Department of Computer Science

Submitted in part fulfilment for the degree of BEng

Exploring Time-Distance Tradeoffs in Multi-Objective Task Allocation for Multi-Robot Systems using Genetic Algorithms

Ayman Omiyo Zahir

01 May 2024, v4.0

Supervisor: Alan Millard

Acknowledgements

I want to thank my supervisor, Alan Millard, for his excellent guidance throughout this project. His patience and feedback were essential in helping me navigate both the challenges of my research and my personal development.

I also wish to thank my family for their love and support. My parents, Ishtiaque and Nasreen, and my brothers, Rayyan and Yasin, deserve special mention for always motivating me to challenge myself and be ambitious.

I thank my uncle Zakir and his family for their hospitality while I studied abroad.

I thank my friends, old and new - who believed in me when I never did and with whom I have shared many sleepless nights trying to make something of ourselves.

Statement of Ethics

This project has been conducted with no legal, social, ethical or professional implications in accordance with the University of York Computer Science Department's ethical standards. All the relevant sources have been cited, along with all the open-source software used. No personal data was used in this report.

Table of Contents

1 Executive Summary..... 1
2 Introduction..... 3
3 Literature Review..... 4
3.1 Multi Robot Task Allocation (MRTA) Overview..... 4
3.1.1 What is MRTA?..... 4
3.1.2 Motivation for research in MRTA..... 5
3.1.3 Applications of MRTA..... 6
3.2 General Topology and Approaches to MRTA..... 7
3.2.1 MRTA Topologies: Organisation and Assignment..... 7
3.2.2 Market-based Approaches..... 8
3.2.3 Behaviour-based Approaches..... 8
3.2.4 Optimization-based Approaches..... 8
3.3 Evolutionary Algorithms..... 10
3.3.1 Genetic Algorithms in MRTA..... 10
3.3.2 Multi-Objective Optimisation: Pareto Fronts and NSGA-II..... 12
3.6 Project Aims..... 14
4 Methods and Implementation..... 15
4.1 Problem Statement..... 15
4.2 Problem Environment Setup and Visualisation..... 16
4.3 Evolutionary Algorithm Framework..... 17
4.4 Design and implementation of Genetic Algorithm..... 18
4.4.1 Genome Representation..... 18
4.4.2 Fitness Functions: F1, F2 and F3..... 19
4.4.3 Crossover Operator..... 20
4.4.4 Mutation Operators..... 20
4.4.5 NSGA-II Implementation..... 20
4.4.5 Hyperparameters and setup of Genetic Algorithm..... 21
5 Results and evaluations..... 22
5.1 Evaluating F1 and F2: Distance versus time..... 22
5.2 Evaluating F3: Balancing time and distance..... 25
5.3 Optimising F3: Crossover and mutation rates..... 27
5.4 Exploring the pareto front: F3 vs NSGA - II..... 29
6 Conclusions..... 31
Appendices..... 33
Bibliography..... 39

Table of Figures

Fig 1 MRTA problem representation [2] depicting a mix of aerial/ground-based robots, allocating to a mix of visual inspection, heat sensing and maintenance tasks 4
Fig 2 Classification of 8 MRTA problems [2] 5
Fig 3 MRTA applied for mTSP drone delivery to customers, adapted from [14] 6
Fig 4 Central vs Decentral vs Hierarchical organisation, adapted from [2] 7
Fig 5 Local vs Global optima [34] 9
Fig 6 Evolutionary algorithm cycle [38] (left) and genome structuring (right), 10
Fig 7 pareto fronts with pareto optimal front in red 12
Fig 8 NSGA-II with non-dominated and crowding distance sorting [51] 13
Fig 9 Problem definition and setup of 21 tasks allocated across 4 robots 15
Fig 10 Design and encoding of genome into task and robot halves 18
Fig 11 Calculation of distance travelled (left) and time taken (right) per robot 19
Fig 12 Fitness over generation graphs for F1: distance (left) and F2: time (right) 22
Fig 13 Best distance solution (left) and typical solutions (right) for F1 23
Fig 14 Best time solution (left) and typical solutions (right) for F224

Table of Tables

Table 1 The aims of the project 14
Table 2: Hyperparameters of F1 F2 and F3 21
Table 3 Evaluation metrics for fitness function at (C1=0.01, M1=0.001, M2=0.001) 23
Table 4: ANOVA and Kruskal Wallis tests for F1, F2 and F3 differences 25

1 Executive Summary

This project aims to develop a genetic algorithm (GA) to solve a multi-objective Multi-Robot Task Allocation (MRTA) problem, focusing on optimising fuel consumption and time taken for factory plant inspections. By identifying optimal parameters for the GA and comparing our solution with the NSGA-II algorithm, we aim to explore and visualise trade-offs between these objectives.

The motivation for this work stems from the increasing complexity and demand for efficiency in modern warehouses and industrial environments. Leading companies like Amazon and Ocado have successfully deployed fleets of robots to optimise operations, underscoring the potential of multi-robot systems (MRS) to enhance efficiency, reliability, and scalability. However, deploying MRS in real-world scenarios presents the challenge of MRTA, which involves balancing conflicting objectives such as fuel consumption and time. Addressing these trade-offs is crucial for creating efficient, sustainable robotic systems capable of operating in dynamic environments.

To address the MRTA problem, we employed genetic algorithms, inspired by natural selection, which are well-suited for solving complex, multi-objective optimization problems. The project involved designing a GA with fitness functions to minimise fuel consumption (F1) and time taken (F2). We developed a weighted fitness function (F3) to balance both objectives and implemented crossover and mutation operators to evolve diverse solutions. We then compared our weighted GA solution with the NSGA-II algorithm to explore and visualise the pareto front, identifying optimal solutions and trade-offs.

The weighted fitness function (F3) effectively balanced the trade-offs between minimising distance and time. It provided better overall performance compared to single-objective functions (F1 and F2) by offering solutions that were both fuel-efficient and time-effective. Using NSGA-II, we visualised the pareto front, identifying a set of optimal solutions that highlighted the trade-offs between objectives. This allowed us to understand the relationship between distance and time, providing insights into how slight increases in travel distance could significantly reduce inspection time. Our findings demonstrate the importance of multi-objective optimization for complex, real-world tasks, such as factory plant inspections. The NSGA-II solutions allow for subjective selection based on context, providing flexibility for real-world applications where priorities can change based on immediate needs.

The genetic algorithm (GA) designed for this project achieved a balanced optimization, with the weighted fitness function (F3) providing a mean distance of 69 metres and a mean time of 48 seconds. These results were more efficient than those from F1 and F2 alone. Hyperparameter tuning identified optimal rates for crossover (0.05) and mutations (0.1 for tasks, 0.05 for robots), enhancing the GA's performance. Comparing the GA with NSGA-II, we found that NSGA-II offered a diverse set of pareto optimal solutions, highlighting efficient trade-offs between distance and time. These solutions were more robust and diverse compared to those obtained with F3 alone.

Our research contributes to the field of MRTA and multi-objective optimization by demonstrating the effectiveness of genetic algorithms in solving complex MRTA problems. The insights gained from this study can be applied to various MRTA scenarios, such as search and rescue, warehouse logistics, and environmental monitoring. While all project aims were successfully achieved, several limitations and areas for future work were identified.

Due to time constraints, we used a fixed set of parameter values instead of exploring adaptive parameter control. Future research could explore adaptive parameter control to enhance performance. Testing the algorithm with varying numbers of tasks and robots would help assess its scalability and generalizability. Extending the research to dynamic MRTA scenarios, where tasks and environmental conditions change in real-time, would provide a more comprehensive understanding of the algorithm's applicability. Implementing a more advanced version of NSGA-II with fine-grained parameter tuning could produce a more diverse and efficient pareto front.

This research has successfully demonstrated the potential of genetic algorithms and NSGA-II in optimising multi-robot task allocation, providing a robust framework for future advancements in the field. By balancing distance and time, we provided a robust framework for optimising task allocation. Our findings fortify the existing but sparse body of research on balancing distance and time and generate new insights by examining why these trade-offs occur. This initial set of results regarding the pareto front is consistent with work done in recent MRTA research. This research contributes to the field of MRTA and multi-objective optimization, highlighting the potential of these methods in advancing the efficiency and effectiveness of multi-robot systems. Future work should address the identified limitations and further explore the dynamic capabilities of MRTA solutions.

2 Introduction

In recent years, the advent of Industry 4.0 and the surge in robotics technology have revolutionised various sectors, particularly in warehouse automation and logistics. Leading companies like Amazon and Ocado have deployed fleets of robots to optimise operations, highlighting the practical potential of multi-robot systems (MRS). These systems have shown significant advancements in efficiency, reliability, and scalability. However, effectively deploying MRS in real-world scenarios involves addressing the complex challenge of Multi-Robot Task Allocation (MRTA).

The complexity of real-world MRTA problems lies in balancing multiple conflicting objectives. For instance, reducing fuel consumption might increase the time required to complete tasks, while minimising time can lead to higher energy usage. Addressing these trade-offs is essential for creating efficient, sustainable robotic systems capable of operating in dynamic environments. Moreover, it means that these robotic fleets could become more sustainable.

This project aims to address the complexities of MRTA by developing a genetic algorithm (GA) that optimises fuel consumption and time in factory plant inspections. Very few works have qualitatively explored as to why this occurs. By identifying optimal GA parameters and comparing our solution with the NSGA-II algorithm, we will explore and visualise trade-offs between these objectives in order to generate insights.

Genetic algorithms, inspired by natural selection, are well-suited for solving complex, multi-objective optimization problems. They evolve diverse solutions, thoroughly exploring the search space. Our project leverages GAs and NSGA-II to gain insights into optimising MRTA scenarios, ultimately enhancing operational efficiency and reducing costs. In essence, our research addresses the multi-objective nature of real-world MRTA problems, aiming to develop robust, scalable, and efficient task allocation strategies for modern robotic systems.

By conducting this research, we aim to advance the field of MRTA by developing robust and efficient algorithms applicable to various real-world problems, ultimately enhancing the performance and sustainability of multi-robot systems. This research not only contributes to the academic understanding of MRTA but also has practical implications for industries aiming to implement advanced robotics solutions in their operations.

3 Literature Review

3.1 Multi Robot Task Allocation (MRTA) Overview

The following subsections will give context to the field of multi-robot task allocation, outlining its definition, motivation and applications.

3.1.1 What is MRTA?

In recent years, robotic systems development has moved from deploying specialised single-robot systems to multi-robot systems (MRS) [1]. There are numerous advantages to MRS over single robot systems. These systems use multiple robots to solve problems more effectively such as moving large or distributed objects that may be difficult or impossible for a single robot [1]. MRS also improves performance, as multiple robots working in parallel can significantly reduce task completion time. It provides robustness and reliability, ensuring tasks are completed even if one robot fails, unlike a single robot system where a failure can halt the entire operation. Furthermore, MRS design tends to be simpler for complex tasks, encouraging the use of simpler, cheaper robots [1].

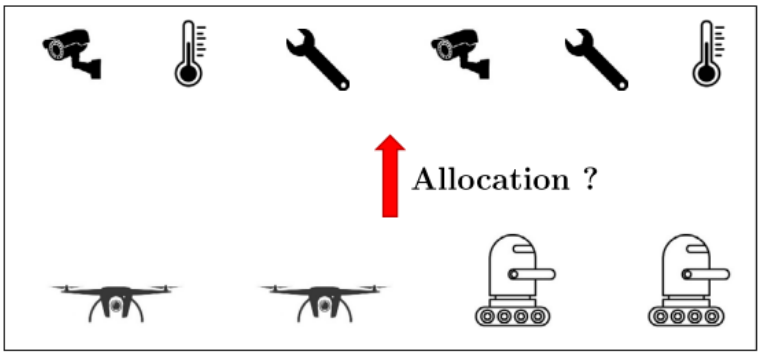


Fig 1 MRTA problem representation [2] depicting a mix of aerial/ground-based robots, allocating to a mix of visual inspection, heat sensing and maintenance tasks

However, to effectively build and deploy MRS, we need to first answer, "Which robot should carry out which task and in what order?" This introduces the Multi-Robot Task Allocation (MRTA) problem, a challenging and popular research area in MRS [1]. As seen in Figure 1, the MRTA problem is a combinatorial optimization problem [2]. It involves optimally assigning a set of tasks to a group of robots to optimise an objective function - such as system performance or cost (e.g., total time taken) subject to constraints [2]. Due to its combinatorial nature, MR-TA has multiple variants. Academically, MRTA problems and their complexities are classified based on taxonomies that define the relationship between robot capabilities, task requirements, and time [3].

Robot capabilities can be single-task (ST) versus multi-task (MT) robots: Robots can execute only one task at a time or execute multiple tasks simultaneously. Similarly, task requirements can be single-robot (SR) versus multi-robot (MR) tasks: Tasks require exactly one robot to complete or multiple robots. Time is considered either instantaneous (IA) or time-extended (TA) assignment: independent tasks are instantly assigned to robots without future planning versus a sequence of tasks being assigned to robots as time extends as a factor [3]. Other factors, such as whether robots are homogeneous and task dependencies, further complicate the problem space [4].

