. Introduction

Multi-robot systems can provide greater performance over single-robot systems through independent movement, redundancy, and heterogeneity. They are also becoming increasingly ubiquitous as the performance of robotic components improve, and their costs decrease. But multi-robot systems are more difficult to control, leading to significant research attention in the areas of flocking, exploration, and routing.

Multi-robot routing is the selection of how robots should move in order to complete desired tasks. It is also known as the Multiple Travelling Salesman Problem (mTSP) and the Multiple Depots Vehicle Routing Problem (MDVRP). The majority of research in this area is focussed on three aspects of these problems: completing a subset of tasks to maximise a pay-off metric, completing tasks energy efficiently, and completing tasks quickly. However, there is also a strong desire to consider new objectives for these problems [1].

Optimal solutions are available for the energy efficiency objective with the use of Mixed-Integer Linear Programs (MILPs) [2], and fast heuristic methods can be used while providing proven upper bounds [3].

The fast completion objective is non-linear, so proving solution quality is much more difficult. Few algorithms with tight mathematical guarantees are available for this objective. Nevertheless, techniques such as sequential single-item auctions and Genetic Algorithms (GAs) can provide fast and reliable solutions. There has also been research in solving multiple objectives for the mTSP. One of the most common approaches for multi-objective problems is NSGA-II [4], where solutions are sorted into Pareto fronts and provide a set of possible solutions that a user can select from.

A common assumption in these systems is that robots have the ability to communicate with one another at all times. While this is valid for some systems, it is not always the case for scenarios such as exploration, search and rescue, and environmental monitoring. As such, many available algorithms are not applicable to these cases. To overcome this, connectivity can be considered during the solving process.

Some research has focussed on optimal placement of robots to provide the best possible connectivity [5]. Another metric is to consider bandwidth as well as connectivity for streaming video while exploring [6]. Or the use of hybrid control systems to combine the benefits of a globally-optimal centralised system and a distributed system using inner and outer control layers [7], [8]. However, these techniques do not consider energy efficiency or completion time of multiple tasks.

Another approach is to plan paths assuming that robots can only communicate when they occupy the same space, and so forms paths that are guaranteed to overlap when looped infinitely [9]. Multi-robot exploration has also been considered, with the requirement that robots need only be connected when exploring new areas [10]. Tuning rules can be used to adjust how much information can be collected before reporting it back to a base station [11]. A comparison of four communication-based exploration techniques is available with a short taxonomy of different communication-constrained exploration [12].

Some heuristics consider energy efficiency as an objective, attempting to complete tasks as energy efficiently as possible while maintaining connectivity. One such heuristic is the Connected Nearest Neighbour (CNN) [13], which alternates between task allocation and connectivity maintenance phases. A bounded solution has been found for a two-robot problem where robots alternate movement on a small grid [14]. All tasks can be completed within 9/2 of optimal energy efficiency while maintaining connectivity, but under the limiting conditions of a small grid with alternating movements.

Thus far, state-of-the-art of connectivity-constrained mTSP has dealt with connectivity without consideration of energy efficiency, or energy efficient approaches that have best-effort connectivity. However, it may be desirable to sacrifice connectivity to improve energy efficiency, or vice versa. Our technique provides a *range* of solutions that a decision-maker can then select from. In this paper we consider a trade-off between energy efficiency and connectivity. A decision-maker can then make an informed decision about what they consider a desirable balance between these objectives. We also show that our technique outperforms other approaches when the aim is optimal connectivity, and analyse the conditions for when this is the case.

School of Mechanical Engineering, The University of Adelaide, 5005, Australia nicholas.sullivan@adelaide.edu.au

II. Problem Statement

In the multiple Travelling Salesman Problem (mTSP), there are a set of robots initially located at one or more depots, and a set of tasks at different locations that must be completed. Each robot will travel to the tasks allocated to it, known as the robot's tour. Our intent is to find allocations that minimise a given objective. Typical objectives are to minimise the total task completion time, which is done by minimising the maximum robot tour (MiniMax), and to minimise the total energy usage, which is done by minimising the sum of robot tours (MiniSum). When solving for these objectives, the cost function to traverse an edge is constant. The time that a robot takes travelling from one task to another is not affected by the previous tasks it has completed, nor is it affected by what the other robots are doing. However, when considering connectivity between robots, this is no longer the case. Robot path costs cannot be calculated independently, and the physical movement that robots take between two tasks now needs to be considered.

