A Comparative Analysis of Multi-Robot System Strategies for the Automation of Green Wall Maintenance

Sylvester Wachira Ndaiga[1]

MSc Robotics and Computation

Dr. Steven Hailes

Submission date: 03 September 2018

Abstract

This paper presents an analysis of two operational strategies of a UAV-based, Multi Robot System tasked with performing the robotic maintenance of a Green Wall System in simulation. Hypothesis testing is employed to present a statistical comparison of swarm-inspired vs generic lawn mower motion approaches to coverage path planning and cooperative control in the robot collective. Each agent localizes, classifies and acts upon randomly distributed coplanar targets, functionally symbolic of a plant-abundant, vertical wall planter. Global objectives are probabilistically respected by locally acting agents via limited sensing and communication techniques. Additionally, a data pipeline and in-simulation, sample dataset are provided for research posterity. Finally, an outline of open research questions and directions is laid out for future investigation.

Contents

Acknowledgments

God,Nunu,Steven Hailes,Carlo Pincorillo, Andrew Symmington,Family,Friends.Cardi B.

If you look for truth, you *may* find comfort. If you look for comfort, you *will* find despair.
CS Lewis

Chapter 1

Introduction

The emergence and proliferation of drones in the commercial, consumer and military sectors is widely evidenced and supported in literature and industry. This is largely on account of continual advances in hardware miniaturization, cost and robustness coupled with technological leaps in navigational intelligence and communication modalities. While encouraging, notable limitations are continually surfaced in furthering their applicability to unstructured environments. The latter manifests as task and obstacle ambiguity and makes robotic operation challenging at best and intractable at worst with an extensive body of attendant work available in the robotics literature. A novel approach to solving this environment dynamicity problem is presented in the form of Robot Swarms Systems; collectives of locally acting robots that work towards the attainment of set objectives. This capability of Robot Swarms Systems to outperform Monolithic Systems is referred to as *force multiplication* [88] and is a central driver of the field. It is a promising discipline and a notable entry in Yang et als' *Grand challenges of Science Robotics* [88]. Alternative nomenclature for the term Robot Swarms exists in the robotics literature [14] [72] [44]; for exactness we elect to make use of the term Multi Robot Systems in this work as forwarded by [44].

Core to this papers' focus is the application of an Unmanned Aerial Vehicle-based, Multi Robot System (MRS) to the automated maintenance of Green Wall Systems (GWSs) as coined in [66]. The efficiency and capital constraints identified in current GWSs include but are not limited to [42]:

- High initial capital.
- Exorbitant energy costs.
- Extensive environmental control system requirements.
- High Carbon Footprint.

We postulate that the automation of such systems by a MRS will not only mitigate the mentioned costs, but result in much more robust and flexible systems that are amenable to integration in human society as envisioned by Mark Weiser in his aphorism: "*The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it*" [85]. More directly, the selection of GWSs as an application area is informed by the growing academic [57] [36] [59] and industry [11] [42] interest in the economic viability of sustainable urban ecologies; key among these being Vertical Farming Systems. The latter is considered increasingly exigent given the concerning rise in food insecurity [88] in the wake of global population growth trends juxtaposed against finite agricultural and arable land availability [12]. Henceforth, this paper refers to the proposed system under consideration as a Green Wall Multi Robot System (GREW-MRS).

In this work, a GREW-MRS is designed and developed in two conditioned studies to assess the performance benefits conferred in utilising both swarm-inspired optimization and behaviours. The pragmatic benefits of employing such systems in place of individual agent / monolithic systems include [88]:

- *System modularity* - the collective robustness to disturbances and resilience to adversarial disruptions.
- *System flexibility* - the collective responsiveness to human control and ability to adapt to changing conditions.
- *System economy* - the collective cost effectiveness granted by the simplistic robot designs espoused as a fundamental trait of swarm robots.

1.1 Objectives

The main goal of this paper is to statistically highlight the performance effect of Swarm Optimization (SO) and Swarm Behaviours on Multi Robot Systems when applied to the automation of Green Wall System maintenance. This will be compared to a naive Lawn Mower Motion [20] approach that utilises a sweeping movements to perform region filling; inadvertently maximising target space coverage probability at the expense of time to task completion. The evaluation shall be approached as a hypothesis test, with the null H_o hypothesis, (1) and alternate hypothesis, H_a (2) stated thusly:

Hypothesis (H_o): *The mean time-to-threshold in the Lawn strategy is equal to that of the Swarm-inspired strategy.*

Hypothesis (H_a): *The mean time-to-threshold in the Lawn strategy is* ***less than*** *that of the Swarm-inspired strategy.*

To intelligibly do so, a brief ontology of the field of Swarming Systems must first be laid out. This is on account of the esoteric and somewhat exotic nature of the topic at hand, albeit reasonably catered to by key literature [14] [44] [34]. Additionally, an investigation into appropriate simulation platforms and hypothesis testing techniques is elucidated that ensures experimental replicability and validity. The latter are necessary, keystone conditions for reproducible experimental research of which this work aspires to produce.

1.2 Challenges

A number of notable challenges exist, chief among them, a dearth of established research frameworks and methodologies towards the performance analysis of goal-oriented Multi Robot Systems inspired by Swarming Systems. This particularly true for convergence and efficiency evaluation of metaheuristic optimization algorithms present in the Swarm Optimization literature [89]. A causal corollary of which is the arguably highly-extensible nature of the considered systems that allows for a exceedingly diverse swarm configuration space. This makes the task of performing benchmark analysis a tricky affair given all the variants of each implementation that are possible. However, a number of similarly oriented, statistical evaluation approaches to popular Swarm Optimization (SO) algorithms exist in literature [73] [89] with sample benchmark datasets available [71].

A balance between computation cost and real-world fidelity must be met when attempting to perform robotic simulations. To utilize accurate dynamic models and hyper-realistic visualization, the chosen simulator must be run on or make use of high performance computing resources and features. This is a capability hardly achievable with currently available robotic simulators, but one that is receiving some attention in the simulation literature [74]. This additionally places a constraint on the ease of performing comparative studies that require considerably extensive datasets to generate and to analyse. These operations both consume a significant time to store and complete, especially on non-specialised hardware. In the case of this project, the evaluation experiments were executed on a portable Laptop with the following specifications:

- Intel^{R} Core^{TM} Quad-Core i7-4700MQ CPU rated for 2.40GHz
- 12GB of DDR3 Synchronous RAM rated for 1600MHz.
- 1 Terabyte Harddrive rated for 5400 rpm.
- NVIDIA^{R} GeForce^{R} GT 755M with 2GB Graphics Memory.

While the above system is a moderately capable, it nonetheless took 10 hours, 15 minutes and 41 seconds (computed from outputted logfile metadata) to generate the sample dataset provided

with this project. Another notable computation trap was noticed by the author in the storage of the simulation data. It was noticed that while utilising a single CSV formatted file to store the simulation data is simpler in theory to handle, it would likely be impractical for extensive experiments due to the fact that it will lead to the generation of massive dataset files that will prove difficult to load in memory by analysis libraries such as Pandas [4]. The sample dataset provided sits at 4.2 Gigabytes uncompressed and could not be simply loaded into the Jupyter Notebook [5] environment without a conversion into the HDF5 [8] versatile data storage format through a chunking process. Adequate time and consideration must therefore be placed on how one chooses to generate and work with the simulation data.

1.3 Contributions

This work makes the following contributions to the field:

- A novel approach to comparatively evaluate the performance of GREW-MRS strategies and implementations. The application of statistical testing is also introduced with novel measurement variables proposed for wider consideration. Data generation is addressed with appropriately developed tooling provided.
- A novel and simple simulation pipeline with code made freely available. This works' software repository is made freely available and is scripted as an ARGoS experiment that is dynamically configured with the help of a shell script. It can be found at `https://github.com/wndaiga/swarm_ucl` [62].
- A sample dataset of 10 independantly and randomly seeded simulation trials over a range of Green Wall target/plant numbers (2-50).

1.4 Outline

In 2, we provide the historical context and progress of the fields of Swarm Optimization and Robotics research, detailing their genesis in cellular automata and the efforts taken to better define and demarcate the field into interrogative interest areas. A cursory overview of Coverage Path Planning (CPP) and Green Wall Systems (GWSs) is also provided as a contextual backdrop to this work. The precise aim of this chapter is to surface and instill key dogmatic themes of the fields and thereafter walk the reader through prior related work. In 3, we present the experimental design with its' novel measurement variables and the system design implemented in this project, thereafter providing an overview of the selection criteria deemed necessary to perform the statistical

experimental analysis central to this project. The endogeneous and exogeneous sources of innaccuracies are delineated with the cooperative control and path planning strategies laid out in full. In 4, we evaluate our approach using the Student's t-test [50] and identify gaps and oversimplifications where applicable. In 5, a number of possible research directions and areas are suggested for future investigation.

Chapter 2

Background

The GREW-MRS application objective requires the interaction of a myriad of research domains, including but not limited to UAV Stabilization, Optimal Control, Navigation, Obstacle Avoidance, Wireless Communications and Computer Vision [38]. This wide problem scope is hardly solvable in monolithic systems due to the energy and computation constraints present in currently available platforms. Multi Robot Systems are uniquely positioned to robustly and flexibly solve for these constraints through cooperative control.

In this work, agent navigation is formulated as a Travelling Salesman Problem which is easily solvable through a myriad of Swarm Optimization (SO) algorithms that exist in literature. For ease of tractability, we will not consider obstacle nor collision avoidance schemes; however, these are well-researched in the literature [34]. This project will however focus on two SO implementations; the Discrete Particle Swarm Optimization (DPSO) and Ant Colony Optimization (ACO) algorithms. Their selection was on account of their popularity in the literature [80] and simplicity in design. SOs have been shown to be well adapted to solving these types of problems and exhibit beneficial qualities to systems that implement them.