As per Figure 2, MRTA can be characterised into 8 problems according to these factors with SR-ST-IA problems being the least complex. However, in more niche MRS scenarios, MRTA becomes more complex and falls into the realm of NP-hard problems, especially MR and MT problems with heterogeneous robots and multiple constraints [3]. Understanding these 8 problem characteristics is crucial for effectively modelling and solving any MRTA problem.

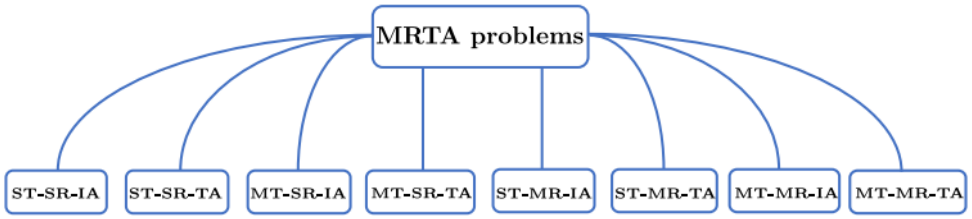


Fig 2 Classification of 8 MRTA problems [2]

3.1.2 Motivation for research in MRTA

The development of MRTA is vital to fully utilise MRS in real-world applications that are optimal, scalable and consistent. Through optimization, MRTA allows the creation of systems tailored to specific objectives, making them more commercially viable by maximising task completion or minimising costs, such as travel distance or time. Importantly, the objective function can also be adjusted to minimise energy usage, promoting sustainability and reducing carbon footprints.

This is particularly important given the current challenges of climate change and the negative environmental impact of industrial robotics [5]. Equally, agricultural robotics help combat climate change on the opposite end [6]. These factors continue to drive research in MRTA, particularly from an optimization standpoint towards more sustainable robotic systems.

3.1.3 Applications of MRTA

MRTA problems have been studied since the 90s [7], interconnecting disciplines like computer science, mathematics, operations research, robotics and artificial intelligence. Effective application of MRTA relies not just on its classification, but also on how it's modelled across the disciplines. For instance, consider the multiple travelling salesman problem (mTSP) which allocates and optimises routes for 'm' salesmen to visit 'n' cities, starting and ending in the same city [8]. By replacing salesmen and cities with robots and tasks, it becomes an ST-SR-TA problem. With proper modelling, MRTA has been recently used to solve real-world problems in areas such as search and rescue [9], surveillance [10], environmental monitoring [11] or warehouse logistics [12], [13]. For example, it has been modelled as mTSP to efficiently allocate robots for loading, unloading, sorting and assembly in automated warehouse scenarios as well as using teams of drones for goods delivery [14], depicted in Figure 3. MRTA is also used to ensure humanitarian safety in risky environments, such as inspecting crops, power grids and even disaster response missions [6], [15], [16]. However, these instances pose challenges as the systems often operate under imperfect conditions with inaccurate environmental information. This includes robot failures, unexpected events, dynamic obstacles and tasks, and more, making task allocation difficult. Hence, recently, there's been a significant increase in efforts for dynamic MRTA research [16]–[18]. There is still a lack of work in dynamic MRTA due to its complexity. Instead, dynamic MRTA works have been oriented towards multi-objective optimisation [19]. This focuses on allocating tasks to tackle real world complexities where it's crucial to optimise multiple objectives simultaneously by understanding their relationships - for example, ensuring safety, reducing energy and cost while still returning profits. Interestingly, most previous work has focused on SR-ST problems and only recently are we seeing increasing research on MR and MT problems [2].

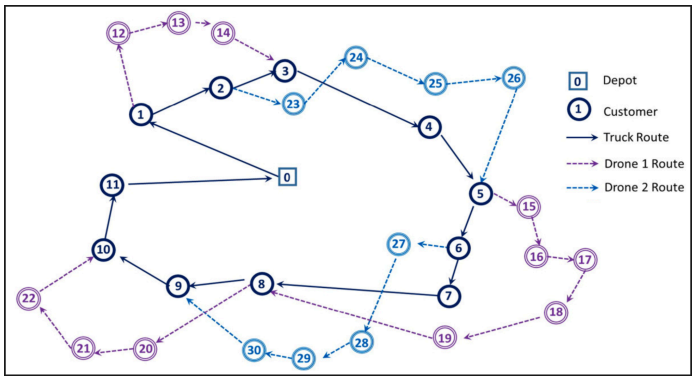


Fig 3 MRTA applied for mTSP drone delivery to customers, adapted from [14]

3.2 General Topology and Approaches to MRTA

The following subsections outline MRTA topologies and the main approaches to solving MRTA, focusing on optimisation methods.

3.2.1 MRTA Topologies: Organisation and Assignment

MRTA can be categorised into various topologies. Organisationally, we have central, decentral and hybrid methods [20] as illustrated in Figure 4. In centralised methods, robot coordination is intentional and controlled beforehand through a central agent based on global information. This facilitates efficiency by reducing duplication of effort and resources, thus saving time and costs [21]. It is the more popular method due to easier implementation [1]. However, it lacks robustness as the entire system fails if the central system fails. They are less efficient as more robots and tasks are introduced thus restricting scalability [1]. This computational overhead makes it hard to implement for dynamic scenarios. Hence, it is ideal in smaller static scenarios where global information is easily available.

Conversely, decentral systems have no central agent. Instead, allocation is emergent as robots communicate and delegate actions amongst themselves based on local information during missions [21]. It is robust and ideal for dynamic scenarios - robots can make faster real time decisions and if a robot fails the others can still complete the task without a central point of failure [37]. Hence, it is less computational intensive to add robots and thus scalability is no longer an issue [1]. However, without global knowledge, it produces suboptimal solutions at best. These solutions are also emergent, making behaviour harder to predict.

In practice, hybrid hierarchical approaches are also used to leverage the best of both central and decentral approaches [2].

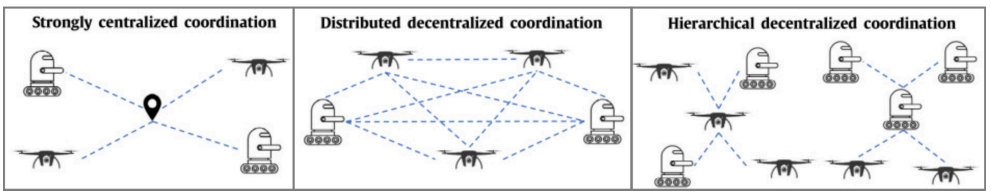


Fig 4 Central vs Decentral vs Hierarchical organisation, adapted from [2]

Task assignment is approached in three ways: Offline, Iterated, and Online [3]. Offline assignment handles the allocation of known tasks within a known environment, with allocation done before the mission starts. This is ideal for centralised methods when the environment and tasks are static, such as in warehouse automation. However, it is unsuitable in dynamic scenarios where the environment changes during the mission.

Conversely, iterated assignment deals with allocation of incoming tasks in dynamic scenarios. When new tasks are added, robots are released from their previous tasks, regardless of whether they've been completed, and instead assigned to the remaining tasks.

Online assignment also allocates incoming tasks, but, instead of cancelling the previously assigned tasks, new tasks are received after the completion of previous ones. Both Iterated and online are ideal for decentral or real-time approaches. Next, we introduce the three main approaches.

3.2.2 Market-based Approaches

These approaches are inspired by concepts in economic theory such as auctions and voting. An auction assigns a set of goods to a set of bidders according to their bids and auction requirements, directly paralleling MRTA. Robots communicate and negotiate by bidding for tasks based on their capabilities, applying market theory to optimise objectives [22]. This exhibits desirable features such as efficient solutions despite local information and limited resources, robustness against uncertainty and scalability [23], making them popular for decentral auctioneer systems such as CNP [24] for negotiating cooperative tasks and Trader-Bots [25] for dynamic environments. However, they lack formalisation in development and negotiation protocols are hard to develop. They also produce poorer solutions and require more computation in comparison [26].

3.2.3 Behaviour-based Approaches

In behaviour based approaches, tasks are divided into behaviours which motivate robots to perform tasks according to predefined rules for prioritisation and behaviour combinations without inter-robot negotiation. ALLIANCE [27] and BLE [28] are both pioneering decentral algorithms. More recent work attempts MRTA by drawing behaviours from game theory and social choice theory [29], [30].

3.2.4 Optimization-based Approaches

Optimization based approaches have emerged as the most popular and varied method to solving MRTA related problems, outperforming other strategies in terms of performance and complexity [31], [32]. They are varied and help solve a range of MRTA problems. However, their biggest strength lies in the ability to explore global solutions instead of local in the search space due to the randomness of the algorithms used [33].

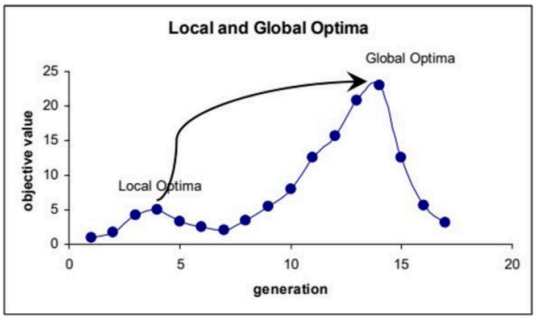


Fig 5 Local vs Global optima [34]

Figure 5 represents this behaviour. Optimization approaches fall into two categories: deterministic, which guarantees the repeatability of optimal solutions, and stochastic, which employs randomness to find diverse solutions for larger optimization problems. The deterministic Hungarian Algorithm (HA), introduced in 1955 [35], is widely used for MRTA problems. It has been refined over the years and applied to warehouse control and search and rescue operations. However, it struggles when tasks outnumber robots [2], a common scenario in most multi-robot systems. Another notable approach is linear programming models such as MILP. Recently, MILP was applied successfully to implement MRS for disaster responses, mapping survival kits to potential victims [16]. They have good solutions for the real world that outperform HA , but lack robustness and are computationally expensive, especially for large-scale problems [2]. Hence, stochastic methods were developed to provide a faster alternative to finding optimal solutions via randomness.

Trajectory-based methods like simulated annealing [36] rely on a single solution that randomly searches the space to find the global solution. However, they tend to get stuck at local optima [1]. In contrast, population-based methods solve problems by continually improving a population of candidate solutions through an iterative process [83]. It constitutes genetic algorithms (GA) from evolutionary computation [57] and particle swarm [37] or ant colony optimization techniques [38], [39] from swarm intelligence. These bio-inspired methods, which simulate nature, have emerged as a promising approach for MRTA.

In particular, genetic algorithms are the most popular optimization approach for MRTA [2], outperforming any colony [40] . Pivotal research was conducted in studies [11], [41] that used GA to solve allocation problems for environmental inspection of fixed locations in known environments. Since then, more advanced GA have been applied in various MRTA scenarios both centrally and decentrally, and in hybrid with other methods. While slower, GA converges to diverse optimal and near-optimal solutions and is thus ideal when optimising for multiple objectives closer to real world scenarios.

3.3 Evolutionary Algorithms

Evolutionary algorithms are inspired by Darwinian evolution and natural selection [42]. By mimicking the evolution of diverse species it provides an alternative approach to finding optimum solutions. Initially, a population of candidate solutions represented as genomes is randomly generated and evaluated against a fitness function and the best solutions are selected for the next generation. Mutations and crossovers are then applied to the genes of the selected population to add random variations, hoping to find new and improved solutions for the next generations. This process is repeated, mimicking evolution, until some termination condition is fulfilled, as depicted in Fig 6 (left) [38].

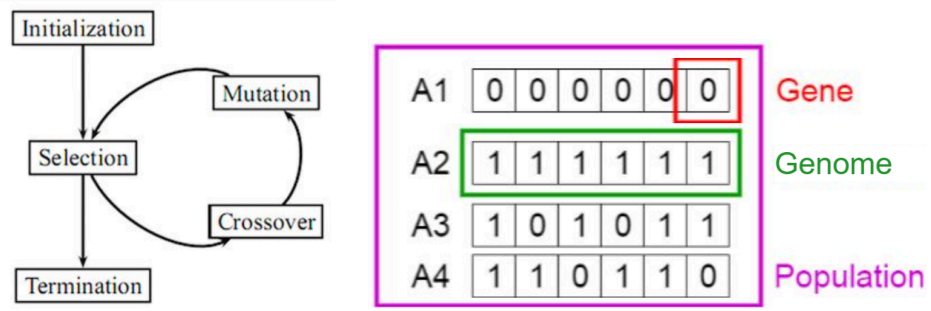


Fig 6 Evolutionary algorithm cycle [38] (left) and genome structuring (right),

Given the increasing complexity of MRTA problems, they provide a suitable alternative, popularly via genetic algorithms . GAs tackle the MRTA problem by representing it as a genome. A population of genome solutions is incrementally evolved by optimising a fitness function through specific mutators and crossovers tailored for allocation and scheduling issues which are introduced next.

3.3.1 Genetic Algorithms in MRTA

Genetic algorithms (GA) are popular evolutionary algorithms and they show superior performance for solving scheduling problems [43] like MRTA. To evolve for various problems, a variety of genetic operators have been developed. These include encoding schemes, crossover, mutation, selection and fitness functions.