Let us consider how this affects solving scalability for a single-robot mTSP (i.e. a TSP) with N tasks. There are N choices for the first task, $N-1$ choices for the second, $N-2$ for the third etc. resulting in $N!$ possible allocations. If there are M options for how the robot travels between two tasks, there are M options between tasks 1 and 2, M options between tasks 2 and 3 etc. The number of possible allocations becomes $N! * M^N$. Task-to-task paths are usually calculated first and stored in a lookup table, resulting in a once off cost of $O(N^2)$. Task-to-task paths that consider connectivity must be calculated dynamically, so each allocation requires traversing the path to calculate its cost. This results in a final time of $O(N! * M^N * N)$, compared to the standard TSP cost of $O(N!)$. The value of M depends on how many task-to-task paths we wish to consider, these paths could be any number of different shapes, and go for any amount of time. This is a phenomenally large increase to an already NP-hard problem. As such, we instead only use the fastest task-to-task paths to reduce the scaling to a more feasible $O(N! * N)$.

A. Motivating Example

Consider a natural environment that is being researched, where soil samples from different locations must be collected regularly. Samples are collected from within a 10km by 10km area, so robots are sent every week to autonomously collect these samples. The robots can communicate via radio, but the range is limited to 3km. There are mountains which block robot movement and communication. The robots also have a risk of breaking down during collection. The researchers have two main aims for the sample collection robots:

- Energy efficient. Saving fuel saves money.
- Able to dynamically re-plan. In the event of a robot failure, the remaining robots should detect this failure, re-plan to collect the remaining soil samples, and report the location of the broken robot when they return home.

It is also desirable for the robots to complete the collections quickly, but we do not treat this as an objective.

TABLE I
Objectives.

Desire	Objective	Meaning
Energy efficient	MiniSumTime	Minimise the sum of all travel times.
Fast completion	MiniMaxTime	Minimise the maximum robot travel time.
Periodic connectivity	MiniMaxNetGap	Minimise the longest time between full robot connectivity.

III. Objective Functions

A number of objectives are explored, listed in Table I. The traditional objectives, energy efficient completion and fast completion, are referred to as MiniSumTime and MiniMaxTime to avoid confusion with our new objective.

We consider a new objective designed for regular network connectivity between robots. MiniMaxNetGap focuses on regular check-ins between robots, so that if a robot fails, the robots can re-plan accordingly.

Calculating SumTime and MaxTime for a given solution is simple, and can be done quickly from the cost matrix. This is because the edge cost for each robot is independent of all other edges being used. However, MaxNetGap cannot be calculated this way, as connectivity is determined by what all robots are doing. Instead, we iteratively simulate the robots along their proposed paths, and calculate connectivity based on inter-robot distance and obstacles. We use distance-based connectivity, but any connectivity function can be used.

IV. Solution Technique

To solve problems such as those listed in Section II, we need a set of solutions that meet the objectives. This then allows a higher-level planner such as a human operator or task allocation system to select a solution depending on their needs. Our solution technique follows several steps. Firstly, paths are found between all relevant locations. A multi-objective Genetic Algorithm (GA) known as the *Primary GA* is applied to produce a number of solutions. Waypoints are iteratively added and a single-objective *Secondary GA* is applied. We refer to this process as a *Dual GA*.

A. Path-finding

To calculate paths between each vertex (robot start position or task location), we make use of MATLAB's Robotics System Toolbox. A probabilistic roadmap (PRM) is formed, from which paths between vertices are calculated and stored in a path lookup table. Figure 1 shows an of example of paths between vertices.

B. Primary Genetic Algorithm

A solution is an allocation of tasks that result in all tasks being completed. A Genetic Algorithm (GA) searches for solutions by breeding and evolving the existing solutions. Solutions are encoded as genomes. We represent the genomes as two-part chromosomes [15], as outlined in Figure 2.

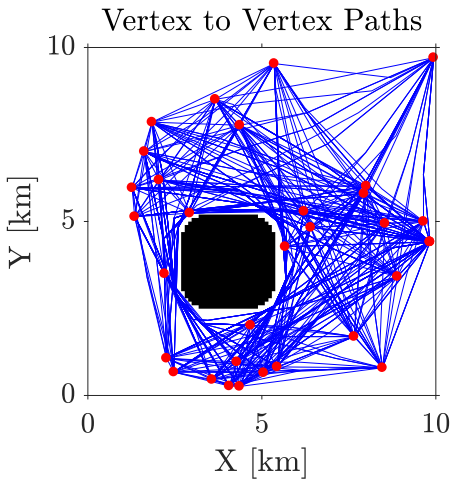


Fig. 1. An example of the pathfinding process. White parts are traversable, and black parts are not. Each vertex (red circle) represents a robot's start position or task location. Paths between vertices are calculated using a probabilistic roadmap (PRM).