Foundational to the development and advancement of the Swarm Robotics (SR) field is the ability to generate and evaluate high-fidelity, dynamical models in simulation [81]. The importance of such tooling cannot be overstated given the overheads and constraints involved in the testing of costly mobility and transportation platforms whilst enhancing the ease of validating several system aspects. The widespread usage of simulations in the field can largely be attributed to the fact that they are easier to setup, less expensive, normally faster and more convenient to use than physical swarms [60]. Fortunately, this has been bolstered by the availability of advanced and highly extensible multi-robot simulators such as Gazebo, USARSim, ARGoS [68], WeBot and MORSE [30]. Complementary to this is the growth and advancement of three key drivers of robot swarms; the hyper-convergence of hardware and software technologies, novel wireless network-

ing features and strategies and a notable reliance of cognitive systems on Machine Learning and Artifical Intelligence [88]. Of note when considering wireless networking technologies is the incorporation of robust mesh networking specifications [41] that confer practical solutions to encumbured local communication in robot swarms. However, more central to this need for standardised tooling is the fact of the disciplines' pre-paradigmatic stage, defined by Kuhn et al [53] as a nascent period marked by a lack of scientific consensus on appropriate terminologies, methods and experiments. These are necessary for the construction of a scientific framework within which verifiable and replicable research can be performed. This is especially pertinent if prevailing literature on the projected impact of the field is to be realized [88]. With the pioneering work of Beni et al [14], apt consideration is accorded to the terminology of the discipline of Swarm Systems. Therein, the taxonomies of the distinctive qualities of Swarm Optimization (SO) and Swarm Robotics (SR) systems are laid out with further elucidation provided as to the nature of their genesis and applicability. While the foundation laid out by [14] is found to be necessary, it is not entirely sufficient to develop a conceptually-complete framework for scientific investigation. This is cautiously addressed by the works of [44] and [72] that in addendum, reinforce the fact of the disciplines' early stage and lack of common scientific framework. This can be seen to be reflected in the numerous alternate and sometimes conflicting classifications found in existing literature [80]. Wherever possible, and for purposes of clarity and consistency, this project will highlight the choices made in the definition of key terms and ideas.

We maintain that a small-scale system employing probabilistically-modelled agent behaviours is adequately capable of achieving performant system operation. In this work, agent navigation is addressed as a path planning problem solvable with the help of Swarm Optimization (SO). The latter is a growing sub-field of Swarm Intelligence (SI), a field whose industry reports opine widespread commercialization by 2020, driven by the increase in its' usage for solving big data problems, the rising adoption of swarm-based drones in military and the need for SI in transportation and logistics [7]. This provides the SI research community with key market gaps and consequent opportunities towards the development of novel SI applications for varied industries. This project hopes to serve as one such reference study.

2.1 A Brief Overview

In [14], a crucial distinction between Swarm Optimization (SO) and Swarm Robotics (SR) is conveyed where the former is subsummed under Swarm Intelligence (a subfield of Artificial Intelligence) as meta-heuristic applicable to the optimization of objective functions (pattern analysis) and the latter is largely concerned with the coordinated operation of physical agents (pattern synthesis). In the main, they both detail avenues through which intelligent behaviour is achieved by a decentralised, non-synchronous group of quasi-homogeneous, simple units, not in "*Avogadro-*

large" numbers. Here, intelligent behaviour is defined as the production of improbable and unpredictable ordered outcomes.

In dealing with physical agents, a noteworthy benefit conferred by these modularized, mass-produced, interchangeable and possibly disposable robotics systems are reliability guarantees by way of the highly redundant nature of said systems' members. This confers these systems certain performance and robustness gains over monolithic systems [44]. Complementary to this, intelligent behaviour through pattern analysis allows for practical solutions to NP-hard and NP-complete problems such as combinatorial optimization in path planning [86]. They have additionally been applied to a vast cornucopia of areas in design, scheduling and planning, data mining, machine intelligence and many others with an extensive body of work available in the literature [89].

2.1.1 Swarm Optimization

The field of Swarm Optimization (SO) is only just approaching the three decade mark with Gerardo Beni and Jing Wang credited with first coining the term *Swarm Intelligence* (SI) in 1989 [35]. Despite it's rather young history, its' use in solving optimization problems is well covered and encouraged in literature, largely attributable to it's enhanced performance when compared to other optimization methods such as neural networks, machine learning and genetic computation. This has been investigated in the optimization literature, with SO algorithms characterized as exhibiting generally better performance in computing speed, accuracy and memory size [83].

Core to this performance is the tuned attainment of optimal parameters that informs their exploitation and exploration characteristics in solution state space. This tuning reliance presents the two main caveats of SO algorithms; they are normatively prone to premature convergence and are not robust against local minima [23]. However, this is readily corrected for by grafting SO implementations with Generative Iterative Algorithms (GIA) such as Evolutionary Algorithms, Simulated Annealing (SA) and Tabu Search (TS). GIAs are especially suitable for this use case due to their ability to solve ill-posed problems where some parameters are prior unknowns [90] through random population mutations. This ability to hybridise SO algorithms is a growing research area with numerous algorithm variants continually proposed [83] [22] [67]. These extensions, when properly designed and implemented, increase the convergence speed and improve the considered systems [83]. In the main, the following benefits are conferred to systems that implement SO algorithms for pattern analysis [23]:

- *Scalability*.
- *Fault tolerance*.
- *Adaptation*.

- *Speed*.
- *Modularity*.
- *Autonomy*.
- *Parallelism*.

2.1.2 Swarm Robotics

Beni et al [14] made a case for the redefinition of Swarm Intelligence (SI) as applied to robotics systems, noting that the realization of swarm intelligent robots is a distant goal. In doing so, they formalized "*swarm robotics*", "*collective robotics*" and "*distributed autonomous robotic systems*" as more appropriate terms; the selection of which should be independant of group size inherred by their scalability property. This counterintuitively means that application-specific considerations given to the diversity and scale of the developed systems actually makes them less "*swarmy*".

A number of core swarm characteristics are known and established in the swarm robotics literature [18]:

- Robots are autonomous.
- Robots are situated in the environment.
- Robots sensing and communication is limited and local.
- Robots do not have access to centralised control and/or global knowledge.
- Robots cooperate to complete tasks.

It is should however be taken into account that there presently exist different characterizations of Swarm Robotics (SR) in the literature [72] [14] [26] [28]. The most applicable swarm description of the GREW-MRS was found in the works of Ioochi et al [44]. They provided a suitable definition and framework for the characterization of swarm robotics as a *Multi Robot System* (MRS) forwarded as a particular instance of a *Multi Agent System* (MAS). It is concerned with the reactivity and social deliberation of the collective which consist of resource-constrained agents that exhibit limited sensing and communication capabilities. Ioochi's taxonomy is depicted in figure 2.1 and details the varying feature-sets and capabilities presently available to MRS applications.

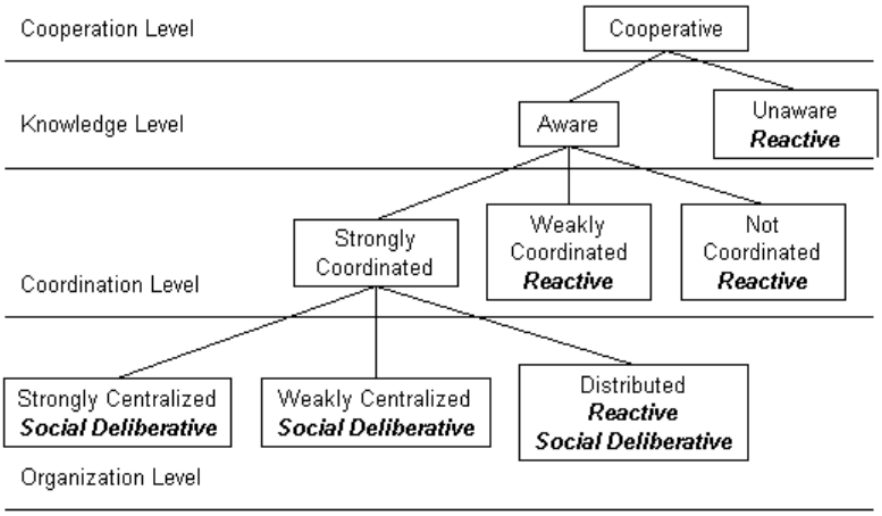


Figure 2.1: A Multi Robot System Taxonomy [44]

In the GREW-MRS proposed in this system, we proposition a Cooperative, Unaware and Strongly Coordinated & Centralized system.

2.1.3 Coverage Path Planning

Point-to-point path planning for mobile robots is a well-studied area of robotics research with a myriad of techniques developed and proposed towards achieving robust, collision-free, goal-oriented navigation in the presence of obstacles and uncertainties. While this is an established and commonplace feature of Monolithic Robots in literature and industry, the same cannot be easily said for Multi Robot Systems (MRS). This is especially so when considering objective-oriented implementations such as Coverage Path Planning (CPP); a topic of major interest in robotics literature [56]. In the case of MRS, this is due to the complexity of the problem that depends on a number of factors [87], some of which are:

- *Robot characteristics*: With MRS, the diversity of the collective is a large determining factor where heterogeneous populations may lead to widely diverging results. As such, homogeneous robot teams are generally preferred as they confer some system reliability.
- *Terrain properties*: A larger workspace would require a larger team size to both efficiently and sufficiently perform required tasks. Additionally, obstacle density and shape can impede the rate at which robot collectives perform work in the target space.
- *Environment Dynamicity*: Changing environments pose a challenge to the maintenance of system guarantees where path planning and obstacle avoidance strategies would increasingly interfere with coordination control.

Given the above limitations, a number of objectives must still be met by the robot collectives. In the case of the GREW-MRS system presented herein, Cao et al [20] defined a number of critera that a robot system must meet for optimal coverage:

- Robot must move through all the points in the target area covering it completely.
- Robot must fill the region without overlapping paths.
- Continuous and sequential operation without any repetition of paths is required.
- Robot must avoid all obstacles.
- Simple motion trajectories (e.g., straight lines or circles) should be used (for simplicity in control).
- An "optimal" path is desired under available condition.