The encoding scheme defines how to represent the problem as a population of genomes, consisting of individual genes. As seen in Fig 6 (right), Genomes are typically depicted as arrays, and the genes within them are encoded based on the given problem. The schemes include binary, hexadecimal, permutation, value and tree based encoding [43]. The ideal scheme should produce a genome that most closely resembles the problem. For instance, binary encoding represents genomes as bit arrays -

making them quick to mutate but difficult to represent problems. Permutation encoding, typically used in MRTA, presents genome arrays as a sequence of numbers - making it suitable for problems involving task ordering [43]. However, designing MRTA problems as genomes is complex and requires advanced genome representations for specific problems such as dividing the array or using multiple arrays and schemes to represent tasks, robots or constraints separately [11], [19].

Selection is crucial in the convergence of the evolution process towards good solutions. It determines which individuals are selected for reproductions, ensuring that the better individuals make up the latter generations. Two of the most common techniques include roulette wheel and tournament with the latter performing well in mTSP problems [44]. Increasing the size of the tournament applies more selection pressure to produce fitter results.

Crossover operators create new and diverse offspring genomes by combining elements from two or more parent genomes. These operators identify and carry over superior parts of the genomes, preserving them in subsequent generations. Notable crossovers include single-point, two-point, k-point, uniform, partially matched (PMX), ordered (OX), precedence preserving (PPX), shuffle, reduced surrogate (RCX), and cycle [43]. Operators such as PMX, OX, PPX, RCX, cycle, and edge crossover are typically used for MRTA related problems where order matters [43].

While OX crossover provides better exploration than others, it is less efficient in solving TSP problems [45]. Similarly, cycle and RCX tend to converge prematurely towards local optima instead of global. Overall, PMX is the most popular operator in MRTA due to superior performance and exploration capabilities [43]. Proposed in 1985 to solve the TSP, it's now one of the most used operators for ordering and MRTA problems [46].

Mutation operators help maintain the genetic diversity of populations across generations. They facilitate the search for global optima by applying random mutations to the genes. Typically, certain or random genes from the genome array are slightly modified. The type of mutation used depends on the encoding and the problem itself. To ensure good orderings of tasks are preserved, well-known operators like swap, shuffle, insert, scramble and inversion mutation are used in MRTA depending on the problem [43].

Significant challenges include fine-tuning parameters for mutation and crossover probabilities and exploring their interactions, as these greatly affect GA performance [47]. This is typically done by testing different values before the run. Without optimal crossover, we lose good genes, and without optimal mutation, we stagnate at local optima. A proper balance

between these operators is needed to ensure optimal solutions. Fitness functions are crucial as they determine how fit individuals are per generation. Defining and selecting efficient fitness functions is a significant challenge in GAs as we need to balance computational cost and suitability for our goals. For real-world problems, we also need functions that optimise multiple objectives simultaneously.

3.3.2 Multi-Objective Optimisation: Pareto Fronts and NSGA-II

Most recent GA-based MRTA solutions optimise for single objectives like distance, fuel, tasks, or time [2]. However, real-world problems often involve multiple, conflicting objectives where improving one worsens another. Thus, these objectives need to be optimised simultaneously through Multi-Objective Optimization (MOOP) [48]. The goal of MOOP is to obtain a set of non-dominated solutions, known as the pareto optimal set, where no objective can be improved without detrimentally affecting another. These solutions lie along the pareto front, where all points represent the best trade-offs between objectives [49]. Solutions on the front are equally good, representing the best of both worlds. Fig 7 illustrates the fronts for minimization problems where the red line represents solutions on the pareto front. The front's shape can be convex, concave, or irregular.

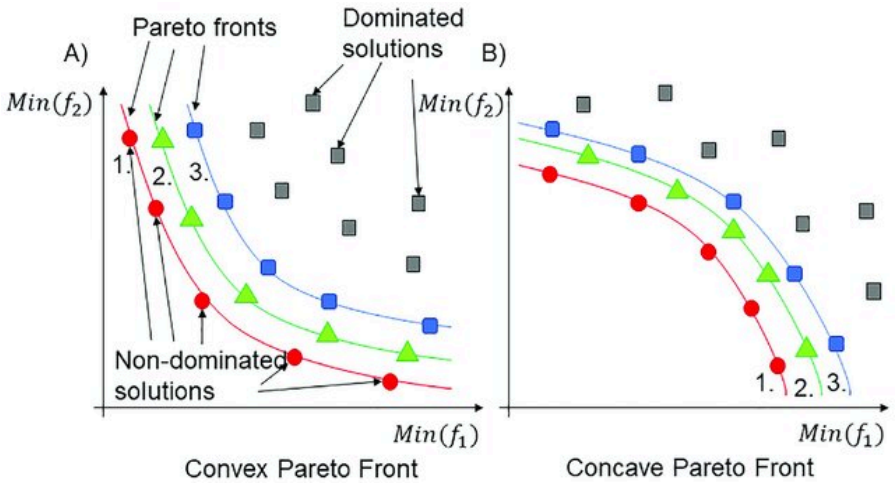


Fig 7 pareto fronts with pareto optimal front in red

In MOOP, we want to find solutions as close as possible to the pareto front that is diverse and spread along the front. Visualising the pareto front helps us understand the relationship between objectives and make informed decisions. For minimization problems, the pareto front should be as close to the bottom left of the graph as possible. Solutions below this point are infeasible, while those above represent worse outcomes. Smaller spreads could mean the multiobjective nature between objectives is small.

The simplest approach to obtaining multi-objective solutions is the weighted sum method, where predefined weights are applied to each objective [50]. These weights, which add up to one, define the priority of each objective. However, finding the ideal weights for a desired pareto-optimal solution is difficult via trial and error, making it a pseudo multi-objective approach. It also struggles when the front shape is irregular.

In contrast, the fast and elitist non-dominated sorting genetic algorithm, NSGA-II, is a popular method for true multi-objective optimization [51]. It combines GAs with non-dominated sorting and crowding distance sorting to find a pareto optimal set of solutions as per Figure 8. It initialises survivor selection by merging the original population Pt with offspring Qt, obtained through crossover or mutation. Elitism is then applied by ranking non-dominated solutions into ranks based on fitness. A solution A is non-dominated over B if A is at least as good in both objectives and better in at least one. The top rank F1 is the pareto optimal front. Crowding distance sorting is used to maintain diversity by selecting solutions that are least crowded. Although more complex, NSGA-II generates a diverse set of pareto optimal solutions and provides a better visualisation of the pareto front.

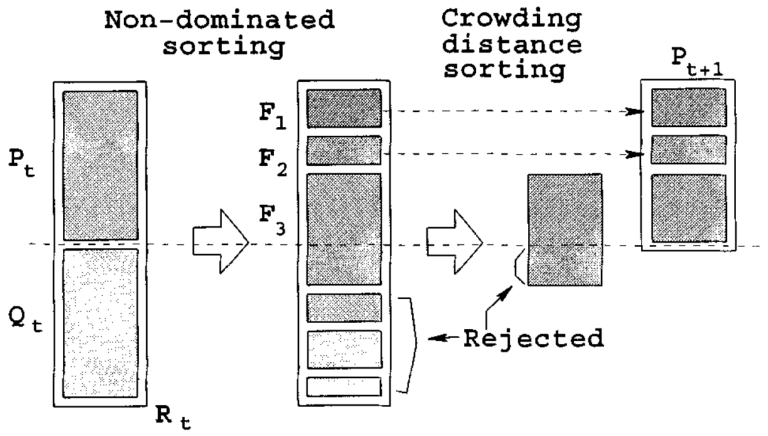


Fig 8 NSGA-II with non-dominated and crowding distance sorting [51]

NSGA-II and its variants have been applied to various problems, in allocation and scheduling [52], [53] and more recently in MRTA [54], [55]. However, most studies lack qualitative exploration of the multiobjective pareto relationship. The pivotal study [11] optimises distance and time separately but it overlooks their potentially antagonistic relationship.

Thus a multi-objective nature between distance and time could be explored. MOOP based MRTA studies typically focus on makespan versus time. While some studies [54] have been conducted they do not qualitatively explore the distance-time relationship. Uncovering the whys can provide insights that enable decision making between distance-time

optimisation. Hence, we aim to use genetic algorithms to explore the distance time relationship. They are a promising alternative to solve MRTA problems. Their distinction lies in evolving a diverse set of solutions that fully explore the search space, to find optimal solutions. This trait is ideal for multi-objective problems that reflect the complexities of the real-world.

However, there exists significant challenges in developing GAs for multi-objective multi-robot task allocation. This not only includes designing the genome and selecting operators but in the exploration and development of multi-objective fitness functions and fine-tuning the operators. In the context of MRTA, we aim to understand the pareto fronts to generate insights for real world use cases. Therefore, there is value in researching genetic algorithms for MRTA in the context of multi-objective problems that reflect the real world.

3.6 Project Aims

Hence, the aim of this project is to develop a genetic algorithm to solve a multiobjective MRTA problem in factory plant inspection. Specifically, we aim to develop simple initial functions for fuel consumption (modelled as distance) and time to explore their relationship. This informs the development of a multiobjective weighted fitness function that optimises both objectives. We aim to find the optimal mutation and crossover probabilities. Furthermore, we aim to evaluate the functions and solutions qualitatively and quantitatively and explore the multiobjective relationship between the two objectives. Finally, we will compare our weighted solution with an NSGA-II algorithm to understand the pareto front and explore pareto-optimal solutions, generating insights for the inspection task. We use DEAP as our evolutionary framework in a Google Collab Notebook and have integrated our GA with a custom made 2D visualiser to analyse our solutions and modify our problem space. To achieve our goals the following aims were decided for the project in Table 1.

ID	Project Aims
1	Design and implement a genetic algorithm and its operators using DEAP and Python that successfully minimises our objectives.
2	Implement a custom 2D visualiser to analyse our solutions using Python libraries
3	Develop and evaluate appropriate fitness functions F1, F2 and explore their relationship to develop a weighted function F3.
4	Evaluate the weighted function F3 and identify the optimal mutation and crossover parameters and evaluate how to balance them.
5	Compare our weighted solution with a true multi-objective NSGA-II algorithm and explore the pareto front and its solutions.

Table 1 The aims of the project

4 Methods and Implementation

This section defines the problem that the project aims to explore and outlines the methods used and their implementation.

4.1 Problem Statement

This project aims to develop a genetic algorithm to solve a static offline multi robot task allocation problem centrally. Specifically, an mTSP for finding out the optimal sequence for inspecting all the locations in an industrial plant by a given number of robots. Mathematically, a set of m robots $R = \{R1, R2, \ldots, Rm\}$ are assigned n inspection locations $T = \{T1, T2, \ldots Tn\}$. Solutions are evaluated by their total fuel consumed and total time taken. It is assumed that one task is assigned to a single robot at a time, ensuring that all tasks are executed exactly once, with all robots starting from their depots simultaneously and returning to their depots upon completion. Since our focus is on task allocation and not path planning, a simple algorithm is used to generate the shortest paths between tasks without collision detection. Figure 9 shows the setup with a possible solution with coloured paths for robots, depicted as circles, in 2D. The grey squares represent tasks.

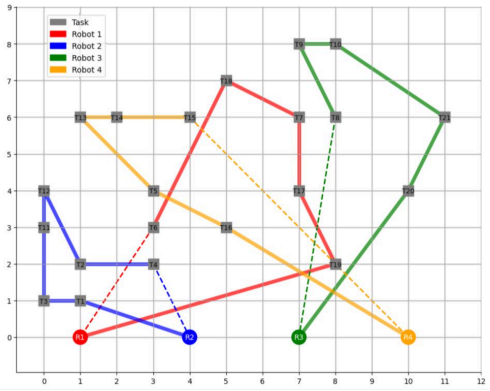


Fig 9 Problem definition and setup of 21 tasks allocated across 4 robots

This is a static SR-ST-TA problem with offline task assignment. GAs are popularly used for solving SR-ST-TA problems and most real world problems also fall in the SR-ST category. We decided to tackle static MRTA which is not solved completely and requires attention [56] . Dynamic MRTA is too novel and complex and thus, beyond the project scope.

Our main goal is to develop fitness functions to optimise both fuel and time, thus exploring their relationship for conflicts to deduce if it's worth developing a weighted multi-objective function. The second goal is identifying ideal hyperparameters for the mutation and crossover operators. Balance between the operators ensures quick convergence

towards good solutions. Finally, we develop a true multi-objective GA with NSGA II and compare against the weighted GA to see if it can provide equal or better solutions. To achieve this, we will need an evolutionary algorithm framework and a simulation or visualisation system for our problem.

4.2 Problem Environment Setup and Visualisation

Before starting, we need to be able to simulate our problem environment and visualise solutions. Available options were explored exhaustively to avoid going back if we choose the wrong system. Typically in multi-robot systems, a robotic simulator is used to accurately simulate and test the behaviour of the system in an environment that closely resembles the real world to reduce the reality gap. This typically includes a physics-based 3D environment, sensors, robots and even various weather and terrain conditions. Some even have pre-built environments specific to task allocation scenarios such as warehouses and factories. Popular simulators include Gazebo, V-REP and ARGoS [57].