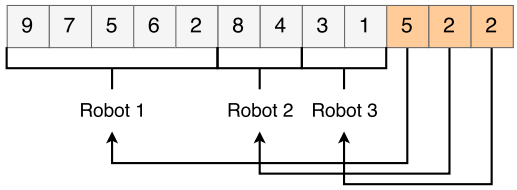


Fig. 2. The genetic algorithm representation for the Multiple Travelling Salesman Problem. The first part specifies task order. The second part specifies path length. In this example, robot 1 will complete tasks 9, 7, 5, 6, 2. Robot 2 will complete tasks 8, 4. Robot 3 will complete tasks 3, 1.

A grouping of genomes represents a population. Parents are selected from the population using weighted random selection, where the fittest are most likely to be selected. Children are created from their parents using crossover and mutation functions. The children are then added to the population as a steady-state process, removing the worst of the population. The concept is that genes which are the fittest will survive and breed, and we remove the least fit.

The initial population is created through randomisation, as well as valid solutions from sequential single-item auctions. Sequential single-item auctions are quick greedy heuristics that form solutions through task bidding. We have used a similar process previously [16].

For crossover, we use the two-part chromosome crossover (TCX) [17], designed for the mTSP. For mutation, we use external swap, where two tasks belonging to two robots are swapped, and path reverse, where a subpath belonging to one robot is reversed.

The primary GA is multi-objective, so we make use of the popular NSGA-II [4]. NSGA-II operates by sorting solutions into Pareto fronts. The first Pareto front contains all solutions for which there is no other solution that *dominates* it. A solution dominates another if it is better for at least one objective, but not worse for any objective. An example of a population that has been sorted into Pareto fronts is shown in Figure 3. Populations are first ranked by which front they are in, with ties being resolved by *crowding distance*. Crowding distance is a measure of uniqueness to similar solutions, and is used to promote diversity within the population. The pseudo-code for calculating Pareto fronts and crowding distances are well described in the original paper [4] and will not be repeated here. The tuned values for the primary GA are listed in Table II.

TABLE II
THE TUNED PRIMARY GENETIC ALGORITHM PROPERTIES.

Replacement		50%	
Crossover Rate		80%	
Mutate-External		1/2	
Mutate-Reverse		1/2	
Robots	Tasks	Population	Iterations
2	5	100	300
	15	100	1000
	30	100	3000
	50	100	10000
3	5	140	1000
	15	140	3000
	30	140	10000
	50	140	20000
4	5	140	1000
	15	140	6000
	30	140	13000
	50	140	30000

C. Secondary Genetic Algorithm

In some cases, the primary GA is unable to find solutions with full connectivity. This is discussed in detail in Section VI. To remedy this, we iteratively add random waypoints, and apply a single-objective GA for each added waypoint. For example, consider the solutions in Figure 3. The solution with the most connectivity (labelled C) has non-zero MaxNetGap score, indicating that there are times that the robots are not connected to one another.

As will be discussed in Section VI, we know that as the number of tasks in the problem increases, the harder it is for the primary GA to find a solution that maintains connectivity. This implies that as we add waypoints, finding a solution that maintains connectivity becomes more difficult. As such, we do not solve the problem from random initialisation for each added waypoint. Instead, we tune the secondary GA to incrementally improve solutions. The population of the secondary GA is seeded from the solution with the highest connectivity, and the waypoint is inserted into a random index for a random robot. Mutation operations are performed to improve the solutions without searching for brand new configurations, and no crossover is performed. We use the mutations: external swap, where two robots swap tasks; reverse, where a sub-path of a robot is reversed; internal swap, where two tasks in a path are swapped; and transfer, where a task is gifted from one robot to another. The tuned parameters for the secondary GA are listed in Table III.