However, Galceran et al [34] critique that it may not always be possible to achieve all six requirements in dynamic environments and posit that a prioritization of criteria is necessary.

Choset et al [21] classified Coverage Path Planning (CPP) algorithms as heuristic or complete depending on their global coverage guarantees. In this work, a semi-heuristic approach is considered given the probabilistic nature accorded to the coordination control implemented at the swarm level. Independantly of this initial classification, CPP algorithms can also be classified as online or offline; in the latter the environment is assumed to be known whereas online approaches do not have access to the same and must rely on real-time sensing and area-sweeping techniques. Offline systems, while simpler to conceptualize and implement in simulation, do not reflect real-world conditions where the environment is known to be dynamic and under the influence of exogenous factors. Online strategies offer more robust solutions to this issue despite being much harder to grapple with from a technical standpoint towards the achievement of globally consistent maps. Online strategies are better understood as methods towards learning environment maps with numerous innovative approaches available in the enhanced and hybridised forms of Filters, Smoothers and Graphs generally subsumed under Simultaneous Localization and Mapping (SLAM) systems. SLAM systems have been a have been a major area of research for over two decades (Merging Partially Consistent Maps) with numerous implementation proposals available [55] [37] [61] [64] [77] [82]. However, the consideration and implementation of online systems was decidedly a secondary focus area of this work and subsequently, a simpler, offline method was utilised. This is also consistent with the literature, it is found only possible to find an optimal solution to the Coverage Path Planning problem for an a priori known, or partially known environment [34] which by definition entail the utilisation of offline systems.

2.1.4 Green Wall Systems

Green Wall Systems, also referred to as Living Walls [76] or Vertical Greening Systems [57] in the literature are approaches to the enhancement and restoration of built-up urban environments with a number of notable benefits being:

- Reduction of greenhouse gas concentrations in urban areas.
- Mitigation of Urban Heat Island (UHI) effect due to the evaporative cooling effect of vegetation.
- Reduced costs associated to Building Environmental Controls (e.g. HVAC systems).

The crucial urgency of addressing the above is made imperative by recent legislative efforts towards combating climate change, with one pivotal example being the *Paris Agreement* of 2015 [84]. However, an identified and somewhat ironic issue with currently available green walls is their seemingly stunted evolution as scalable, sustainability interventions on account of a lack of published comparative studies and research on the same [57] [48].

(a) An Indirect Green Facade [57] (b) A Living Wall System [57]

Figure 2.2: There exists classifications of vertical wall gardens available in literature [57].

Of particular interest to the author is the similarity in the applied scenarios of the presented green wall system and vertical farming. The two applications can be considered equal in most technical respects, with the main difference largely being the consideration of human consumption qualities in the case of vertical farms. Vertical Farming can be defined as an effort towards the development of sustainable, urban-located agriculture and has attracted considerable industry interest in recent years [12]. Recent market projections indicate a market size estimated to amount to over 2.25 Billion USD by 2024 in the United States alone [11].

The viability of automating green wall system maintenance as well as vertical farming tasks lies in the typically high cost overruns associated with the construction and maintenance of these largely closed ecological systems. This is traditionally done with the help of human labour, with the extenuating risk that these tasks may expose them to (e.g. the scaling of exceedingly tall installations). Additionally, the manual and semi-manual piloting of UAVs can be long and tedious for operators especially when the presence of wind gusts and turbulence is endemic in urban areas; a consequence of the Urban Canyon Effect. The use of a GREW-MRS to achieve automation of the same would drastically reduce this encumberance and constitutes a novel approach towards advancing the economic practicality [15] of aerial robots in daily human life beyond the current focus on consumer curiosities and entertainment light shows [10].

2.2 Related Work

While the determination of standardised metrics to evaluate Swarm and Multi Robot System performance vary from application to application, a number of definitions stand out in the robotics

Figure 2.3: Project *Via Verde*: Green Wall Systems deployed in Mexico to combat urban pollution [70]

literature that beg mention. In the case of Coverage Path Planning, terms such as path length and time to completion, are present in the literature [34] when considering the optimality of generated solutions.

Cuevas et al [23] developed the Social Spider Optimization (SSO) that they comparatively evaluated against the Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) algorithms. They utilised a similarly utilised hypothesis testing framework to compare the mean values of each sampled algorithm. In it, they forwarded the Average-Best-So-Far (AB), Median-Best-So-Far (MB) and Standard-Deviation-Best-So-Far (SD) solutions as dependant measurements, testing for a 5% significance level finding strong evidence against the null hypothesis that assumed no significant difference between the measured sample variables.

Although it is difficult to surface similarly applied research in the literature, there are works that exhibit parallels to this project. Brien et al [65] helped define the field of formal verification systems as applied to autonomous robots. They developed a verification software module to assess the probability that inputted performance criteria of autonomous robots can be met despite the uncertainty of real-world conditions. However this was limited to monolithic systems with added work necessary to extend their module to MRS implementations. Selvi et al [73] performed a comparative analysis of the Particle Swarm Optimization and Ant Colony Optimization algorithms but did not forward a basis of statistical measurment. Phung et al [67] developed an enhanced Travelling Salesman Problem (TSP) tractably solved by a discretized PSO algorithm as applied to UAV-based, Structural Health Monitoring (SHM). They formalised Coverage Path Planning (CPP) as a multi-level, Inspection Path Planning (IPP) problem. Their approach alluded to a bipartitioning of the same into an initial Art Gallery Problem, and thereafter optimizing the given solution as a Travelling Salesman Problem. Overlapping photos of the target surfaces are taken once the IPP objective is met for later processing. Guerrero et al [38] similarly presented

work on the use of Quadcopters towards the structural inspection of bridges, notably implementing the Path Planning as a Travelling Salesman Problem (TSP). DiFranco et al [24] proposed an energy-aware Coverage Path Planning (CPP) algorithm that minimizes the energy consumption of multi-rotor UAVs while achieving multi-objective surveying of a target space.

Fredslund et al [33] investigated the use of multiple leaders in maintaining formations in Swarm Systems through local sensing and minimal communication i.e. without the sharing of global information with local agents. Matari et al [58] utilised simple behaviours towards the development of a global flocking behaviour in a group of 13 mobile robots. These behaviours included Safe-Wandering, Following, Dispersion, Aggregation and Homing. Bircher et al [15] designed and developed a fast algorithm for the efficient inspection path planning of varied structures based and implemented on monolithic UAV systems. They advanced a fast implementation of the Lin-Kernighan-Helsgaun Heuristic (LKH) TSP solver. Optimization of such algorithms is a key area of work in this field due to the limited computing and energy resources available on current aerial quadcopter systems and a valuable research direction for the field.

A major distinction difference between this project and that of [58] and [33] is the domain space of the mobile robots. The latter works were implemented on terrestrial-based mobile robots whereas this project considers an application to UAV-based robots. As a matter of observation, the author found there to exist a much larger body of work related to the Swarm and Multi Robot System analysis of terrestrial-based robots. This can be said to be attributable to their rather simpler 2D domain space as compared to UAV-based systems' 3D domain.

Chapter 3

Implementation

In this work, the development of a GREW-MRS towards automated Green Wall System maintenance is presented in simulation. There exists a wide and diverse problem-set associated with the stated objective that is covered in the corpus of robotics literature. We forward the usefulness of Swarm Optimization (SO) techniques to solving Coverage Path Planning (CPP) formulated as a Travelling Salesman Problem (TSP). More specifically, we implement the Particle Swarm Optimization (PSO) [50] and Ant Colony Optimization (ACO) [27] metaheuristic optimization algorithms. Coordination Control of each MRS agent is probabilistically controlled through virtual stigmergy and a strongly centralised hierarchy. Additionally, Cao et al's [20] CPP optimality criteria was adapted through a consideration of a number of limitations as suggested by [34]:

- The utilised SO algorithm implementations are 2D-based and therefore we cannot guarantee that there will be no overlapping paths.
- A lack of motion planning features in the ARGoS simulator meant that obstacle avoidance while necessary, would need to be implemented either from the ground up or through planners such as OMPL [91]. Additionally, the referenced lawnmower problem is notable in that it is not designed to account for obstacles [34].
- The use of probabilistic behaviours for cooperation control means that complete target area coverage cannot be guaranteed.

In this work, a semi-heuristic, offline approach to solving the CPP problem is implemented. In the CPP literature, this is typically solved by cellular decomposition techniques where the target space is broken into sub-regions [21]. The MRS developed herein however have access to a global map that contains the position coordinates of all target locations. The latter are then passed in to the TSP solver as city nodes and an optimal path generated thereafter. This solver makes

use of the Discretized Particle Swarm Optimization (DPSO) or Ant Colony Optimization (ACO) algorithms with two key benefits being their relative speed and robustness to combination explosion, an encumbrance to traditional algorithms such as the Cupity Algorithm and the Dynamic Programming Algorithm. However, these benefits come at the cost of optimal global solution guarantees as previously mentioned [86].

In this papers' case, the DPSO and ACO were selected on account of their prolific prominence in field [73]. The implementations utilised herein were based off of C++ codebases of the same openly available online [43] [19], both found to have followed the relevant literature on PSO [50] and ACO [25] algorithms.

3.1 Experimental Design

A number of guiding principles exist in the experimental design literature and are known to be necessary to planning research experiments [32]:

- Empirical.
- Measurement.
- Replicability.
- Objectivity.