These tools are typically used when testing the behaviour or control software for specific robot models. However, they require substantial computational resources and have a steep learning curve. Most of them, built using the Robotic Operating System (ROS), have significant overhead [58]. ARGoS stands out as a lightweight option, but lacks pre-built scenarios for task allocation. Considering their computational demand, these tools aren't suitable for our GA, particularly for task allocation where the focus is the outcome, not simulating robot behaviour. Simulating and evaluating each solution would exceed my device capabilities and lengthen the GAs execution time. MRTA exacerbates this as the search space can get exponentially complex - needing larger population size and more generations, thus taking longer to converge to good solutions.

Instead, for task allocation, we decided to use a 2D system to visualise our outcomes as solution paths. For the purpose of this experiment, an abstract 2D visual of the robot paths and the outcomes of our fitness function is sufficient as we are able to model our problem and constraints. . Our approach allows us to run our genetic algorithm at a reasonable speed.

The project required a 2D multiagent visualiser compatible with DEAP and Python. NetLogo [59] and MATLAB were strong candidates with interactive 2D simulations and visualisations, pre-built MRS scenarios and real-time data plots. However, they were incompatible with Python and DEAP. Likewise, Python based systems such as V-MAS [60], offered a plethora of pre-built multi-robot scenarios in a vectorised 2D physics engine. Yet, it was hindered by poor documentation, a lack of task allocation scenarios, and a focus on reinforcement learning. Similarly, Mesa [61] is web-based and well

documented but lacks clear examples of use in multi-robot scenarios. Moreover, its grid-based visualisation was too abstract to represent our environment.

Eventually, as most available solutions had limitations, we decided to code a simple 2D representation from scratch using Python and its matplotlib libraries in a Google Collab Notebook. This allowed us to quickly code a visualiser tailored to our needs while having flexibility to modify components easily without having to dig through documentation or limit ourselves to a specific tool.

The visualizer employs Matplotlib to create a 2D grid representation of the environment (12 x 9m), as shown in Fig 7. It displays 21 tasks as grey squares and 4 robots. Using the solution genome of tasks and robot allocations, it generates and visualises robot paths. Each robot is linked to its assigned tasks in sequence by calculating the shortest path, with a dashed line indicating the return path to the depot. Each robot's path is colour-coded for easy identification, facilitating quick and qualitative analysis of the final solutions. With an appropriate visualisation, we then decided on our evolutionary algorithm framework for evolution.

4.3 Evolutionary Algorithm Framework

Evolutionary algorithm frameworks provide us with the operators that allow us to perform the evolutionary processes as described in our literature review and Fig 6. Typically they provide built-in functions to facilitate population generation, selection, mutation, and crossover. Amongst these frameworks, Genetic Algorithm Library (GALib) is a high speed C++ alternative but it is outdated and struggles with complex optimization tasks. ECJ (Evolutionary Computation in Java) and MOEA (multi-objective evolutionary algorithms) are Java-based, but they failed to meet our needs. ECJ allows custom classes to represent individuals [120] but has limited documentation, and MOEA specialises in multiple-objective algorithms which limits applicability for single objective problems.

After considering options, we decided on DEAP [62] as it has features similar to the other frameworks for both multi and single objectives. Moreover, it's user-friendly, easy to pick up, and built on Python. As it is highly modular, we can mix and match different components to quickly create prototypes and test proof of concepts. Unlike PyMoo, PyGAD and GeneAI, DEAP benefits from extensive documentation and an active community. Any issues can be quickly addressed via Stack Overflow or looking at the source code comments. It is also widely recognised in academia and industry. DEAP has functionality for hyperparameter and operator adjustments. This allows efficient fine-tuning of our parameters.

Most importantly, DEAP provides a wide range of built-in tools specific to MRTA problems such as permutation representations, PMX and ordered crossover as well as NSGA-II selection. These are well documented which allows for quick modification if needed. Additionally, it has tools such as hall-of-fames and logbooks to keep track of the best solutions and data. All of this, plus familiarity with Python and DEAP made it the ideal option without having to learn or build from scratch. The next step was to visualise our environment and solutions.

4.4 Design and implementation of Genetic Algorithm

This subsection explains the design of our genetic algorithm and its operators, explaining their implementation and how they work.

4.4.1 Genome Representation

Our solution's genome or individual needs to accurately represent the problem by considering both task schedules and robot assignments. We accomplish this by splitting an array of integers into two equal parts. The first part includes n unique integers, to represent a random permutation of the n tasks. The second part contains n integers, representing the n assignments of m robots. Therefore, this half also considers the number of tasks assigned to each robot, which can vary from none to all n tasks.

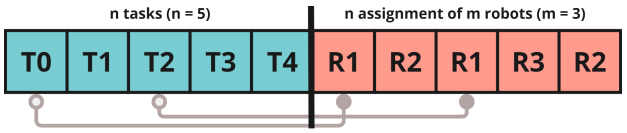


Fig 10 Design and encoding of genome into task and robot halves

Figure 10 depicts an individual. For example, R1 is assigned T0 first, then T2. The create_chromosome function uses Python's random library to create each half separately, which are then joined to form an individual. We used integers to identify tasks, starting from 0, since DEAP's operators like PMX are only compatible with integer lists starting from 0. 2D coordinates of each task and the time taken to complete them are stored in a Python hashmap task_dict as an integer list and integer, respectively. Robot coordinates are stored similarly in robot_dict. The task or robot number serves as the key, with their details stored inside a list. We used hashmaps as they are easy to implement and modify at any point. Functions can access them efficiently and globally by simply referring to the task or robot number. This method provided a more streamlined and intuitive approach for experimentation than defining classes for robots and tasks and manually defining coordinates for each instance.

This encoding ensures our GA effectively searches the large fitness landscape. Permutation crossover and mutation on the task half result in large jumps across the search space due to the spread-out nature of tasks. Meanwhile, mutating the robot half leads to frequent but localised searches, as changing one gene does not drastically affect others. We also considered alternative approaches, such as defining the robot half by the number of tasks assigned and linking contiguous task segments to each robot [11]. However, these methods cause more jumps across the search space and are mainly useful when there are minimum constraints on the number of tasks per robot. Since our study aimed to fully explore the distance-time relationship, a constrained method was not suitable.

4.4.2 Fitness Functions: F1, F2 and F3

The project aims to define functions that minimise total fuel consumed as F1 and total time taken as F2. F1 models fuel consumption as the sum of the euclidean distance travelled by all robots to complete their tasks and return to their depots. The evaluation function divides an individual into tasks and robots halves and inputs them into the F1 (updated_distance_function) function. For each robot, it retrieves the task and robot coordinates from the hashmaps and iterates through their task sequence to calculate the distances from current robot position to the task locations, updating the robot position after each task as per Figure 11 (left).

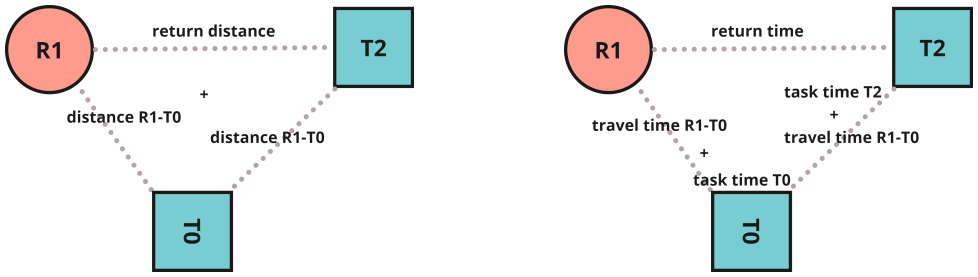


Fig 11 Calculation of distance travelled (left) and time taken (right) per robot

Similarly, F2 calculates the total time taken by robots to complete all tasks and return. It calculates the sum of travel times for each robot path similar to F1 but also adds the time taken for each task by accessing the task_dict hashmap. The fitness is defined as the maximum time taken among all the robots as it signifies the finish time for the last robot. To eliminate bias the robot speeds and time taken per task were constant at 1 m/sec and 5 secs.

Finally, we developed our weighted function $F3 = W1 \cdot F1 + W2 \cdot F2$. We apply weights $W1$ and $W2$ to $F1$ and $F2$ in order to balance fuel and time. The weights define how much to prioritise each goal. $F1$ and $F2$ were equally prioritised by setting their weights to 0.5.

4.4.3 Crossover Operator

Crossover operators ensure our generations carry over good genes from previous solutions. Designing them for permutation encodings in MRTA is challenging. The aim is to exchange genes without disrupting beneficial orderings. As stated, Partially Mapped Crossover (PMX) is popularly used for MRTA problems and we apply this to the first part of our individual. This method creates an offspring by choosing a subsequence of tasks from a parent and preserves the order and positions of as many tasks as possible from the other parent. It ensures task sequences that minimise fitness are preserved while creating new offsprings. DEAP's built-in PMX crossover returns two offspring but we want one. We defined a custom crossover_single_child function that takes two parents, splits the task halves and applies DEAP's built-in PMX to return only one child as a new individual while retaining robot assignments. The crossover probability C1 is defined as a variable for experimentation. PMX is depicted in appendix B.

4.4.4 Mutation Operators

As stated, mutators generate diversity in the population, allowing us to search for previously unseen solutions. For the task part, merely mutating gene values would not lead to new permutations. Hence, we applied DEAP's shuffle index mutator which rearranges the positions of tasks, leading to new genomes. For the robot part, DEAP's uniform integer mutator was used to randomly replace each robot number with another in the range from 1 to m. This method effectively alternates between the m robots, resulting in a new assignment of robots to tasks. The custom mutate function divides and applies these mutators to each half. Each has their mutation probability, M1 and M2, defined as a variable for experimentation. The setup of the parameters and other operators are explained next.

4.4.5 NSGA-II Implementation

The implementation of NSGA-II was quite straightforward as DEAP allowed us to replace the tournament selector with the built in selNSGA2 selector. Next, instead of defining the function as weighted F3 we included both F1 and F2 as a set for the evaluation function. Necessary adjustments were made such as defining two fitness weights to minimise simultaneously. We then defined our survivor selector algorithm that controls the generation of the offspring population during initialisation. DEAP's library has a built-in function varOr() that does this by mutating a portion of the population (MUTPB) and crossing over (CXPB) the rest. Offsprings are resulted from either crossover or mutation but not both. However, the function was not compatible with our custom genome. Therefore, after consulting the

documentation we defined a modified varOrModified() function tailored for our genome. Lastly, we defined the pareto_eq() function that applied crowding distance to avoid similar solutions based on fitness. This was passed as a lambda function on DEAP's built in paretoFront functionality to store the pareto optimal solutions that were not too similar as a hall of fame. The hall of fame ensured we remembered optimal solutions from all previous generations. Since our focus was obtaining a suitable pareto front to explore the distance-time relationship it was not necessary to fully optimise our NSGA-II. Hence, we varied the proportion of mutation and crossover, MUTPB and CXPB, in conjunction with the rates C1, M1 and M2 until we obtained a relatively good front at large populations and generations. (MUTPB, CXPB = 0.5, M1, M2=0.1, C1=0.05, popSize=500, offspring(LAMBDA)=1000, generations=1000). Uniform PMX was also used to encourage more crossover since we were not mutating at the same time.

4.4.5 Hyperparameters and setup of Genetic Algorithm

To identify the optimal mutation and crossover probabilities for F3 and their effects on our GA performance we need to keep all other parameters constant while changing their values. For our three probability parameters, M1, M2 and C1, we define a list of rates to iterate through in order to find the optimal rate from 0.001 to 0.5 (e.g., a rate of 0.1 means a 10% likelihood).

Hyperparameters	**Values**
Population size	300
Number of generations	500
Selection operator	Tournament, size 10
Probability of task mutation: **M1**	[0.001, 0.005, 0.01, 0.05, 0.1, 0.5]
Probability of robot mutation: **M2**	[0.001, 0.005, 0.01, 0.05, 0.1, 0.5]
Probability of crossover: **C1**	[0.001, 0.005, 0.01, 0.05, 0.1, 0.5]

Table 2: Hyperparameters of F1 F2 and F3

The selection method used is a tournament with a default size of 10 to ensure the population evolves better solutions with a reasonably high selection pressure. The population is set relatively high at 300. If the size is too low, the search space is small, and we likely reach local optima. This enabled searching exhaustively without taking too long for our offline solution. The generations were set to 500. From initial test runs, most solutions converged to near optimal solutions around 200 generations, but we wanted to see the full GA potential. If convergence was not clearly shown the generations were extended as required. Lastly, appendix A depicts our genetic algorithm including how operators and hyperparameters interact to generate a solution and the 2D visualisation of the outcome of the task allocation and scheduling.

5 Results and evaluations

Taking into account the stochastic nature of genetic algorithms, each experiment was run 50 times per fitness function or parameter variant, ensuring repeatable and representative mean performance measures. Typically a success rate, based on a known criterion, measures performance. But this is unknown for most MRTA problems. Hence, we measure the performance via the mean best fitness (MBF) for time and distance. The study recorded the best solutions from the runs as our GA runs offline and centrally. Metrics such as standard deviation and best solutions were included to explore performance diversity. For efficiency we measured the mean generations to obtain a best fitness value. This was done by manually observing the data plots generated.