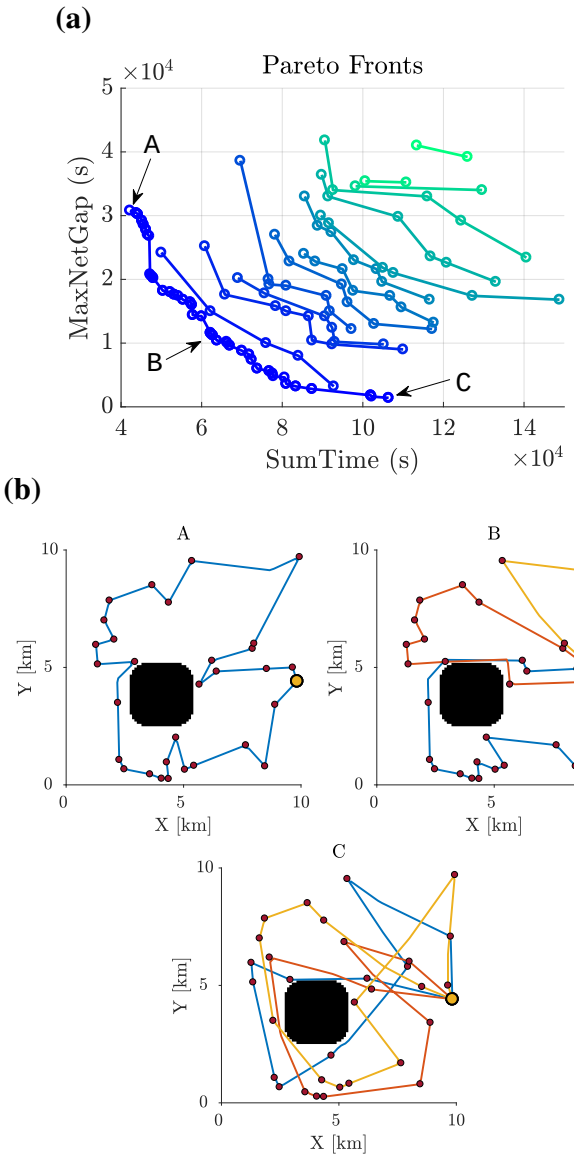


Fig. 3. An example solution set for a problem with 3 robots and 30 tasks. a) Each point in a line (Pareto front) represents a particular solution. b) robots (large circle) must visit locations (smaller circles). They cannot enter or communicate over the mountain (black). (A) The most energy efficient solution found. (B) A balance between energy efficiency and connectivity. (C) The solution where robots stay most connected so that they can respond to events.

TABLE III
THE TUNED SECONDARY GENETIC ALGORITHM PROPERTIES.

Population	100
Replacement	50%
Crossover Rate	0%
Mutate-External	1/4
Mutate-Reverse	1/4
Mutate-Internal	1/4
Mutate-Transfer	1/4
Waypoints Searched	50
Iterations	PrimaryGA Iterations / 50

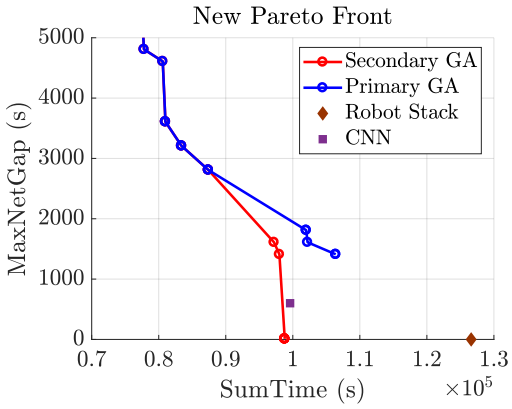


Fig. 4. The end of the first Pareto Front. Each point represents a solution, which has a cost for two objectives on the X and Y axis. The Primary GA was unable to find a solution that minimised the MaxNetGap objective. The Secondary GA found new solutions that filled this gap. It is more energy efficient (SumTime) than the Connected Nearest Neighbour (CNN) heuristic, and moving robots together as a stack.