3.1.1 Empirical

To ensure empiricity, the experimental design was formalized as detailed by Field et al [32] with a research question developed and posited as follows; "*Do Swarm Intelligence and Behaviours and systems affect the task performance of multi-agent quadcopter systems in Green Wall Systems*?". The analysis of this affect was undertaken by performing a hypothesis test that further refined the research question into null 1 and alternate 2 hypotheses previously introduced. To perform this analysis, simulator-based experiments were developed based on a designed and consequently proposed GREW-MRS. In order to infer causality, we must set up two scenarios where the independant variable is present and one where it is absent. These are referred to as the experiment and control conditions respectively. Added care must be taken to ensure they both conditions are identical in all senses except for the presence of the cause. This reduces the possibility of *the third variable problem* or *the tertrium quid* where an unidentified, confounding variable effects some unanticipated change in the dependant variable [32]. These latter measures of scenario similarity are covered in 3.2. In formulating our experiment thusly, the empirical trait requirement of research experiments is met.

In the main, a leadership-based, robot organisation strategy was employed which has been shown [29] to be an effective method in navigational guidance both in the absence and presence of conflicting information. The distinct differences between the control and experimental conditions are listed below:

- Navigation path planning.
- Local communication.
- Global behaviour emergence.
- Use of local information.

These properties are derived from known swarm robotics characteristics [26]. In navigation path planning, we have centralised and decentralised modes.

Control Condition

In the control condition, a naive lawn/seed-spreader path planner [34] is implemented along which tasked eyebots can travel and probabilistically act on locally sensed identified targets. The algorithm below was implemented and deployed to each agent in the simulated GREW-MRS. This is shown in Algorithm 1.

Algorithm 1 Control Condition Algorithm

procedure EXPERIMENTSIMULATION ▷ Simulation Entry Point
  while $termination\_criteria\_not\_satisfied$ **do**
    INITIALIZE;
    COORDINATIONSTRATEGY;

procedure INITIALIZE
  if $System\_Initialized \neq TRUE$ **then**
    $Map_{global} \leftarrow Ordered\_Lawn\_Path\_Waypoints()$; ▷ No Path Optimization
    $Waypoint_{current} \leftarrow 0$;
    $System\_Initialized = TRUE$

procedure COORDINATIONSTRATEGY
  $Map_{local} = Map_{global}$;
  if $Probability_{move} \geq Random_{move}$ **then**
    Move to $Waypoint_{current}$;
    if $Target_{nearest} \in Map_{local}$ and is unattended **then**
      if $Probability_{completion} = 1$ **then**
        Update Map_{global}; ▷ Update Target Metadata
        $Waypoint_{current} \leftarrow Waypoint_{current} + 1$;
      else
        Hold;
    else if $Target_{nearest}$ is attended **then**
      $Probability_{land} \leftarrow Probability_{land} + Probability_{land\_increment}$;
      $Probability_{move} \leftarrow Probability_{move} - Probability_{move\_increment}$;
  else if $Probability_{land} \geq Random_{land}$ **then**
    Land;
  else
    Rest;

Experimental Condition

In the experimental condition, an online Swarm Optimization (SO) algorithm (PSO or ACO) is implemented to generate shortest TSP path solutions of locally communicated target maps. A global map is initialized and made available to each eyebot at start up time to ensure consistent map information is shared between all agents in the collective. Additionally, a probabilistic framework is implemented to control the rest to move and rest to land state transitions of that is informed by the local sensing of targets and local communication in each agents neighbourhood. This is shown in Algorithm 2.

Algorithm 2 Experimental Condition Algorithm

procedure EXPERIMENTSIMULATION ▷ Simulation Entry Point
  while $termination\_criteria\_not\_satisfied$ **do**
    INITIALIZE;
    SOCIALRULE;
    COORDINATIONSTRATEGY;

procedure INITIALIZE
  if $System\_Initialized \neq TRUE$ **then**
    $Map_{global} \leftarrow Path\_Optimized\_Targets()$; ▷ Metaheuristic Optimization
    $Waypoint_{current} \leftarrow 0$;
    $System\_Initialized = TRUE$

procedure SOCIALRULE
  if $message\_recieved \neq NULL$ **then**
    if $message\_contains\_attended\_task$ **then**
      $Probability_{land} \leftarrow Probability_{land} + Probability_{land\_increment}$;
      $Probability_{move} \leftarrow Probability_{move} - Probability_{move\_increment}$;
    else if $message\_contains\_assigned\_task$ **then**
      $Probability_{land} \leftarrow Probability_{land} - Probability_{land\_increment}$;
      $Probability_{move} \leftarrow Probability_{move} + Probability_{move\_increment}$;

procedure COORDINATIONSTRATEGY
  if $Probability_{move} \geq Random_{move}$ **then**
    $Map_{local} \leftarrow Path\_Optimized\_Assigned\_Targets()$; ▷ Metaheuristic Optimization
    Move to $Waypoint_{current}$;
    if $Target_{nearest} \in Map_{local}$ and is unattended **then**
      if $Probability_{completion} = 1$ **then**
        Update Map_{global}; ▷ Update Target Metadata
        $Waypoint_{current} \leftarrow Waypoint_{current} + 1$;
      else
        Hold;
    else if $Target_{nearest}$ is attended **then**
      $Probability_{land} \leftarrow Probability_{land} + Probability_{land\_increment}$;
      $Probability_{move} \leftarrow Probability_{move} - Probability_{move\_increment}$;
  else if $Probability_{land} \geq Random_{land}$ **then**
    Land;
  else
    Rest;

3.1.2 Measurement

This paper advances an approach towards the implementation, analysis and evaluation of coverage path planning and coordination control in multi robot systems for Green Wall System maintenance. This analysis and evaluation requires the proposition of validly independant and dependant variables that can be associated to the objective tasks' performance. Statistical samples for both the control and experiment conditions can then be generated. The indepedant/causal variable was characterised as the *usage or non-usage of Swarm Intelligence and Behaviours*. The dependant/outcome variable was selected to be *the simulation-time taken to achieve 90% target completion coverage*.

Given these variable characterisations, the classification of the independant variable can be seen to be a nominal, two-level measure whereas the dependant variable is a interval measure. Factorial validity of this measure was found to make intuitive sense [32] while it's reliability was ensured by performing multiple trial sampling. Given these prescribed variable types, the student's t test was utilised to compare the means of the two samples [46]. The parametric characteristic was preferable on account of the following [32]:

- There exist a greater variety of tests available that would allow for a wider experimentation base.
- Parametric tests are generally better at identifying experimental effects.

The generation of sample data qualifies the measurement requirement trait of this experiment.

3.1.3 Replicability

Replicability was assured by providing a standardized, script-based method of generating new datasets which is elaborated in 3.2. This is especially pertinent in generating large simulation datasets where parameter setting can be prone to human error. The developed script-based approach provides a single entry point for trial data generation ensuring all necessary parameters are explicitly set, utilised and stored, thereby guarding the mentioned error. Additionally, we provide open access to the dataset generated and analysed in this work to ensure that the parameter-matched experiments can be run and re-analysed by fellow researchers in the field.

3.1.4 Objectivity

The notion of scientific objectivity, while invaluable to the field, is hard to empirically prove as it is multivariate in nature. It assumes that a truth exists independantly of inspection or observation which in and of itself has been a tenet of philosophical discourse for millenia. It can however be espoused by researchers in their impartiality to the experiments' outcome.

3.2 System Design

The system as presented constitutes a simulation pipeline consisting of the following main sub-components:

- Trial Simulation.
- Data Collection, Pre-processing and Analysis.

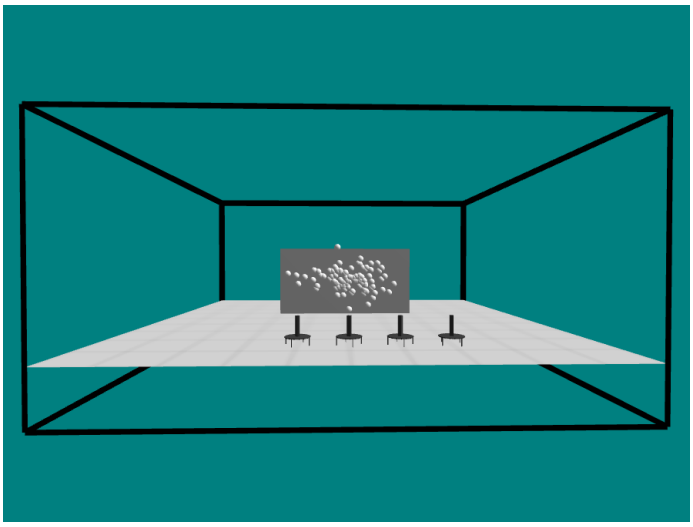
3.2.1 Trial Simulation

There is a recognised lack of reliable and robust swarm-capable simulators [63]. Robotics Development Environments (RDEs) have come to play a significant role in robotics research. Despite a lacking availability of systematic RDE evaluation in the literature [52], this work will summarily present comparison criteria key to the viable selection of an autonomous robot simulation environment. In the case of this project, the key RDE characteristics, as defined in [52], were identified to be *Open-source codebase*, *High-fidelity*, *Scalability*, *Multi-Robot Capable*, *Hardware Support* and *3D Capable*. We include an additional criteria; *Active Development*. This is on account of the rather large churn of released RDEs, a key example being Miro [31] whose source could not be found online. The proposed RDEs in [52] are suitable for an initial selection pool with a number of new releases included that meet the given seletion criteria defined previously. The qualifying RDEs were found to be:

- Gazebo [51].
- MissionLab [3].
- ARGoS [69].
- MORSE [30].

For this study, trial simulation was performed using the ARGoS simulator, chosen on account of it's ability to simulate large numbers (in the thousands) of swarms robots efficiently and flexibly [68]. The author of the simulator, Prof. Carlo Pinciroli, was found to be actively available on both the development forum page and responded to integration queries as and when needed. In setting up the simulation, the Green Wall System scene environment was purposefully designed to typify a simplified, real-world scenario as shown in Figure 3.1a.