5.1 Evaluating F1 and F2: Distance versus time

In our initial evaluation of functions F1 and F2, we maintained constant mutation and crossover parameters at preliminary values. With C1= 0.01, M1 = 0.001, and M2 = 0.001, the results were satisfactory enough for a fair comparison between the two functions. Each run produced a fitness-over-generation graph, where a lower fitness score is better. F1 denotes fitness as average fuel consumption modelled after distance. We kept 21 tasks and 4 robots constant by default.

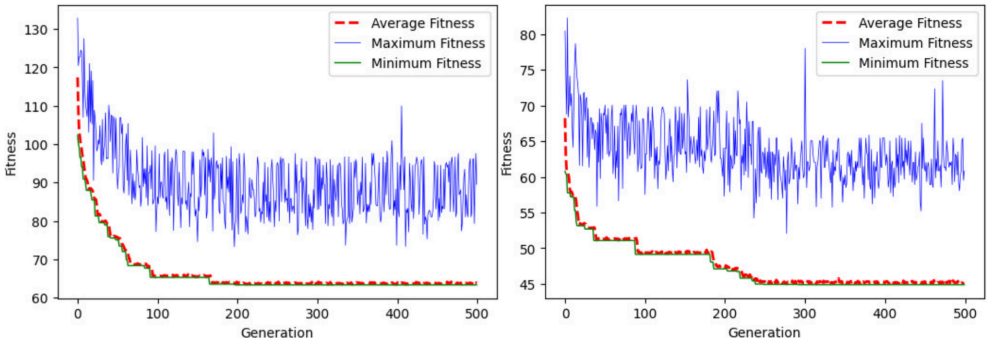


Fig 12 Fitness over generation graphs for F1: distance (left) and F2: time (right)

Figure 12 (left) depicts a typical fitness graph for F1, showing a swift initial convergence, as evidenced by the drastic decrease in distance from 118 to 63 in the first 100 generations. The tests generally reached quick convergence around 150 generations, followed by slight, gradual improvements before plateauing at approximately 200 generations. This trend is expected as the search space narrows over time, and more mutation and crossover are needed to discover better solutions.

Evaluation Metrics	Fitness Functions		
	F1	F2	F3
Mean Best Distance / MBF F1 (m)	61.60	79.87	68.70
Mean Best Time / MBF F2 (secs)	66.85	46.87	47.61
Best Solution Distance (m)	52.16	70.38	59.54
Best Solution Time (secs)	74.37	44.03	45.56
Standard Deviation of Distance	5.64	7.57	5.57
Standard Deviation of Time	10.12	1.98	2.19
Avg. Minimum Tasks Allocated	2.14	4.92	4.52
Avg Maximum Task Allocated	8.60	6.00	6.08
Mean Generations	120	209	175

Table 3 Evaluation metrics for fitness function at (C1=0.01, M1=0.001, M2=0.001)

Table 3 reveals that F1 yields a balanced mean distance of 62 m (MBF) and a mean time of 67 secs. Notably, the best solution achieved a significantly shorter distance of 52m but required more time, 74s. This prompted a qualitative analysis of the solution.

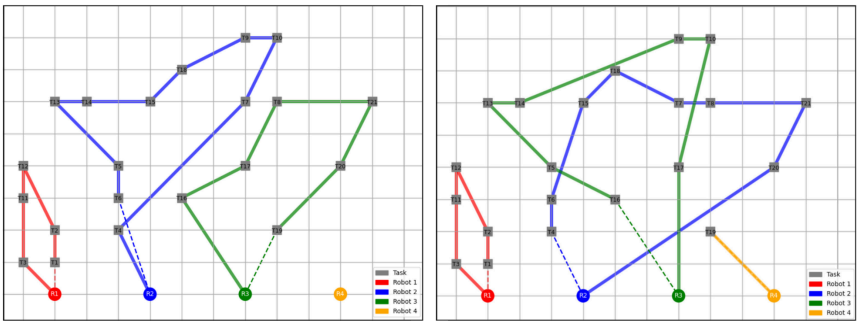


Fig 13 Best distance solution (left) and typical solutions (right) for F1

In Figure 13, the F1 best solution, which produced the shortest distance, did not use robot 4 (yellow). Thus, only three robots were used to complete the tasks. Other F1 solutions with minimal distances often assigned no tasks or only one or two tasks to certain robots while others were assigned more tasks. This trend is evident in Table 3, which displays a notable disparity between the average minimum and maximum tasks allocated per robot (2.14 and 8.60, respectively).

This uneven task distribution is attributed to the task allocator's tendency to assign tasks conservatively to robots, driven by a desire to save fuel - the less distance a robot travelled, the more fuel saved. However, this suggests that overemphasis on only minimising distance can undermine the goal of minimising time, as the reluctance to assign tasks efficiently to all robots meant fewer tasks were completed simultaneously. Additionally, the GA employs PMX well to preserve the beneficial similar allocations generated by mutators searching locally. Notice how robot 1 (red) in Figure 13 has the

same efficient path in both solutions. To explore further and improve time efficiency, we evaluated the F2 function, which minimises time. Figure 12 (right) shows that for F2, fitness also converges swiftly before plateauing. However, we observed a slower convergence rate compared to F1, with gradual decreases in time around 100 generations - thus generally slower to obtain stable solutions around 209 generations compared to F1. This may be due to more complex time calculations exhibiting a larger search space due to its simultaneous task allocation consideration.

Compared to F1, table 3 shows that F2 yields a mean distance of about 80 m and a lower mean time of 47 secs. The best solution also achieved a higher distance of 70m but achieved a more efficient time of 44 secs. While a lower time is expected since F2 minimises time, these results further suggest conflicts between time and distance This prompted a qualitative analysis of the F2 solutions.

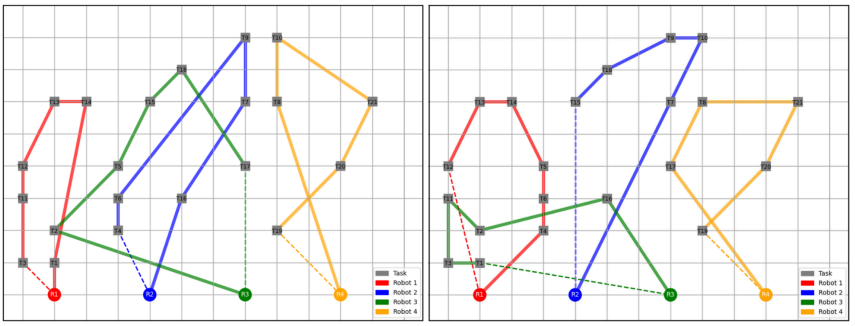


Fig 14 Best time solution (left) and typical solutions (right) for F2

In Figure 14, the F2 best solution, producing the shortest time, now uses all 4 robots efficiently unlike F1. This was seen in all other F2 solutions and is evident in Table 3, with less disparity between the average minimum and maximum tasks allocated per robot (4.92 and 6.0, respectively). While they had more efficient use of robots, we can see that, for both instances in figure 14, robot 3 and 4 (green, blue) branches out from their lane to complete more tasks quickly, even if they are further away.

This approach under F2 sacrifices fuel efficiency for faster task completion - prioritising reaching more tasks while other robots are en route saves time but increases distance and leads to more erratic robot paths that may complicate path planning in run time. This suggests that only minimising time undermines distance. While it is expected that shorter distances mean less time, a trade-off still exists as we cannot minimise one fully without increasing the other. To test statistical significance between F1, F2 and F3 we combined them and ran Shapiro-Wilk tests for normality. All three distances were normally distributed but F2 was not. Hence, we ran ANOVA test on the distances and Kruskal Wallis on time as F2 was not normally distributed. With p-value = 0.05 we ran post hoc Turkey HSD and Dunns test

respectively to obtain p-values. Table 4 shows that for both cases we obtain very low p-values. Hence, F1 significantly minimises distance compared to F2 while F2 minimises time more effectively. Hence, a clear tradeoff exists between distance and time and a weighted F3 function is needed to balance this. See appendix for the full data.

Hypotheses to test	Test depending On normality	p-value (post-hoc)	Significant difference?
Distance differences between F1 and F2	ANOVA	0.00	Yes
Time differences between F1 and F2	Kruskal-Wallis	3.468726e-16	Yes
Distance differences between F3 and F2	ANOVA	0.00	Yes
Time differences between F3 and F2	Kruskal-Wallis	0.8206379	No
Distance differences between F3 and F1	ANOVA	0.00	Yes
Time differences between F3 and F1	Kruskal-Wallis	1.515864e-15	Yes

Table 4: ANOVA and Kruskal Wallis tests for F1, F2 and F3 differences

5.2 Evaluating F3: Balancing time and distance

The F3 fitness function equally weights the distance travelled and time taken in order to balance multiple objectives. Table 3 shows we achieve this with a mean distance of 69m and mean time of 48 secs, both in between F1 and F2 results. The figures below illustrate this relationship further as each function has different outcomes.

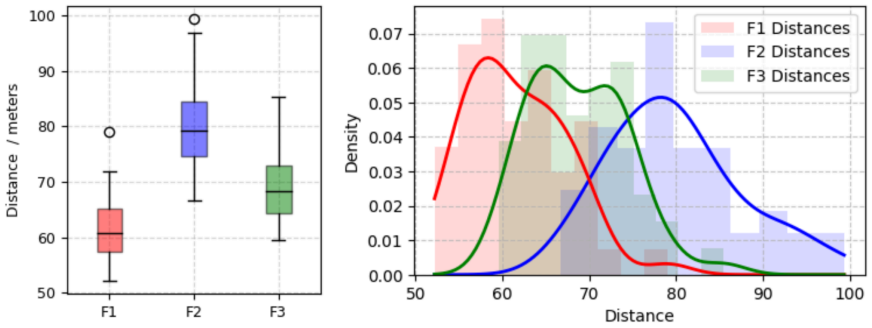


Fig 15 Comparing distance travelled F1, F2, F3: Box plot (left) and histogram (right)

Figure 15 shows how our functions compare in minimising distance. As stated, F1 minimises distance the most with a tight concentration of lower distances in red whereas F2 struggles with higher distances and a wider spread in blue. There is minor overlap between them due to outliers as GA performance was not optimal. Notably, F3 is a middle ground between F1 and F2, with higher distance than F1 but lower distances than F2. This observation is statistically significant as distance differences for F3 between F1 and F2 are at very low p-values in table 4.

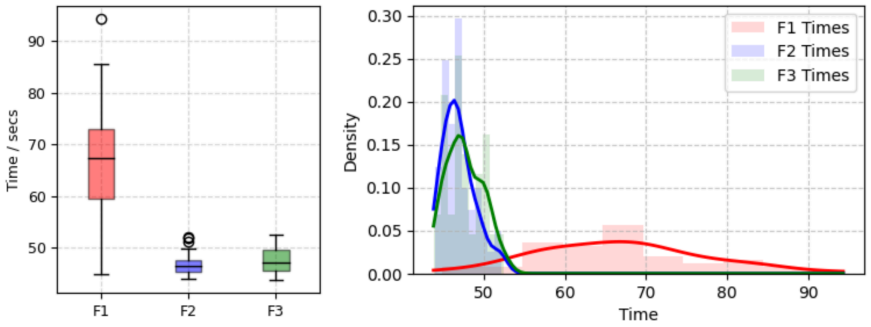


Fig 16 Comparing time taken F1, F2, F3: Box plot (left) and histogram (right)

Next, Figure 16 illustrates the comparisons in minimising time. As stated, we observe the inverse relationship with distance and time as now F2 minimises time the most with a tight concentration of low times in blue whereas F1 struggles with higher times and a wider spread in red. Contrary to expectations of a middle ground, F3 displays a close alignment with F2's lower time results, shown in green. It performed better than a middle ground for time taken.

This aligns with table 4 as time differences for F3 between F1 is significant at low p-values but insignificant between F2 at 0.820. This strongly suggests that via F3, by compensating for a slight increase in distance travelled compared to F1 we can reliably obtain a better time performance similar to F2 that was previously unreachable via F1 while also minimising distance compared to F2. F3 also provides more reliable distance and time results. Evidenced by the tight distributions in Figures 15 and 16 and lower standard deviation of 5.5 and 2.2 for distance and time respectively in table 3.

Fig 17 shows the best and typical balanced solutions for F3. The best solution had a distance of 59m and a time of 46 secs. It is the most optimal solution from our results for the default scenario of 21 tasks and 4 robots. The distance is balanced between F1 and F2's best solution and the time is only slightly higher than F2s.