D. Summary of Technique

The summary of steps for the Dual-GA are as follows:

1) Generate paths between each vertex pair (robot or task location), avoiding obstacles.
2) Seed the primary population with heuristics and randomly generated solutions.
3) (NSGA-II) Rank the primary population by Pareto Front, with crowding distance as the tie-breaker. Select and duplicate half of the primary population by weighted sampling. Crossover and mutate these solutions to produce children. Replace the weakest half of the primary population with the children. Repeat for the desired number of iterations.
4) Select the solution, s, with the lowest MaxNetGap score. Repeat for w waypoints:
   a) Add a waypoint within the map boundaries.
   b) Duplicate s, allocate the waypoint to a random robot, and use this solution as a seed for the secondary population. Repeat this until the secondary population is full.
   c) Rank the secondary population by MaxNetGap score. Select and duplicate half of the secondary population by weighted sampling. Mutate these solutions to produce children. Replace the weakest half of the secondary population with the children. Repeat for the desired number of iterations.
   d) Combine the primary and secondary populations. Rank the combined population by Pareto Front, removing all solutions that are not in the first front.
5) Repeat Step 4 until a solution where MaxNetGap equals zero is found, or until a time limit is reached.

A graphical example of step 1 can be seen in Figure 1, step 3 can be seen in Figure 3, and the final result can be seen in Figure 4. An example solution that prioritises connectivity is shown in Figure 5.

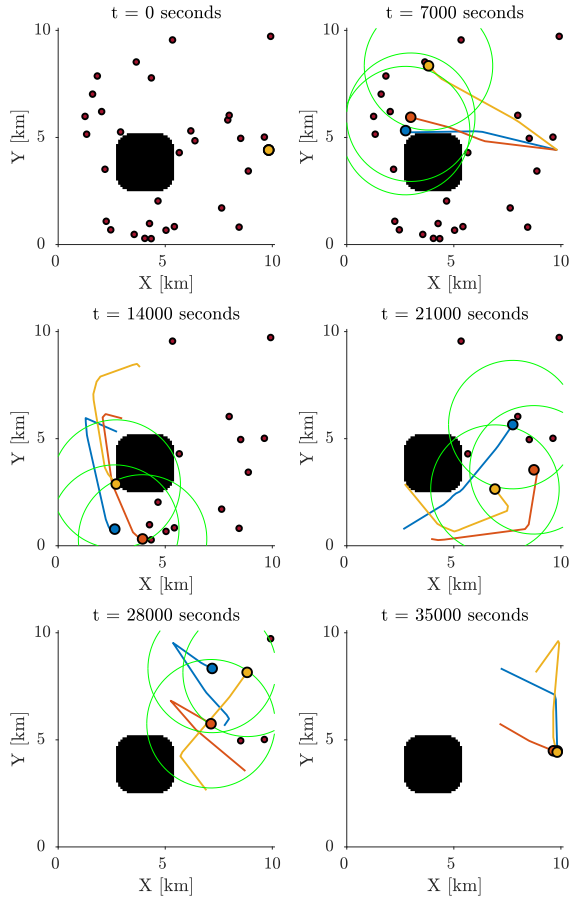


Fig. 5. Snapshots of robots (large filled circles) completing tasks (small filled circles) while maintaining connection range with one another (large thin circles). They cannot move or communicate through the mountain (black area).

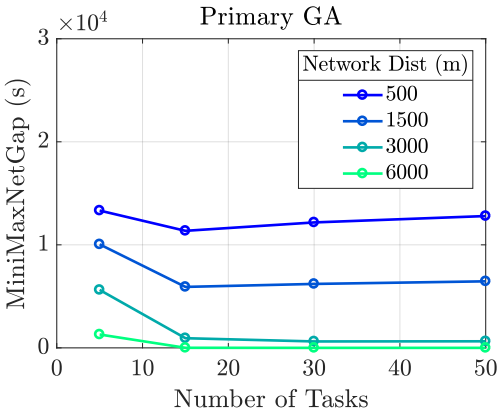


Fig. 6. A plot of the solutions with the smallest MaxNetGap objective score, from the Primary Genetic Algorithm (GA). The performance scales with networking distance, as it is simpler for robots to maintain connectivity when their communication distance is large.

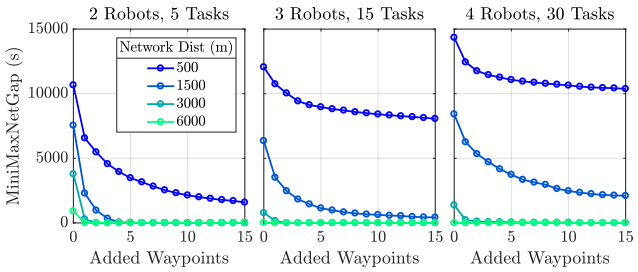


Fig. 7. The MiniMaxNetGap improvement from each added waypoint in the Secondary Genetic Algorithm (GA). Overall performance is dependent on networking distance between robots, as it is simpler for robots to maintain connectivity when their communication distance is large.