The swarminoids' [26] eye-bot quadcopter was selected as this studys' simulated, aerial platform due to its' established usage in heterogeneous swarm robotics research. By design, they are autonomous flying robots which can attach to an indoor ceiling, and are capable of analysing the environment from a privileged position to collectively gather information inaccessible to terrestrial-based robots [26]. It is shown in 3.1b and features the following sensing capabilities:



(a) Simulated Green Wall System Scene
The simulated scene run in both control and experiment conditions. The vertical wall is laden with circular (plant) targets with local agents embodied by eye-bot quadcopters.

(b) Eyebot Quadcopter
Image of a real-world Eyebot, designed and developed as a robotic agent in the Swarminoid Project [26].

Figure 3.1: Simulator Entities

- Custom 360 deg pan-tilt camera system, equipped with a 3MP camera.
- Optical 360 deg infrared environment distance scanner.
- Advanced 3D relative positioning sensor for swarm coordination/communication.
- Custom 6-Degree of freedom inertial sensing.
- Sonar and differential pressure sensors for altitude determination.
- Magnetometer for heading determination.
- Horizontal RGB led rings (local visual communication).

To perform trial simulation, a number of implementation details were essential to developing a suitable, end-to-end simulation pipeline. These include:

- Uncertainties.
- Cooperative Control.
- Coverage Path Planning.

Uncertainties

Assured robot simulation realism is a growing focus area [81] in the dynamic simulator literature with notably recent and advanced examples being AirSim [74] and Morse [30] [54]. To close this simulation-realism gap in ARGoS, a number of independantly seeded random generators were required to ensure systematic endogeneous and exogeneous noise and disturbance sources could be appropriately modelled while still ensuring control and experiment similarity in all extraneous senses. The following enumerate all the modelled sources as implemented:

- Positioning Noise - Positional noise was modelled as a Gaussian distribution and is statically seeded through all trials. A Kalman Filter is utilised for state prediction whose implementation was based off an openly licensed and available C++ codebase [40].
- Mapping Noise - Mapping noise was modelled as a Gaussian distribution and is statically seeded through all trials. This applies to generation of the global map targets. It is assumed that these target locations are obtained from a previous target mapping stage.
- Task Completion - Task Completion Probability was modelled as a Uniform Integer Distribution and is statically seeded through all trials. Task completion is an independant uncertainty whose current probability does not depend on past or future predictions.

- Target Classification - Target Classification Probability was modelled as a Uniform Integer Distribution and is statically seeded through all trials. This is a random shuffle stream that is used in the classification of each target found the simulated Green Wall System. This is utilised by the leader agent that performs all target evaluation.
- Target Placement - Target Placement was modelled as a Gaussian Distribution and is the only dynamically seeded generator over each trial. This is done to better simulate the fact that for any given number of plant targets on the Green Wall System, a varied number of positional configurations exist.

Cooperative Control

In [56], three methods are identified towards the generation and maintainance of shape formations in autonomous, vehicle-based mobile robots:

- Virtual Structure.
- Behaviour-based.
- Leader-following

Through both the control and experimental conditions, a leader-follower strategy, where a target evaluation eyebot agent operated in advance of the slave agents to ensure efficient target classification and task actioning thereafter. In the case of the seed-spreader strategy, agents are strictly concerned with maintaining a virtual leader-based hierarchy to ensure optimal inspection/maintainance coverage. The social-behaviour based strategy is only implemented in the experimental condition as qualified feature of Swarm Robotics (SR). This entailed the sharing of information between the robot collective with each packet sent containing the target id of the current plant target with its' associated task id. Neighbouring agents parse the message and either:

- Store the target in its' global map as a task to complete. The agents' moving and landing probabilities are increased and decreased respectively.
- Ignore the target if it does not match the agents' task assigned id. The agent's moving and landing probabilities are maintained in this scenario.

Swarm robotics systems are characterised by decentralised control, limited communication between robots, use of local information and emergence of global behaviour [26]. An example of this emergence is brought to the fore in simulation where the following was observed:

- Agents in the experimental condition are seen to wait until a number of tasked targets are assigned. This wait time is controlled by the probability to move increment value.

- Agents in the MRS can move out of the land state to *verify* the state of assigned tasks in an effort to maximise its' landing probability.

The ability to inspect and tweak the experimental condition is seen as a valuable feature that enables for a wider emergent behaviour experimentation base. In tandem, this requires a keener understanding of the underlying parameters, possibly necessitating expert domain knowledge to optimally assign values in a problem dependant manner. Bayandir et al [13] provide a review of swarm robotic tasks and viable techniques used to achieve them. To design the organisation systems necessary for guaranteed MRS operation, the task space was delimited to 4 distinct domains:

- Evaluation Task.
- Water Task.
- Nourish Task.
- Treatment Task.

Due to the relative simplicity of the tasks and social rules implemented in this project, a minimal functionality layer was required. This involved the utilisation of *shared memory* between all locally acting agents and a stigmergy-based approach to validating target status. The latter is an simulation assumption enabled by the real-world use of RFID tag markers. This presents a simplistic avenue to guarantee target completion.

Coverage Path Planning

In our work, CPP is achieved by two methods:

- In the lawn mover motion, a waypoint decomposition of the target space is performed by iteratively generating coordinate points that ascribe to the defined motion. Here, a number of parameters must be set to control the increment steps necessary in the lateral and longitudinal directions.
- In the behaviour-based coordination control, a swarm-inspired neighbour-listening approach is used to generate local task waypoint maps for each slave agent. The leader agent is charged with evaluating each plant target as listed in it's global map.

The CPP problem can be suitably mapped onto a connected graph, $G = (N, E)$ where each node $n \in N$ is a target plant position. This connected graph is conveniently solvable by meta-heuristic algorithms tuned to find the shortest Hamiltonian path between a pair of nodes.

Discrete Particle Swarm Optimization The Particle Swarm Optimization (PSO) metaheuristic algorithm, introduced in 1995 by Kennedy et al [50]. The original formulation was devised for the optimization of continuous non-linear functions, whereas this project required the use of a discretized solver. Whereas the Ant Colony Optimization algorithm was inspired by the social structure of ants, the PSO algorithm is more general in its' assumption of societal paradigms. However, core to it's motivating hypothesis was the attempt to model human social behaviour [50]. Phung et al [67] provide a reference pseudocode implementation from which the implemented C++ PSO optimizer was derived.

Ant Colony Optimization The Ant Colony Optimization (ACO) metaheuristic algorithm, was introduced in 1999 by Dorigo et al [25]. They enhance the Simple Ant Colony Optimization (S-ACO) algorithm that is limited to generating solutions to shortest path problems without the ability to apply additional constraints. This is done by the encoding of the whole ant search process as pheromone trails along each graph connection/edge. These pheromone trails are subsummed under a stochastic local decision policy; in the literature, this policy can be modified and adapted for different convergence and solution characteristics. This includes pheromone trail evaporation rates and daemon actions [25]. Dorigo et al [25] provide a reference pseudocode implementation from which the implemented C++ ACO optimizer was derived.

Lawn Sweeping Motion Cao et al [20] introduced a region filling strategy for sweeping operations in a robot lawn mower (RLM), thereby setting the stage for advanced studies into coverage path planning (CPP) strategies [34]. In this work, we implement a similar region filling strategy, with one caveat being the lack of obstacle avoidance and replanning. In the implemented C++ Lawn Generation algorithm, an iterative waypoint generation scheme was developed whose horizontal and vertical step parameters dictated the waypoint density of the outputted map. This worked adequately well for our purposes, though it was found that a parameter optimization step would be necessary to tune the waypoint density to the required target number and spread. Alternatively, performing a cellular decomposition of the target space as informed by the positions of the concerned targets would result in an optimal lawn path generation scheme; better amenable to a performance comparison against swarm-inspired approaches as presented here.

3.2.2 Data Collection, Pre-Processing and Analysis

The ability to generate and analyse simulation data is cardinal to the evaluation of the works presented in this project. It necessitates the development of a data pipeline that involves the collection, pre-processing and analysis of said data. To do so in an efficiently automated manner, a dynamic shell script was developed to manage both the collection and management of simulation data. This script can be found in this project's shared codebase repository [62] and is named *run.sh*. This script was tested and run on an Ubuntu 16.04 System as well as Window Subsystem for Linux with a number of cross-platform bugs quashed to ensure feature parity. A number of bash script flags are made available to the user to better configure multiple trial runs. These are shown in code snippet 3.1:

```
./run.sh usage:
        -a  Select path planning algorithm/strategy (pso, aco or lawn).
        -b  Build the main argos project. Use after editing source files.
        -d  Set the number of drones to place in simulation.
        -e  Set experiment source file. Currently defaults to "main".
        I)  Create experiment environment and install package dependancies.
        -j  Run the jupyter environment.
        -n  Set number of targets/plants to place in simulation.
        N)  Set value range of targets/plants to place in simulation.
        -s  Set the number of independantly seeded trials to run.
        -t  Set the target coverage/inspection percentage during trial.
        -v  Enable argos vizualization. Disabled by default for speed.
        h | *)  Print this usage info.
```

Source Code 3.1: Script Entrypoint Options

A simple user manual is provided in the linked code repository [62]. The script should first setup the project environment using the *I* flag. An example of a full suite simulation is shown in code snippet 3.2.

```
sudo ./run.sh -t "0.90" -a "pso aco lawn" -s "10" -n "2 50"
```

Source Code 3.2: Full-Suite Simulation Command

A number of simulation CPP Parameters were set as follows and kept constant for all experiments as shown in table 3.1. For our purposes, these statically set parameters were determined

by experimentation and lead to guaranteed target completions thresholds across a wide range of target numbers in simulation. The PSO and ACO parameters...