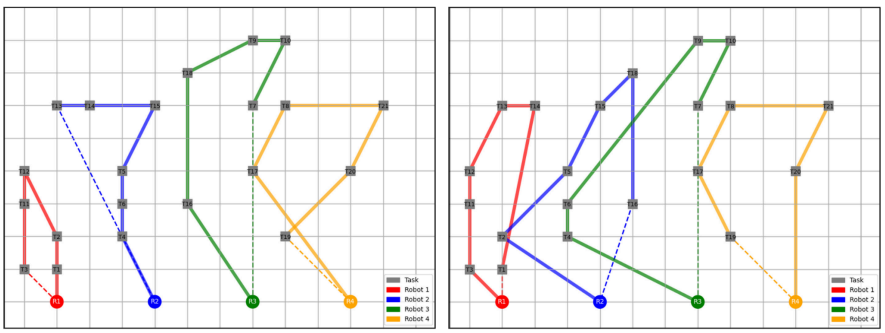


Fig 17 Best balanced solution (left) and typical solutions (right) for F3

In these instances, robots are efficiently allocated to well distributed tasks unlike F1. Notably, robots form distinct, almost circular, paths that are less erratic than F2. Robots stay in their lanes and are less prone to collisions due to less overlap. Unlike F2, there is a reluctance to deviate towards tasks further away. In summary, F3 successfully achieves a beneficial middle ground. It balances the trade-offs between time and distance travelled, providing good solutions. Next we find optimal parameters for F3.

5.3 Optimising F3: Crossover and mutation rates

To ensure our crossover and mutation operators converge to good solutions quickly, we need to tune their rates. Excessive mutation results in random searching. Similarly, low crossover rates slow convergence, while high rates can reduce diversity. Thus, we find optimal rates for crossover (C1) and mutations (M1 and M2).

For the GA, which only needs to provide a good solution before the mission, parameter tuning was used by varying one parameter rate at a time from a set (0.001,0.005,0.01,0.05,0.1,0.5) while keeping the others fixed at 0.01 (C1) and 0.001 (M1 and M2). A total of 6 rate values ranging from 0.5 to 0.001, were incremented by alternating factors of 5 and 2. This approach allows us to explore rates by a factor of 10, including their midpoints. The maximum value of 0.5 prevents excessive mutation and crossover, avoiding too much randomness and mutation rates rarely go below 0.001.

Due to time constraints, we used a fixed set of values instead of finding exact optimal rates. This approach is simpler than adaptive parameter control, where values change during the run. Adaptive control is complex and better suited for dynamic tasks, which are beyond our project scope. For our static problem, parameter tuning was sufficient and is commonly used for finding optimal initial rates.

The results are illustrated in Figure 16 below. The rates across C1, M1 and M2 were plotted against the normalised mean performances - fitness, distance and time. The lower these values the better the performance. The best performing rate is highlighted in yellow.

Figure 18 (a) suggests the best rate for C1 crossover is 0.05. Rates higher and lower than that lead to slightly worse performance possibly due to low diversity and slow convergence respectively. However, at a rate of 0.005, performance dips significantly. This suggests that at certain low crossover rates, in combination with the fixed mutation rates of M1 and M2 at 0.001, we might not converge to good solutions quick enough.

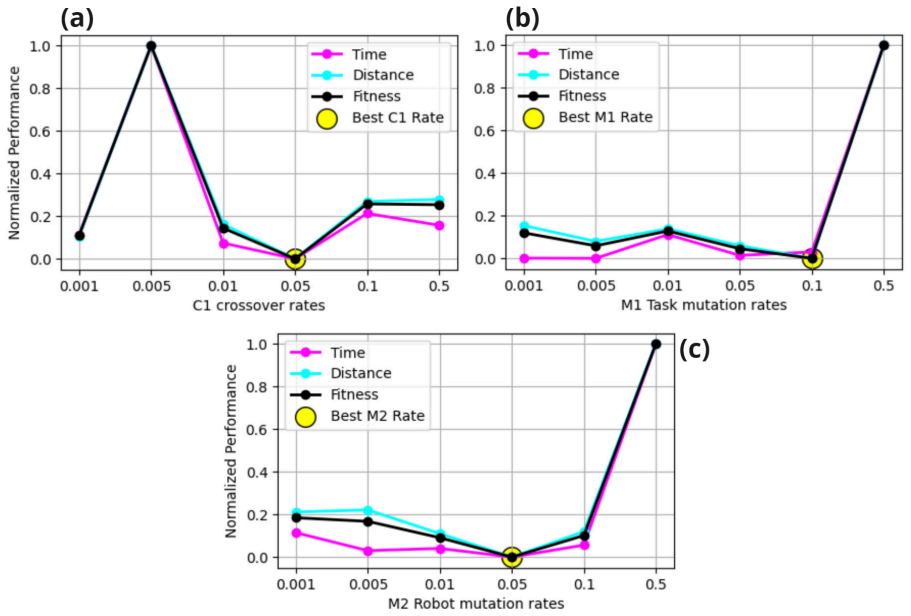


Fig 18 Rate vs performance for C1 (a), M1-Tasks (b), M2-Robots (c)

Moving on to plots b and c in Figure 18, in both cases, from left to right, higher mutation rates lead to better performance (lower values) as it enables us to search more exhaustively till we reach the optimal rate of mutation. It suggests that the optimal rate of shuffle mutation of tasks (M1) is 0.1. Similarly, the rate of flip mutation of robot assignments (M2) is 0.05. After this optimal point the performance worsens significantly as mutations become so frequent that we start searching randomly. This is depicted in the fitness plots in Appendix C where the lines never go down. Hence these results suggest that the optimal hyperparameters for F3 from our defined set are C1=0.05, M1=0.1 and M2=0.05.

The overall mutation rate is higher than crossover which suggests that in our GA, performance is better when the ratio of mutation to crossover is higher. Furthermore, M1 being higher than M2 (0.1 > 0.05) suggests that our GA relies more on mutating tasks more frequently than mutating robot assignments. This could be due to the setup as shuffling of tasks has more effect on fitness than robot assignments as tasks are more spread out than robots. It could also be influenced by the fact that shuffling is inherently less frequent than flipping each robot gene.

For our static offline problem we are generally not concerned with the number of generations taken. However, it is interesting to note that while M1 and M2 were optimal, as per Appendix C, it generally took significantly more generations (approx. 400) to converge according to their fitness plot. Lower mutation rates of 0.01 converged much quicker (approx. 100 gen) to similar but not as good solutions. Hence, lower mutations might be better for problems that are dynamic or with fewer generations.

5.4 Exploring the pareto front: F3 vs NSGA - II

While the weighted F3 function performs well it does not let us visualise the pareto front of best solutions. To explore the pareto front our NSGA-II solutions were plotted. This was compared with solutions from F1, F2 and F3 to evaluate the front. Ideally, the pareto front should be close to the bottom right with a diverse spread of solutions.

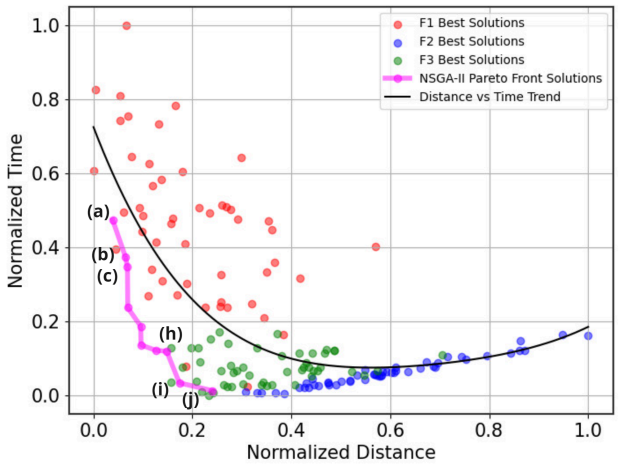


Fig 19 Scatter plot of best solutions from F1, F2, F3 and the pareto Front (50 runs)

The scatter plot in Figure 19 shows the best solutions across all three functions (F1, F2, F3) and NSGA-II. It complements our prior results by illustrating the distinct clusters of solutions. While F3 provides some good solutions, they are too concentrated in one area, failing to give a clear visual of the pareto front.

Using NSGA-II, we produced a pareto front with 10 equally good solutions at the edge of the feasible solution space. It requires multiple runs to obtain a suitable front. The pareto front displays the basic shape of an effective multi-objective optimization result. Paired with the spread of solutions from the three functions, the general shape of the pareto front emerges as a convex curve.

The steep initial segment of the pareto front suggests that small increases in distance can lead to significant reductions in time. For a factory setting, this means that slightly extending the travel path of robots could considerably decrease the total inspection time. This strategy could be particularly beneficial in scenarios where time is a critical factor, such as in high-production periods or during time-sensitive maintenance checks. As the front becomes less steep, further increases in distance yield smaller time improvements. This suggests that beyond a certain point, making robots travel longer distances does not effectively reduce inspection time. In practice, this indicates the need to find an optimal balance where the

additional energy and wear on robots from longer distances do not outweigh the marginal time savings. Importantly, all solutions along the front are equally good and thus allowing room for subjective choice depending on our priorities in different contexts. Figure 19 shows the range of pareto optimal solutions.

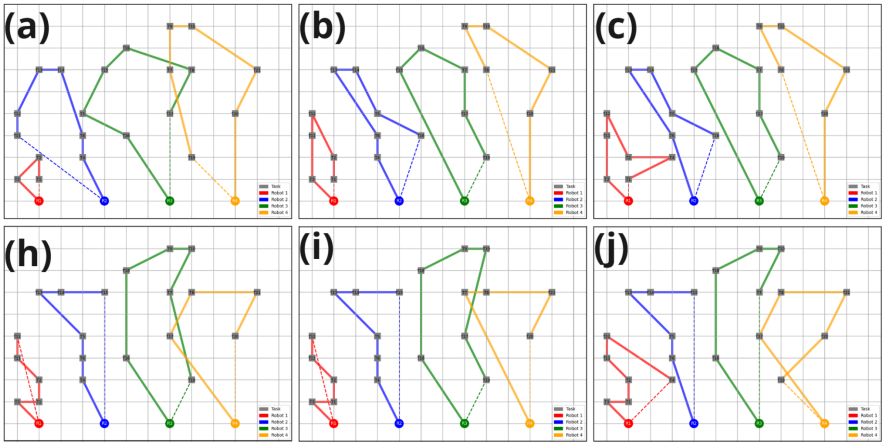


Fig 20 Optimal solutions along the pareto front

Here, (a) represents a solution on the top end of the pareto front (Figure 20) which prioritises minimising distance, as robot 1 (red) has a shorter path. Conversely, we have much longer and balanced paths in (j) - the other end of the pareto front that minimises time. Unlike choosing between F1, F2 and F3, all solutions across the front are equally good. Moving along the front from left to right (a to j) we go from minimising distance to balanced solutions to minimising time. Depending on the scenario, we might choose top-left solutions to minimise distance (e.g., when low on fuel), or bottom-right solutions to prioritise time (e.g., when tasks need to be completed quickly). In every case, the solutions are equally effective.

With just 10 solutions, the pareto front is relatively small and less diverse compared to the solution space range. This suggests that while NSGA-II is finding efficient trade-offs, it could benefit from more advanced or self-adaptive parameter tuning. The actual pareto front might be closer to the two axes than shown in Fig. 20, as further indicated by a few F3 solutions slightly exceeding the pareto front in Fig. 19.

Although implementing an advanced NSGA-II is beyond this project's scope, we successfully showcased a pareto set of equally good solutions, majority offering better trade-offs than F3's solutions, except for minor outliers. While F3 is simple to develop and execute, it rarely performs better and has a larger variance across its solutions. In contrast, NSGA-II consistently provides optimal solutions closer to the front and more diverse than F3's. Additionally, it allows us to select from equally good solutions based on our current needs rather than manually setting weights via F3.

6 Conclusions

The aim of our project was to iteratively develop a genetic algorithm to solve a multiobjective MRTA problem by minimising both time and distance. We had 5 objectives to obtain this as per table 1. For the first aim we were able to successfully design and implement a genetic algorithm along with appropriate operators using DEAP and Python. In doing so we successfully minimised for both time and distance across all our functions F1, F2 and F3.

Secondly, we were able to visualise our solutions in a clear and efficient manner using matplotlib for our offline static solution. We were able to obtain a level of customizability that was not attainable through other software available to us. However, this means that complicating the problem further for more real world problems would become unfeasible. In this instance, future work could include reducing the reality gap by implementing our solution in a 3D simulator, time willing, or even with real robots to further validate our results.

We successfully designed and implemented our base fitness functions F1 and F2 and obtained results that were consistent with previous such work . The weighted fitness function (F3) effectively balanced the trade-offs between minimising distance and time. It provided better overall performance compared to the single-objective functions (F1 and F2) by offering solutions that were both fuel-efficient and time-effective. Our experiments demonstrated that while F1 minimised distance and F2 minimised time, F3 achieved a balanced middle ground, offering reliable and efficient solutions. This prompted statistical testing which strongly suggests that distance and time have some level of conflicting relationship. This was interesting as one would normally expect distance and time to complement each other.