V. Tests

We alter the number of robots (2 to 5), the number of tasks (5, 15, 30, 50), and the connection distance (500, 1500, 3000, 6000, 10000 metres). Each scenario is repeated 50 times to produce an average. This results in 4800 tests, performed with the University of Adelaide's high performance computing cluster, Phoenix.

We compare the results to a Connected Nearest-Neighbour approach [13], which alternates between phases of task allocation and phases of interim points to maintain connectivity. We also compare the results to a trivial technique to maintain connectivity, where all robots visit all tasks by moving together i.e. stack the robots. Robot stacking is calculated by solving the mTSP with a single robot, then having all robots use that path. For example, consider the most energy efficient solution in Figure 3 (labelled A). All robots could follow this path together so that they remain connected.

VI. Results

A. MiniMaxNetGap

The primary GA does not always find solutions where the robots are connected all of the time, i.e. a MaxNetGap score of 0 seconds. For example, in Figure 3, the solution with the most connectivity (point C) does not lie on the x-axis. Looking at Figure 6, we can see that the MaxNetGap score is heavily impacted by the networking distance and the number of tasks. If the network distance is small (relative to the 10km area), robots must remain closer together. This means there are fewer solutions that have full connectivity. The primary GA has trouble with this, resulting in large differences in solution quality. The number of tasks also affects the primary GA MaxNetGap score, but to a lesser extent. We can see from Figure 6 that as the number of tasks increases, it becomes more difficult for the primary GA to find good solutions. However, a low number of tasks limits the number of ways robots can move.

Adding waypoints, as discussed in Section IV, shows an improvement in MaxNetGap score for each task added. In Figure 7, we can see that the improvement is largely dependent on the network distance. We can also see that each new waypoint provides diminishing returns.

Looking at Figure 4, the secondary GA has provided solutions with a greater MaxNetGap score. It also shows two other techniques, a Connected Nearest-Neighbour approach (CNN), and a stacking robot approach. The CNN does not produce a solution with a MaxNetGap score of 0 in this

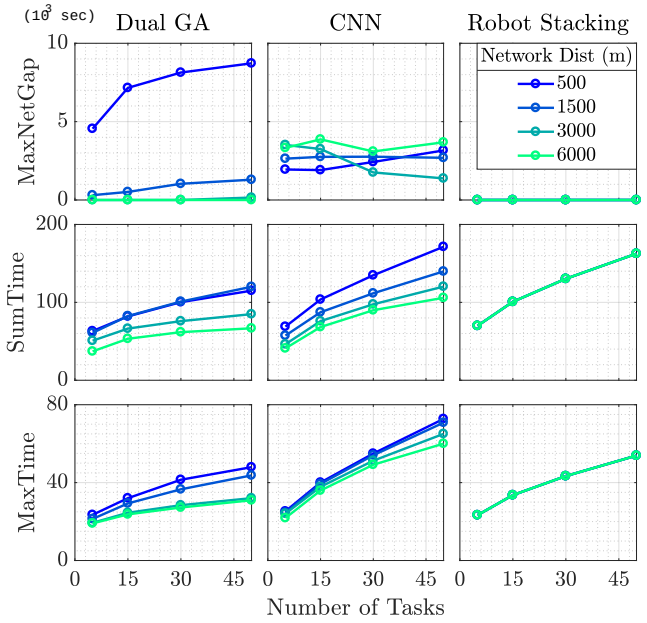


Fig. 8. Relative performances of the three techniques: Dual GA (ours), Connected Nearest Neighbour (CNN), and robot stacking. Three objectives are compared, the longest time between connection (MaxNetGap), the energy usage (SumTime), and the time taken for robots to return home (MaxTime).

case. It assumes that when robots start connected and end connected, the path they follow will keep them connected. With non-straight paths, this is not the case. The current problem has obstacles, resulting in non-straight paths between certain points. Stacking robots, however, results in 100% connectivity because the robots are moving with one another, but is less energy efficient than the other methods.

Figure 8 illustrates a graphical comparison between CNN, Dual GA, and robot stacking methods. The technique that provides the most connectivity is robot stacking, because all robots move together to complete every task. Dual GA provides more connectivity than CNN, with the exception of low connection distances, where Dual GA struggles to find solutions that maintain connectivity. For the energy efficiency objective, MiniSumTime, Dual GA uses the least energy, with the exception of very low number of tasks. Robot stacking uses the most energy. As for completion time, MiniMaxTime, robot stacking is slightly faster than CNN, but Dual GA is substantially faster than the others.