Planner	Parameter Name	Parameter Value
PSO	Self Trust	0.2
	Past Trust	0.1
	Global Trust	0.7
ACO	Number of Ants	10
LAWN	Launch Step	500
	Horizontal Step	0.1
	Vertical Step	0.1

Table 3.1: Simulation CPP Strategy Parameters

Code snippet 3.2 is in fact the very command used to generate the sample dataset provided by this work. The passed flags direct the script to set the target completion threshold at 90%, across all CPP strategies (pso, aco and lawn) with 10 independantly seeded target trials for a range of 2 to 50 targets in simulation. The script as currently implemented generates a simulation dataset in a CSV formatted text file with CPU and simulation statistics saved to profile logs files. The simulator variables stored are shown in table A.1.

In data analysis, preparing the data for efficient processing is a necessary step. This was found to be especially true for this project given the large memory footprints of the generated datasets. For comparison, the sample dataset has uncompressed memory footprint of 4.2 GigaBytes. For ease of data sharing, this was compressed using standard Linux compression tools. Pre-processing was completed with the help of Pandas [4], a Python data analysis library. This involved splicing the dataset into it's constitient considered strategies (pso, aco and lawn) and making sure to optimize datatypes to their expected range values before storing the resultant subsets as HDF5 dataFrames. This has a significant effect on the storage and loading performance of the dataset resulting in a memory footprint of 2.1 GigaBytes.

The analysis of the dataset was performed in a Jupyter Notebook environment [5], a scientific computing tool that has received widespread use and mention in industry and research [75] [1]. The codebase sets up an isolated python environment from which it can run the sample Jupyter Notebook *SwarmAnalysis.ipynb*. It can be easily activated by running the command *./run.sh -j* in the projects' root directory over the command line.

Chapter 4

Evaluation

In Swarm Robotics (SR) research, it is difficult (if not impossible) to directly observe, examine and understand all the collectives' properties and consequent behaviours. Given the latter problem, we resort to collecting random samples from the control and experimental conditions with which we can perform some level of analysis. Hypothesis testing is one such level of analysis that is used to evaluate two mutually exclusive statements about populations to determine which statement is best supported by sample data. These two statements are known as the null hypothesis (3) and the alternative hypothesis (4) with our indepedant/causal variable, a nominal, two-level measure characterised as the *usage or non-usage of Swarm Intelligence and Behaviours*, defining our control and experimental conditions. SciPy [47], a python scientific computing library was utilised that features a vast number of statistical utilities that are essential to modern, data-driven research, one of which is a hypothesis testing api and [16] used to better chart the steps taken to perform a hypothesis test.

Hypothesis (H_o): *The mean time-to-threshold in the Lawn strategy is equal to that of the Swarm-inspired strategy. The mean time-to-threshold in the Lawn strategy is* ***not*** *equal to that of the Swarm-inspired strategy.*

Hypothesis (H_a): *We can consider a significant difference between the mean values of the two strategies.*

We must firstly determine the specific hypothesis test that would be valid given our dependant/outcome variable, an interval measure defined as *the simulation-time taken to achieve 90% target completion coverage*. To do so, certain data requirements must be met to properly determine which analysis can be performed. As initially designed, the dependant variable is continuous. However upon a second examination of the data retrieved from our simulation, a notable discrepancy was identified with missing entries found in the dataset. Further investigation revealed a

high inflexibililty to environment dynamicity, manifesting as very large changes or hard limit contraventions in our dependant variable as a consequence of varying target position configurations.

A consequence of this is that while each strategy trial was designed to be independantly seeded for multiple iterations and the dependant variable averaged across all iterations, we could not guarantee similar sample sizes across all trials. This maintainance of mutually consistent estimates is a well known problem in the inferential statistics literature, typically solved by weighting procedures [49] [17] such as repeated weighting [39]. However, additional data modelling steps are required before a valid weighting procedure can be settled upon. A naive approach that involves fixing the target position configuration across simulation trials cannot be guaranteed to produce successful runs in both the control and experimental conditions as well as not accounting for the need for randomly collected samples. Conversely, this project implemented a more valid data integrity check during simulation runtime that re-seeded a new trial iteration when unsuccessful runs occur. Unsuccessful runs were a result of a logical data logging flaw in which the generation of premature target threshold positives led to invalid logging events. The addressing fix implemented hard limits on the logging event manager, set to terminate a running trial when all eye-bots are in the landed state and the simulator has run for an arbitrarily large total of $200,000$ ticks, settled upon by experimentation. This follows from the statistical analysis literature wherein confounding variable effects must be mitigated with sample data randomly collected, conditions which this project reasonably meets. Here, the confounding variable is the variability in the configuration of target positions.

Ultimately, this nuanced dataset validation process was found to be instrumental to our analysis whilst maintaining the random nature of the sampling process required for statistical validity. Having addressed the data domain problem, the next step involved identifying which hypothesis test is applicable to our cleaned data with a hierarchy of common hypothesis tests is shown in figure 4.1.

Parametric and non-parametric tests make certain assumptions about the sample distributions with the former assuming data normality and homoscedasticity while the latter assumes the opposite. In cases where data normality cannot be guaranteed, parametric tests such as t-tests have been shown to be robust to departures from normality given sufficiently large sample sizes. This affords us some leeway in test selection, but only just so. Additionally, our dependant variable can be seen to be discrete in nature i.e. number of simulation steps. More precisely, it is of interval, count datatype meaning it can only take on non-negative integer values where these numbers arise from counting rather than ranking which calls for a parametric test. In our case, and as recommended in [9], the dependant variable can however be treated as continuous as it is never near zero and ranges over very large numbers. Thusly, we can reasonably presume to use the two-sample t-test as a valid test. Of note are a number of guidelines that exist in the literature for cases of non-normal data [6]:

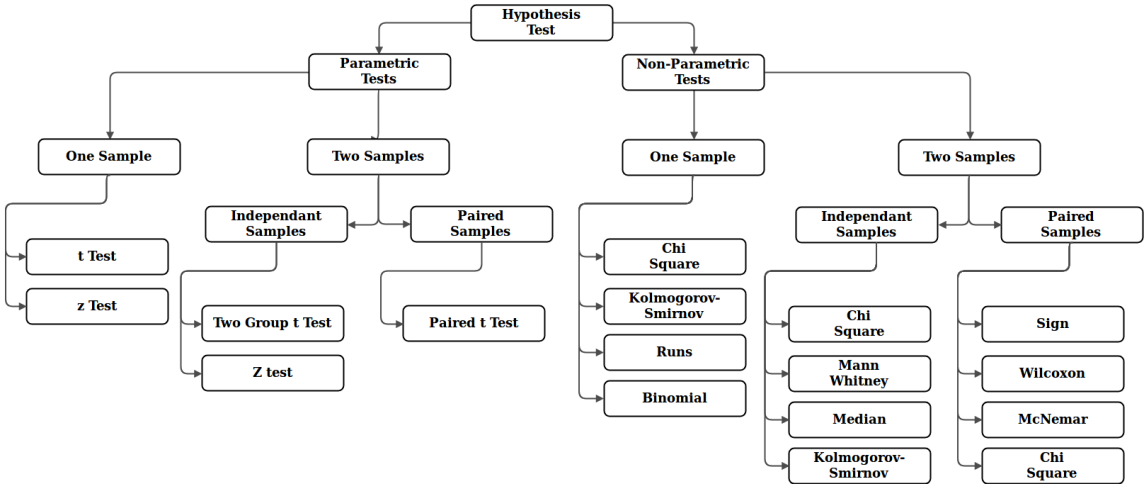


Figure 4.1: Hypothesis Tests

- Two-sample test - Each group should have more than 15 observations.
- One-sample test - Each group should have more than 20 observations.
- One-way ANOVA - For 2-9 groups, each group should have more than 15 observations. For 10-12 groups, each group should have more than 20 observations.

This project presents a sample size of $N = 17$ observations, well within the recommended sample size bounds previously highlighted. There are a number of important considerations that must be taken into account when choosing the sample size which is known to have an error effect. There are two general types of errors that lead to false conclusions about our populations:

- Type I error - known as a false positive; occurs when our test rejects a null hypothesis that is true. The Type I error rate equals our significance level/alpha (α).
- Type II error - known as a false negative; occurs when our test fails to reject a null hypothesis that is false. The Type II error rate is commonly referred to as the beta (β) value.

The statistical power of a test is the probability that a hypothesis test correctly rejects a null hypothesis and is computed as $1 - \beta$. Generally, small sample sizes, noisy data or small effect sizes tend to reduce the statistical power of tests. As such, we may want to increase the sample size in future studies, but in the literature a *power analysis* is typically used to better inform sample size selection as there is a balance that must be maintained. Increasing the sample size enhances the tests' ability to infer small effects, but this comes at a cost; in our case this cost is largely computation time; it takes a long time to generate large datasets. There also exists a point of diminishing

returns where the effect size ceases to grow at which point it looses practical significance. In the main, we would want to collect a sample that is large enough to have sufficient statistical power to detect *meaningful* effect while not being too large to be overly costly/expensive.

Additionally, the sample error, defined as the difference between the sample statistic and the population value, is a metric that must be factored in when deciding when to reject, or fail to reject a null hypothesis. This is because differences observed in the extracted samples might be attributable to the sample error and not a true effect of the population. This is exhibited in the failure to replicate experiment results, a core tenet of good research.