Using NSGA-II, we successfully explored the pareto front, identifying a set of optimal solutions that highlighted the trade-offs between objectives. This allowed us to understand the relationship between distance and time, providing insights into how slight increases in travel distance could significantly reduce inspection time - a crucial finding for practical applications in dynamic and time-sensitive environments. Our findings of a convex pareto front is consistent with other works in the same area

Visualising the pareto front provided valuable insights into the trade-offs between distance and time. This approach can be applied beyond plant inspections to other MRTA scenarios, such as search and rescue, warehouse logistics, and environmental monitoring. The NSGA-II solutions allow for subjective selection based on context. This flexibility is crucial for

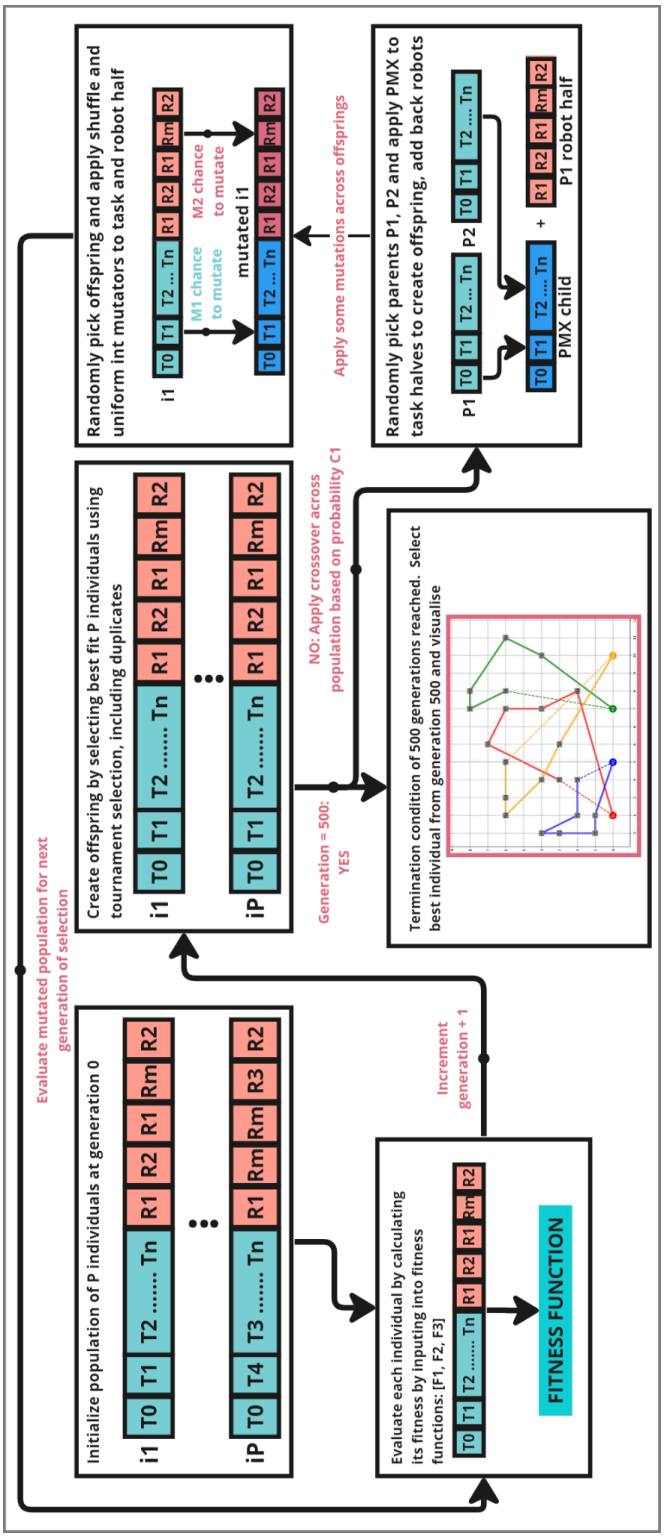
real-world applications where priorities can change based on immediate needs, such as fuel availability or urgent task completion. Future work could include a scenario where a human controller is presented with a pareto set of solutions and they select the solution based on their context.

While all our project aims were successfully achieved, several limitations and areas for future work were identified. In general, these were unachievable as they were beyond the project scope. Due to time constraints, we relied on a fixed set of parameters from a small subset of values instead of using parameter control to change our crossover and mutation rates. While this provided insights regarding the ratio of mutation to crossover, future research could implement self-adaptive mutation rates and crossover probabilities to further explore the validity of these observations while enhancing the performance of both F3 and the NSGA-II solutions.

We did not attempt all possible weights for the F3 function. This was mainly due to the time constraints with the project itself but future studies could explore a wider range of weights to better understand their impact on solution quality and better evaluate F3. Similarly, the study focused on a fixed number of tasks and robots. Testing the algorithm with varying numbers of tasks and robots would help assess its scalability and limitations to larger, more complex environments. For the sake of our project the main goal was the comparison of the fitness functions and evaluation of the solutions. Not ensuring algorithmic efficiency. This study addressed a static offline problem with no obstacles. Future work could explore the generalisability of our results across various maps with different obstacles or placement of tasks and robots. Extending the research to dynamic MRTA scenarios, where tasks and environmental conditions change in real-time, would provide a more comprehensive understanding of the algorithm's applicability. Implementing a more advanced version of NSGA-II with fine-grained parameter tuning could produce a more diverse and efficient pareto front, enhancing solution quality. Additionally, recent research has explored more efficient genome encodings in the form of a single array that does not require splitting between tasks and robots - hence allowing for more exhaustive search across MRTA solutions and the use of less computationally intensive genetic operators.

Nonetheless, we were able to achieve all of our research aims successfully with minor improvements needed. Most of which were beyond our scope. By balancing distance and time, we provided a robust framework for optimising task allocation. Our findings fortify the existing but sparse body of research on balancing distance and time and generate new insights by examining why these trade-offs actually occur.

Appendices:



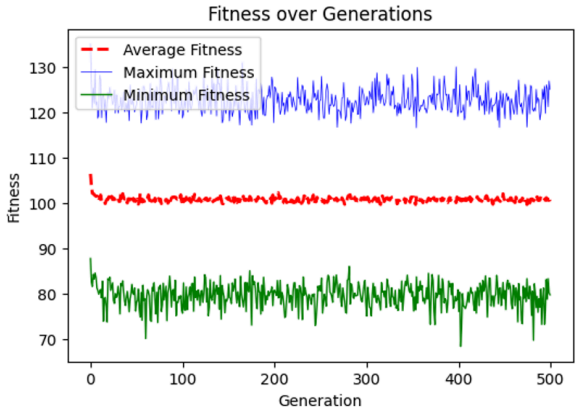
(A) MAIN DESIGN OF GENETIC ALGORITHM WITH ITS OPERATORS

Fig. 4.12. PMX, step 1: copy randomly selected segment from first parent into offspring

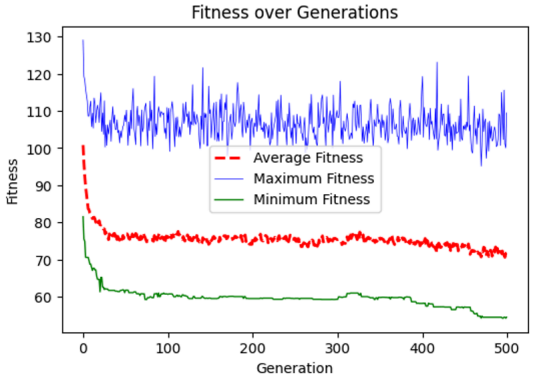
Fig. 4.13. PMX, step 2: consider in turn the placement of the elements that occur in the middle segment of parent 2 but not parent 1. The position that 8 takes in P2 is occupied by 4 in the offspring, so we can put the 8 into the position vacated by the 4 in P2. The position of the 2 in P2 is occupied by the 5 in the offspring, so we look first to the place occupied by the 5 in P2, which is position 7. This is already occupied by the value 7, so we look to where this occurs in P2 and finally find a slot in the offspring that is vacant – the third. Finally, note that the values 6 and 5 occur in the middle segments of both parents.

Fig. 4.14. PMX, step 3: copy remaining elements from second parent into same positions in offspring

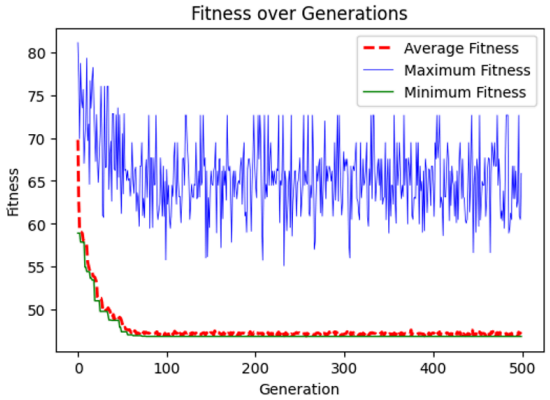
(B) PMX CROSSOVER FULLY EXPLAINED



Too might mutation rates at 0.5 lead to random search



At optimal mutation rates it took roughly longer (400 gens) to find best solution



At lower mutation rates, we obtain a similar but not as good solution in less gens.

(C) MUTATION RATE FLUCTUATIONS

Bibliography

[1] A. Khamis et al., ‘Multi-robot Task Allocation: A Review of the State-of-the-Art’, in *Cooperative Robots and Sensor Networks 2015*, A. Koubâa and J. R. Martínez-de Dios, Eds. Cham: Springer International Publishing, 2015, pp. 31–51[Online]. Availablehttps://doi.org/10.1007/978-3-319-18299-5_2.

[2] H. Chakraa et al., ‘Optimization techniques for Multi-Robot Task Allocation problems: Review on the state-of-the-art’, *Rob. Auton. Syst.*, vol. 168, p. 104492, Oct. 2023[Online]. Availablehttps://www.sciencedirect.com/science/article/pii/S0921889023001318.

[3] B. P. Gerkey and M. J. Matarić, ‘A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems’, *Int. J. Rob. Res.*, vol. 23, no. 9, pp. 939–954, Sep. 2004[Online]. Availablehttps://doi.org/10.1177/0278364904045564.

[4] G. A. Korsah et al., ‘A comprehensive taxonomy for multi-robot task allocation’, *Int. J. Rob. Res.*, vol. 32, no. 12, pp. 1495–1512, Oct. 2013[Online]. Availablehttps://doi.org/10.1177/0278364913496484.

[5] F. Luan et al., ‘Industrial robots and air environment: A moderated mediation model of population density and energy consumption’, *Sustainable Production and Consumption*, vol. 30, pp. 870–888, Mar. 2022[Online]. Availablehttps://www.sciencedirect.com/science/article/pii/S2352550922000161.

[6] R. Sparrow and M. Howard, ‘Robots in agriculture: prospects, impacts, ethics, and policy’, *Precis. Agric.*, vol. 22, no. 3, pp. 818–833, Jun. 2021[Online]. Availablehttps://doi.org/10.1007/s11119-020-09757-9.

[7] E. Nunes et al., ‘A taxonomy for task allocation problems with temporal and ordering constraints’, *Rob. Auton. Syst.*, vol. 90, pp. 55–70, Apr. 2017[Online]. Availablehttps://www.sciencedirect.com/science/article/pii/S0921889016306157.

[8] T. Bektas, ‘The multiple traveling salesman problem: an overview of formulations and solution procedures’, *Omega*, vol. 34, no. 3, pp. 209–219, Jun. 2006[Online]. Availablehttps://www.sciencedirect.com/science/article/pii/S0305048304001550.

[9] Q. Li et al., ‘Distributed Near-optimal Multi-robots Coordination in Heterogeneous Task Allocation’, in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 4309–4314[Online]. Availablehttp://dx.doi.org/10.1109/IROS45743.2020.9341652.

[10] R. Almadhoun et al., ‘A survey on inspecting structures using robotic systems’, *Int. J. Adv. Rob. Syst.*, vol. 13, no. 6, p. 1729881416663664, Dec. 2016[Online]. Availablehttps://doi.org/10.1177/1729881416663664.

[11] K. Jose and D. K. Pratihar, ‘Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods’, *Rob. Auton. Syst.*, vol. 80, pp. 34–42, Jun. 2016[Online]. Availablehttps://www.sciencedirect.com/science/article/pii/S0921889016000282.