VII. Conclusion

The proposed Dual-GA is capable of providing a range of solutions with different balances between energy efficiency and robot connectivity. The primary GA can struggle to find solutions with 100% connectivity, so a secondary GA provides improvements through random placement of waypoints. Together, they provide a wide range of solutions for a decision-maker to consider for the Multiple Travelling Salesman Problem with network connectivity.

VIII. Acknowledgments

This research was supported by the Phoenix High Performance Computing service at the University of Adelaide, an Australian Government Research Training Program Scholarship, and by the Commonwealth of Australia through a Defence Science Partnerships agreement.

References

[1] Jairo R Montoya-Torres, Julin Lpez Franco, Santiago Nieto Isaza, Heriberto Felizzola Jimnez, and Nilson Herazo-Padilla. A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering*, 79:115–129, 2015.

[2] Kaarthik Sundar and Sivakumar Rathinam. Algorithms for heterogeneous, multiple depot, multiple unmanned vehicle path planning problems. *Journal of Intelligent & Robotic Systems*, pages 1–14, 2016.

[3] Michail G Lagoudakis, Evangelos Markakis, David Kempe, Pinar Keskinocak, Anton J Kleywegt, Sven Koenig, Craig A Tovey, Adam Meyerson, and Sonal Jain. Auction-based multi-robot routing. In *Robotics: Science and Systems*, volume 5, pages 343–350. Rome, Italy, 2015.

[4] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[5] Yuan Yan and Yasamin Mostofi. Robotic router formation in realistic communication environments. *IEEE Transactions on Robotics*, 28(4):810–827, 2012.

[6] Yuanteng Pei, Matt W Mutka, and Ning Xi. Connectivity and bandwidth-aware realtime exploration in mobile robot networks. *Wireless Communications and Mobile Computing*, 13(9):847–863, 2013.

[7] James Stephan, Jonathan Fink, Vijay Kumar, and Alejandro Ribeiro. Concurrent control of mobility and communication in multirobot systems. *IEEE Transactions on Robotics*, 33(5):1248–1254, 2017.

[8] Yiannis Kantaros and Michael M Zavlanos. Global planning for multi-robot communication networks in complex environments. *IEEE Transactions on Robotics*, 32(5):1045–1061, 2016.

[9] Yiannis Kantaros, Meng Guo, and Michael M Zavlanos. Temporal task planning and intermittent communication control of mobile robot networks. *arXiv preprint arXiv:1706.00765*, 2017.

[10] Jacopo Banfi, Alberto Quattrini Li, Nicola Basilico, Ioannis Rekleitis, and Francesco Amigoni. Asynchronous multirobot exploration under recurrent connectivity constraints. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 5491–5498. IEEE, 2016.

[11] Victor Spirin, Stephen Cameron, and Julian De Hoog. Time preference for information in multi-agent exploration with limited communication. In *Conference Towards Autonomous Robotic Systems*, pages 34–45. Springer, 2013.

[12] Jacopo Banfi, Alberto Quattrini Li, Nicola Basilico, and Francesco Amigoni. Communication-constrained multirobot exploration: Short taxonomy and comparative results. In *Proceedings of the IROS workshop on on-line decision-making in multi-robot coordination (DEMUR2015)*, pages 1–8, 2015.

[13] Yun Wang and Cheng Hu. Moving as a whole: multirobot traveling problem constrained by connectivity. *Turkish Journal of Electrical Engineering & Computer Sciences*, 23(3):769–788, 2015.

[14] Satyanarayana G Manyam, Sivakumar Rathinam, Swaroop Darbha, David Casbeer, Yongcan Cao, and Phil Chandler. GPS denied UAV routing with communication constraints. *Journal of Intelligent & Robotic Systems*, 84(1-4):691–703, 2016.

[15] Arthur E Carter and Cliff T Ragsdale. A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European Journal of Operational Research*, 175(1):246–257, 2006.

[16] Nick Sullivan, Steven Grainger, and Ben Cazzolato. Algorithms for multi-robot routing with adaptive heterogeneity. *Submitted to Journal of Heuristics*, 2018.

[17] Shuai Yuan, Bradley Skinner, Shoudong Huang, and Dikai Liu. A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. *European Journal of Operational Research*, 228(1):72–82, 2013.