4.1 Dataset Generation

The run script (`run.sh`) in the source codes' root directory is the main entrypoint for this project. It has a number of options that can be dynamically set for a wide range of simulation scenarios. This are available for reference by passing the help flag as a parameter.

```
./run.sh -h
```

```
./run.sh usage:
        -a  Select path planning algorithm/strategy (pso, aco or lawn).
        -b  Build the main argos project. Use after editing source files.
        -d  Set the number of drones to place in simulation.
        -e  Set experiment source file. Default: "main".
        I)  Create experiment environment and install package dependancies.
        -j  Run the jupyter environment.
        -m  Set hard-limit for simulation runtime. Default: 20,000
        -n  Set number of targets/plants to place in simulation.
        N)  Set value range of targets/plants to place in simulation.
        -s  Set the number of independantly seeded trials to run.
        -t  Set the target coverage/inspection percentage during trial.
        -v  Enable argos vizualization. Disabled by default for speed.
        h | *)  Print this usage info.
```

The sample dataset was created with the following settings: - Target Coverage Percentage of 90% - Across all three path planning algorithms/strategies. - Independantly seeding 5 random child trials per parent iteration. - Across a range of 1 to 19 targets.

```
./run.sh -t "0.90" -a "pso aco lawn" -s "5" -N "2 19"
```

4.2 Pre-process Dataset

```
# Import the necessary packages.
import pandas as pd
from scipy import stats
from math import sqrt
from scipy.stats import ttest_ind
from scipy.stats import t
from scipy.stats import levene
import numpy as np
import matplotlib as plt
from math import sqrt
import statsmodels.api as sm
```

We first convert the csv dataset into hdf5 format for data storage and loading efficiency, and compute our sample properties thereafter.

```
chunksize = 10 ** 4
filename = 'output/data_0.csv'
headers = ['Type','TargetNum','TargetThresh','SimStep','Completed','MinimumHold',
        'LaunchStep','InitialRtMProb','RtMDelta','InitialRtLProb','RtLDelta','MinimumRest',
        'InitialMinimumHold','MaximumHold','GlobalReach','ProximityThresh','Attitude',
        'SwarmParticles','SwarmSelfTrust','SwarmPastTrust','SwarmGlobalTrust','SwarmAnts',
        'MappingMean','MappingStdDev','MappingSeed','RtMMin','RtMMax','RtMSeed',
        'RtLMin','RtLMax','RtLSeed','ACOSeed','TaskCompletedMin','TaskCompletedMax',
        'TaskCompletedSeed','TargetShuffleMin','TargetShuffleMax','TargetShuffleSeed',
        'NaiveMapping','VStep','HStep','SimStepMax','SimTrialNum','ArgosSeed']
datatypes = {
'Type':np.string_,'TargetNum':np.uint8,'TargetThresh':np.uint8,'SimStep':np.uint32,
        'Completed':np.uint32,'MinimumHold':np.uint8,'LaunchStep':np.uint8,
        'InitialRtMProb':np.float16,'RtMDelta':np.float16,'InitialRtLProb':np.float16,
        'RtLDelta':np.float16,'MinimumRest':np.uint8,'InitialMinimumHold':np.uint8,
        'MaximumHold':np.uint8,'GlobalReach':np.float16,'ProximityThresh':np.float16,
        'Attitude':np.float16,'SwarmParticles':np.uint8,'SwarmSelfTrust':np.float16,
        'SwarmPastTrust':np.float16,'SwarmGlobalTrust':np.float16,'SwarmAnts':np.uint8,
        'MappingMean':np.float16,'MappingStdDev':np.float16,'MappingSeed':np.uint8,
        'RtMMin':np.uint8,'RtMMax':np.uint8,'RtMSeed':np.uint16,
        'RtLMin':np.uint8,'RtLMax':np.uint8,'RtLSeed':np.uint8,
```

```
        'ACOSeed':np.uint8,'TaskCompletedMin':np.uint8,'TaskCompletedMax':np.uint8,
        'TaskCompletedSeed':np.uint8,'TargetShuffleMin':np.uint8,
        'TargetShuffleMax':np.uint8,'TargetShuffleSeed':np.uint8,'NaiveMapping':np.uint8,
        'VStep':np.float16,'HStep':np.float16,'SimStepMax':np.uint32,
        'SimTrialNum':np.uint8,'ArgosSeed':np.uint8
}

def saveAsHDF(chunk):
        chunk.loc[chunk['Type'] == 'pso'].to_hdf(
                'output/pso.h5',  key = 'data', mode ='a', format='table', append = True)
        chunk.loc[chunk['Type'] == 'aco'].to_hdf(
                'output/aco.h5',  key = 'data', mode ='a', format='table', append = True)
        chunk.loc[chunk['Type'] == 'lawn'].to_hdf(
                'output/lawn.h5',  key = 'data', mode ='a', format='table', append = True)

for chunk in pd.read_csv(filename, chunksize=chunksize, dtype=datatypes):
         saveAsHDF(chunk)

# We then re-load our categorised hdf5 datasets piecemeal and compute their means.
pso = pd.read_hdf('output/pso.h5', 'data')
aco = pd.read_hdf('output/aco.h5', 'data')
lawn = pd.read_hdf('output/lawn.h5', 'data')
means = pd.DataFrame(columns=['pso', 'aco', 'lawn'])

means.pso = pso.groupby('TargetNum').mean()['SimStep']
means.aco = aco.groupby('TargetNum').mean()['SimStep']
means.lawn = lawn.groupby('TargetNum').mean()['SimStep']

print(means)
```

TargetNum	pso	aco	lawn
1	2.0	2.0	2.0
2	547.4	547.2	1108.0
3	548.6	1733.2	2144.4
4	5447.0	5447.0	5258.0
5	4559.4	5573.8	12269.0
6	4330.0	5025.6	16109.8
7	5249.6	4908.4	13759.0
8	4792.2	4798.0	15863.0
9	3745.4	4769.0	14957.2
10	6678.8	4761.8	17078.8
11	5054.0	2106.6	15673.0
12	2174.2	2168.4	15319.6
13	5976.8	5991.0	17299.4
14	5054.4	3224.6	16612.2
15	6288.8	9220.6	18502.0
16	3285.6	3285.8	18051.6
17	8141.4	6227.4	17336.2
18	14865.2	13041.2	18477.0
19	16711.4	13027.8	18062.8

4.3 Data Analysis

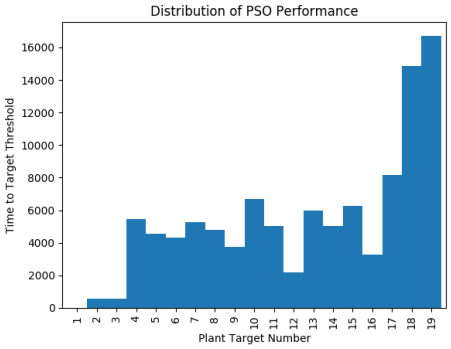
We visualize the plots to determine what their distributions may look like.

```
# Let's plot the mean values of each strategy
m_plot = means.plot(kind='line', style=['r', 'g', 'b'])
m_plot.set_title('Average Strategy Performance')
m_plot.set_xlabel('Plant Target Number')
m_plot.set_ylabel('Time to Target Threshold')
m_plot_fig = m_plot.get_figure()
m_plot_fig.savefig('thesis/images/means_line_plot.png', bbox_inches='tight')
```

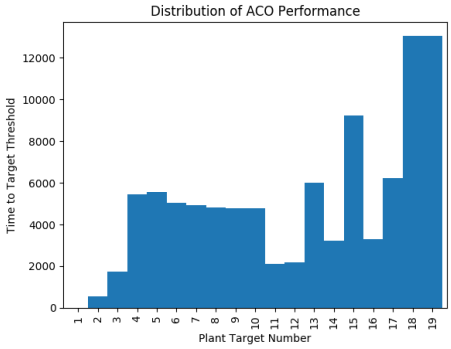
Figure 4.2: Mean Line Plots

We can also plot the strategy performances in bar charts to better infer their distribution.

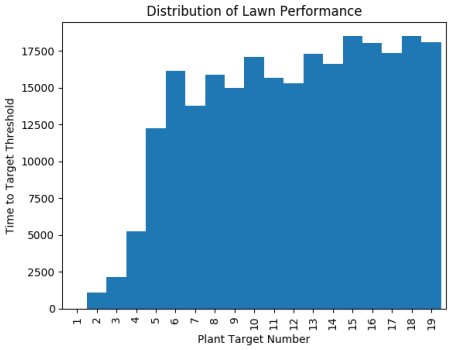
```
pso_plot = means.pso.plot.bar(width=1)
pso_plot.set_title('Distribution of PSO Performance')
pso_plot.set_xlabel('Plant Target Number')
pso_plot.set_ylabel('Time to Target Threshold')
pso_fig = pso_plot.get_figure()
pso_fig.savefig('thesis/images/pso_bar.png',bbox_inches='tight')

aco_plot = means.aco.plot.bar(width=1)
aco_plot.set_title('Distribution of ACO Performance')
aco_plot.set_xlabel('Plant Target Number')
aco_plot.set_ylabel('Time to Target Threshold')
aco_fig = aco_plot.get_figure()
aco_fig.savefig('thesis/images/aco_bar.png',bbox_inches='tight')

lawn_plot = means.lawn.plot.bar(width=1)
lawn_plot.set_title('Distribution of Lawn Performance')
lawn_plot.set_xlabel('Plant Target Number')
lawn_plot.set_ylabel('Time to Target Threshold')
lawn_fig = lawn_plot.get_figure()
lawn_fig.savefig('thesis/images/lawn_bar.png',bbox_inches='tight')
```



(a) Bar Plot of PSO Data



(b) Bar Plot of ACO Data



(c) Bar Plot of Lawn Data

Figure 4.3: Bar Plots

As can be seen in the figure above, there is a marked variability in the mean performance of the strategies, but a generally linear trend can be assumed to be present.

```
# We can comparatively visualize the distributions using a box plot
m_plot_box = means.plot.box()
m_plot_box.set_title('Strategy Distributions')
m_plot_box.set_ylabel('Time to Target Threshold')
m_plot_box_fig = m_plot_box.get_figure()
m_plot_box_fig.savefig('thesis/images/mean_box_plots.png',bbox_inches='tight')
```