[12] K. Cai, ‘Warehouse automation by logistic robotic networks: a cyber-physical control approach’, *Frontiers of Information Technology & Electronic Engineering*, vol. 21, no. 5, pp. 693–704, May 2020[Online]. Availablehttps://doi.org/10.1631/FITEE.2000156.
[13] P. R. Wurman et al., ‘Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses’, *AIMag*, vol. 29, no. 1, pp. 9–9, Mar. 2008[Online]. Availablehttps://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/2082[Accessed: 15May2024].
[14] Z. Luo et al., ‘The Multi-visit Traveling Salesman Problem with Multi-Drones’, *Transp. Res. Part C: Emerg. Technol.*, vol. 128, p. 103172, Jul. 2021[Online]. Availablehttps://www.sciencedirect.com/science/article/pii/S0968090X2100190X.
[15] F. Nekovar et al., ‘Multi-tour set traveling salesman problem in planning power transmission line inspection’, *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 6196–6203, Oct. 2021[Online]. Availablehttps://ieeexplore.ieee.org/document/9463765/.
[16] P. Ghassemi and S. Chowdhury, ‘Multi-robot task allocation in disaster response: Addressing dynamic tasks with deadlines and robots with range and payload constraints’, *Rob. Auton. Syst.*, vol. 147, p. 103905, Jan. 2022[Online]. Availablehttps://www.sciencedirect.com/science/article/pii/S0921889021001901.
[17] M. J. Matarić et al., ‘Multi-Robot Task Allocation in Uncertain Environments’, *Auton. Robots*, vol. 14, no. 2, pp. 255–263, Mar. 2003[Online]. Availablehttps://doi.org/10.1023/A:1022291921717.
[18] K. Lerman et al., ‘Analysis of Dynamic Task Allocation in Multi-Robot Systems’, *Int. J. Rob. Res.*, vol. 25, no. 3, pp. 225–241, Mar. 2006[Online]. Availablehttps://doi.org/10.1177/0278364906063426.
[19] P. K. Padmanabhan et al., ‘Multi-objective Optimisation of Multi-robot Task Allocation with Precedence Constraints’, *Def. Sci. J.*, vol. 68, no. 2, pp. 175–182, Mar. 2018[Online]. Availablehttps://publications.drdo.gov.in/ojs/index.php/dsj/article/view/11187[Accessed: 15May2024].
[20] Y. U. Cao et al., ‘Cooperative Mobile Robotics: Antecedents and Directions’, *Auton. Robots*, vol. 4, no. 1, pp. 7–27, Mar. 1997[Online]. Availablehttps://doi.org/10.1023/A:1008855018923.
[21] B. Horling and V. Lesser, ‘A survey of multi-agent organizational paradigms’, *Knowl. Eng. Rev.*, vol. 19, no. 4, pp. 281–316, Dec. 2004[Online]. Availablehttps://www.cambridge.org/core/journals/knowledge-engineering-review/article/survey-of-multiagent-organizational-paradigms/A07BCCB1379F001DE995F3E5476EE4AB[Accessed: 15May2024].
[22] R. M. Zlot, (2006, December.1), *An auction-based approach to complex task allocation for multirobot teams*, Robotics Institute Carnegie Mellon University. [Online]. Available: https://www.ri.cmu.edu/publications/an-auction-based-approach-to-complex-task-allocation-for-multirobot-teams/. [Accessed: 15 May 2024].
[23] F. Tang and L. E. Parker, ‘A Complete Methodology for Generating Multi-Robot Task Solutions using ASyMTRe-D and Market-Based Task Allocation’, in

Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007, pp. 3351–3358[Online]. Availablehttp://dx.doi.org/10.1109/ROBOT.2007.363990.

[24] Smith, 'The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver', *IEEE Trans. Comput.*, vol. C–29, no. 12, pp. 1104–1113, Dec. 1980[Online]. Availablehttp://dx.doi.org/10.1109/TC.1980.1675516.

[25] M. Dias et al., 'Traderbots: a new paradigm for robust and efficient multirobot coordination in dynamic environments', 2004[Online]. Availablehttps://search.proquest.com/openview/73eb6e02940ed4c4ba72ef5371cab5f8/1?pq-origsite=gscholar&cbl=18750&diss=y.

[26] M. B. Dias et al., 'Market-Based Multirobot Coordination: A Survey and Analysis', *Proc. IEEE*, vol. 94, no. 7, pp. 1257–1270, Jul. 2006[Online]. Availablehttp://dx.doi.org/10.1109/JPROC.2006.876939.

[27] W. P. N. Dos Reis and G. S. Bastos, 'Multi-Robot Task Allocation Approach Using ROS', in *2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)*, 2015, pp. 163–168[Online]. Availablehttp://dx.doi.org/10.1109/LARS-SBR.2015.20.

[28] B. B. Werger and M. J. Mataric, 'Broadcast of local eligibility: behavior-based control for strongly cooperative robot teams', in *Proceedings of the fourth international conference on Autonomous agents*, Barcelona, Spain, 2000, pp. 21–22[Online]. Availablehttps://doi.org/10.1145/336595.336621[Accessed: 15May2024].

[29] J. G. Martin et al., 'Multi-robot task allocation clustering based on game theory', *Rob. Auton. Syst.*, vol. 161, p. 104314, Mar. 2023[Online]. Availablehttps://www.sciencedirect.com/science/article/pii/S0921889022002032.

[30] W. P. N. dos Reis et al., 'An arrovian analysis on the multi-robot task allocation problem: Analyzing a behavior-based architecture', *Rob. Auton. Syst.*, vol. 144, p. 103839, Oct. 2021[Online]. Availablehttps://www.sciencedirect.com/science/article/pii/S092188902100124X.

[31] M. Badreldin et al., 'A Comparative Study between Optimization and Market-Based Approaches to Multi-Robot Task Allocation', *Advances in Artificial Intelligence*, vol. 2013, Nov. 2013[Online]. Availablehttps://www.hindawi.com/journals/aai/2013/256524/abs/[Accessed: 15May2024].

[32] S. Ismail and L. Sun, 'Decentralized hungarian-based approach for fast and scalable task allocation', in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017, pp. 23–28[Online]. Availablehttp://dx.doi.org/10.1109/ICUAS.2017.7991447.

[33] J. C. Spall, 'Stochastic Optimization', in *Handbook of Computational Statistics: Concepts and Methods*, J. E. Gentle, W. K. Härdle, and Y. Mori, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 173–201[Online]. Availablehttps://doi.org/10.1007/978-3-642-21551-3_7.

[34] S. Pandian and V. Modrák, 'Possibilities, Obstacles And Challenges Of Genetic Algorithm In Manufacturing Cell Formation', *Advanced Logistic systems*, vol. 3, no. 1, pp. 63–70, 2009[Online].

Availablehttps://ideas.repec.org/a/pcz/alspcz/v3y2009i1p63-70.html[Accessed: 16May2024].
[35] H. W. Kuhn, 'The Hungarian method for the assignment problem', *Nav. Res. Logist. Q.*, vol. 2, no. 1–2, pp. 83–97, Mar. 1955[Online]. Availablehttps://onlinelibrary.wiley.com/doi/10.1002/nav.3800020109.
[36] P. J. M. van Laarhoven and E. H. L. Aarts, 'Simulated annealing', in *Simulated Annealing: Theory and Applications*, P. J. M. van Laarhoven and E. H. L. Aarts, Eds. Dordrecht: Springer Netherlands, 1987, pp. 7–15[Online]. Availablehttps://doi.org/10.1007/978-94-015-7744-1_2.
[37] J. Chen et al., 'A Multi-Robot Task Allocation Method Based on Multi-Objective Optimization', in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2018, pp. 1868–1873[Online]. Availablehttp://dx.doi.org/10.1109/ICARCV.2018.8581110.
[38] M. Dorigo and G. Di Caro, 'Ant colony optimization: a new meta-heuristic', in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, 1999, vol. 2, pp. 1470–1477 Vol. 2[Online]. Availablehttp://dx.doi.org/10.1109/CEC.1999.782657.
[39] J. Wang et al., 'Multi-robot task allocation based on ant colony algorithm', *J. Comput.*, vol. 7, no. 9, Sep. 2012[Online]. Availablehttps://www.academia.edu/download/31441016/8296-18128-1-PB.pdf.
[40] M. Shelkamy et al., 'Comparative Analysis of Various Optimization Techniques for Solving Multi-Robot Task Allocation Problem', in *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, 2020, pp. 538–543[Online]. Availablehttp://dx.doi.org/10.1109/NILES50944.2020.9257967.
[41] Z. Li and X. Li, 'Genetic algorithm for task allocation and path planning of multi-robot system', vol. 4, pp. 34–38, Nov. 2016[Online]. Availablehttps://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=a718ff284ca365c937104f56fc2aa8cc79a1f01d.
[42] C. A. Coello Coello, 'An Introduction to Evolutionary Algorithms and Their Applications', in *Advanced Distributed Systems*, 2005, pp. 425–442[Online]. Availablehttp://dx.doi.org/10.1007/11533962_39.
[43] S. Katoch et al., 'A review on genetic algorithm: past, present, and future', *Multimed. Tools Appl.*, vol. 80, no. 5, pp. 8091–8126, 2021[Online]. Availablehttp://dx.doi.org/10.1007/s11042-020-10139-6.
[44] K. Jebari et al., 'Selection methods for genetic algorithms', *International Journal of Emerging Sciences*, vol. 3, no. 4, pp. 333–344, 2013[Online]. Availablehttps://www.researchgate.net/profile/Khalid-Jebari/publication/259461147_Selection_Methods_for_Genetic_Algorithms/links/5480d9100cf22525dcb60519/Selection-Methods-for-Genetic-Algorithms.pdf.
[45] D. N. Mudaliar and N. K. Modi, 'Unraveling Travelling Salesman Problem by genetic algorithm using m-crossover operator', in *2013 International Conference on Signal Processing , Image Processing & Pattern Recognition*, 2013, pp. 127–130[Online]. Availablehttp://dx.doi.org/10.1109/ICSIPR.2013.6497974.
[46] D. E. Goldberg, 'Robert Jr. Linge. Alleles, loci, and the traveling salesman

problem', *Proceedings of the First International Conference on*.
[47] A. Hassanat et al., 'Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach', *Information*, vol. 10, no. 12, p. 390, Dec. 2019[Online]. Availablehttps://www.mdpi.com/2078-2489/10/12/390[Accessed: 16May2024].
[48] K. Deb, 'Search methodologies', *Springer, DOI*, vol. 10, pp. 0–387, 2014.
[49] N. Saini and S. Saha, 'Multi-objective optimization techniques: a survey of the state-of-the-art and applications', *Eur. Phys. J. Spec. Top.*, vol. 230, no. 10, pp. 2319–2335, Sep. 2021[Online]. Availablehttps://doi.org/10.1140/epjs/s11734-021-00206-w.
[50] X.-S. Yang, 'Chapter 14 - Multi-Objective Optimization', in *Nature-Inspired Optimization Algorithms*, X.-S. Yang, Ed. Oxford: Elsevier, 2014, pp. 197–211[Online]. Availablehttps://www.sciencedirect.com/science/article/pii/B9780124167438000142.
[51] K. Deb et al., 'A fast and elitist multiobjective genetic algorithm: NSGA-II', *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002[Online]. Availablehttp://dx.doi.org/10.1109/4235.996017.
[52] D. X. Huo et al., 'Multi-objective energy-saving job-shop scheduling based on improved NSGA-II', *Int. J. Simul. Model.*, vol. 19, no. 3, pp. 494–504, Sep. 2020[Online]. Availablehttp://www.ijsimm.com/Full_Papers/Fulltext2020/text19-3_CO12.pdf.
[53] I. M. Ali et al., 'An Automated Task Scheduling Model Using Non-Dominated Sorting Genetic Algorithm II for Fog-Cloud Systems', *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2294–2308, 01 Oct.-Dec 2022[Online]. Availablehttp://dx.doi.org/10.1109/TCC.2020.3032386.
[54] W. Xu et al., 'Improved NSGA-II to solve a novel multi-objective task allocation problem with collaborative tasks', *Proc. Inst. Mech. Eng. Pt. D: J. Automobile Eng.*, vol. 236, no. 14, pp. 3106–3123, Dec. 2022[Online]. Availablehttps://doi.org/10.1177/09544070211072665.
[55] J. Zhou et al., 'Task Allocation in Multi-agent Systems Using Many-objective Evolutionary Algorithm NSGA-III', in *Machine Learning and Intelligent Communications*, 2019, pp. 677–692[Online]. Availablehttp://dx.doi.org/10.1007/978-3-030-32388-2_56.
[56] P. C. Pop et al., 'A comprehensive survey on the generalized traveling salesman problem', *Eur. J. Oper. Res.*, vol. 314, no. 3, pp. 819–835, May 2024[Online]. Availablehttps://www.sciencedirect.com/science/article/pii/S0377221723005581.
[57] L. Pitonakova et al., 'Feature and performance comparison of the V-REP, gazebo and ARGoS robot simulators', in *Towards Autonomous Robotic Systems*, Cham: Springer International Publishing, 2018, pp. 357–368[Online]. Availablehttp://link.springer.com/10.1007/978-3-319-96728-8_30.
[58] S. Macenski et al., 'Robot Operating System 2: Design, architecture, and uses in the wild', *Sci Robot*, vol. 7, no. 66, p. eabm6074, May 2022[Online]. Availablehttp://dx.doi.org/10.1126/scirobotics.abm6074.

[59] S. Tisue, ‘NetLogo : Design and implementation of a multi-agent modeling environment’, 2004[Online]. Availablehttp://www.ccl.sesp.northwestern.edu/papers/2013/netlogo-agent 2004c.pdf.
[60] M. Bettini et al., ‘VMAS: A Vectorized Multi-agent Simulator for Collective Robot Learning’, in *Distributed Autonomous Robotic Systems*, 2024, pp. 42–56[Online]. Availablehttp://dx.doi.org/10.1007/978-3-031-51497-5_4.
[61] *Mesa: Agent-based modeling in Python — Mesa .1 documentation*. [Online]. Available: https://mesa.readthedocs.io/en/stable/. [Accessed: 16 May 2024].
[62] *DEAP documentation — DEAP 1.4.1 documentation*. [Online]. Available: https://deap.readthedocs.io/en/master/. [Accessed: 16 May 2024].