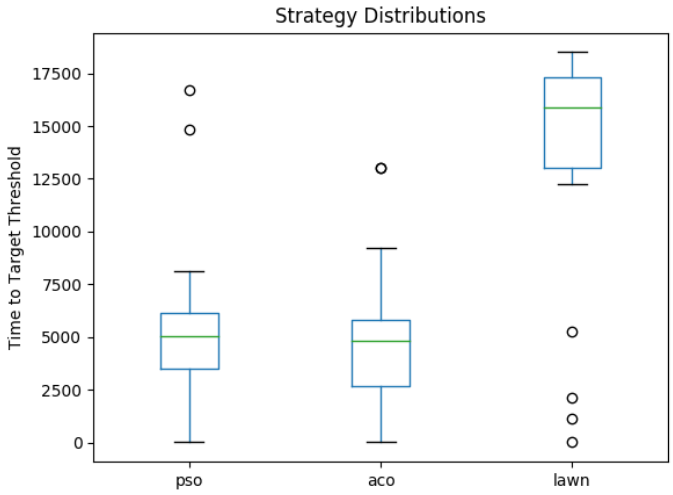
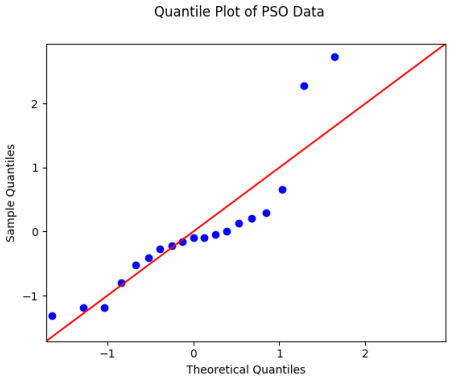


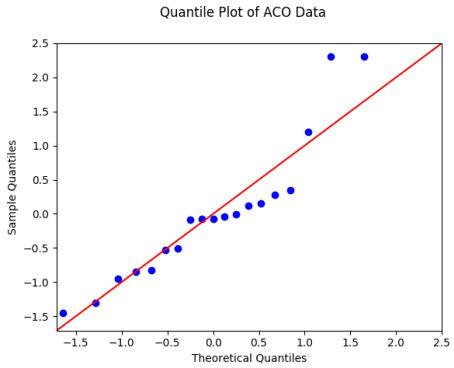
Figure 4.4: Mean Box Plots

We then identify the shape of our data distributions using Quantile-Quantile (Q-Q) plots. In the Q-Q plots below, the quantiles of our sample distributions are plotted against quantiles of a normal distribution as a scatter plot.

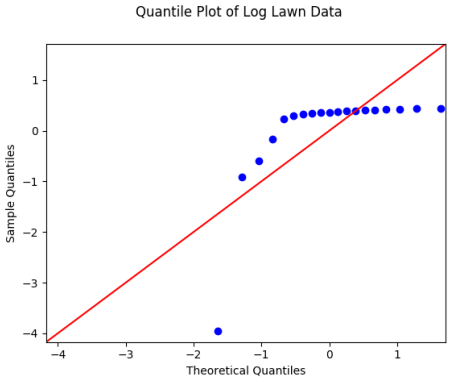
```
# We use statsmodels qqplot feature to test our distributions.
pso_qq_fig = sm.qqplot(means.pso, line='45', fit=True)
aco_qq_fig = sm.qqplot(means.aco, line='45', fit=True)
lawn_qq_fig = sm.qqplot(means.lawn, line='45', fit=True)

pso_qq_fig.suptitle('Quantile Plot of PSO Data')
aco_qq_fig.suptitle('Quantile Plot of ACO Data')
lawn_qq_fig.suptitle('Quantile Plot of Lawn Data')

pso_qq_fig.savefig('thesis/images/pso_qq.png',bbox_inches='tight')
aco_qq_fig.savefig('thesis/images/aco_qq.png',bbox_inches='tight')
lawn_qq_fig.savefig('thesis/images/lawn_qq.png',bbox_inches='tight')
```



(a) Q-Q Plot of PSO Data



(b) Q-Q Plot of ACO Data



(c) Q-Q Plot of Lawn Data

Figure 4.5: Q-Q Plots

It can be visually estimated that our data is normally distributed as the presented datasets generally follow their plotted diagonal lines. We could additionally use Scipys' normaltest utility to determine this normality, noting that the kurtosis test is only valid for sample sizes greater than or equal to 20.

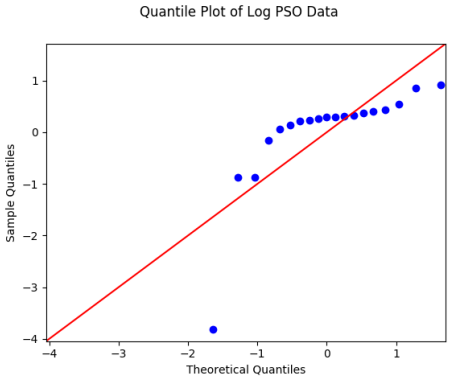
```
ntest_pso = stats.normaltest(means.pso)
ntest_aco = stats.normaltest(means.aco)
ntest_lawn = stats.normaltest(means.lawn)

print('The results of the normality tests are: \n
        PSO: -> tc-value = {:4.3f} tp-value = {:4.3f} \n
        ACO: -> tc-value = {:4.3f} tp-value = {:4.3f} \n
        LAWN: -> tc-value = {:4.3f} tp-value = {:4.3f}'.format(
```

```
        ntest_pso[0],ntest_pso[1],
        ntest_aco[0],ntest_aco[1],
        ntest_lawn[0],ntest_lawn[1])
)

The results of the normality tests are:
PSO: -> tc-value = 11.144 tp-value = 0.004
ACO: -> tc-value = 5.515 tp-value = 0.063
LAWN: -> tc-value = 6.670 tp-value = 0.036
```

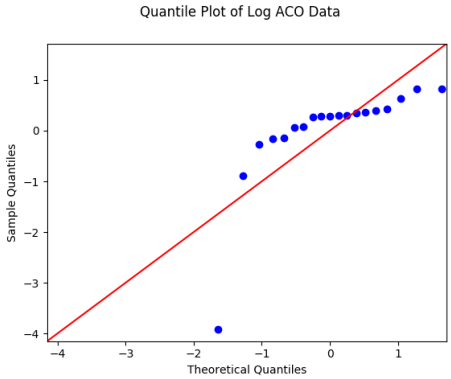
Before we can evaluate the result of this normality test, we must define our significance level (α) that specifies the probability of rejecting the null hypothesis when it is actually true. In this experiment, we elect to set the significance level of the statistical test to 0.05 as is common in the literature. The goal of a statistical test is to try and reject the null hypothesis.

We can now evaluate the result of our algorithmic normality test which returned p-values lower than our alpha level (0.05), indicating a rejection of the null hypothesis that the distributions are normal. However, the sample dataset generated only has $n = 19$ and was surmised to have skewed our tepid normality results as evidenced in the api output. If normality was not proveable in the exact sense, we could perform log, square root, or inverse transformations on our original data which **may** have led to better approximations to the normal distribution.

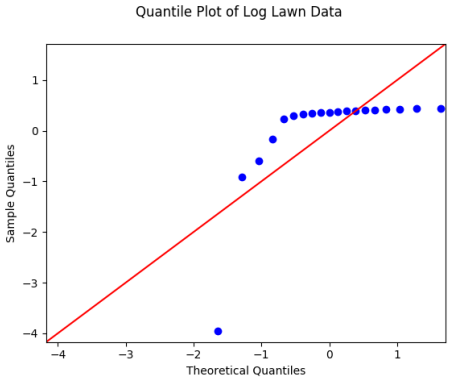
```
# We could perform a log transformation in the event that our original data proved non-normal.
# This doesn't guarantee normality, but is merely a technique used in the literature.
pso_qq_fig = sm.qqplot(np.log(means.pso), line='45', fit=True)
aco_qq_fig = sm.qqplot(np.log(means.aco), line='45', fit=True)
lawn_qq_fig = sm.qqplot(np.log(means.lawn), line='45', fit=True)
pso_qq_fig.suptitle('Quantile Plot of Log PSO Data')
aco_qq_fig.suptitle('Quantile Plot of Log ACO Data')
lawn_qq_fig.suptitle('Quantile Plot of Log Lawn Data')

pso_qq_fig.savefig('thesis/images/pso_log_qq.png',bbox_inches='tight')
aco_qq_fig.savefig('thesis/images/aco_log_qq.png',bbox_inches='tight')
lawn_qq_fig.savefig('thesis/images/lawn_log_qq.png',bbox_inches='tight')
```



(a) Q-Q Log Plot of PSO Data



(b) Q-Q Log Plot of ACO Data



(c) Q-Q Log Plot of Lawn Data

Figure 4.6: Q-Q Log Plots

We have proven normality and now must prove homoscedasticity. A number of tests exist to perform this check, including: - Bartlett's Test (parametric) - Levene's Test (parametric) - Fligner's Test (non-parametric)

They exhibit decreasing sensitivity to strong departures from normality with Fligner's being the most robust. We limit ourselves to these tests as they have implementations in the SciPy Python library. We take a middle-of-the-road approach and go for Levene's Test as computed below.

```
lev_pa = levene(means.pso,means.aco)
lev_pl = levene(means.pso,means.lawn)
lev_al = levene(means.aco,means.lawn)

print('The results of the homoscedasticity tests are: \n
        PSO: -> tw-value = {:4.3f} tp-value = {:4.3f} \n
        ACO: -> tw-value = {:4.3f} tp-value = {:4.3f} \n
        LAWN: -> tw-value = {:4.3f} tp-value = {:4.3f}'.format(
        lev_pa[0],lev_pa[1],
        lev_pl[0],lev_pl[1],
        lev_al[0],lev_al[1])
)
```

```
The results of the homoscedasticity tests are:
PSO: -> tw-value = 0.063 tp-value = 0.803
ACO: -> tw-value = 0.960 tp-value = 0.334
LAWN: -> tw-value = 1.475 tp-value = 0.232
```

As can be seen in the above result, our homoscedasticity test p-values are higher than our alpha level (0.05) which means we have failed to reject the null hypothesis that the distributions exhibit equal variances. If homoscedasticity could not be determinably proven, we could elect to use Welsch's t-Test that works better with unequal sample variances. Alternatively, we could perform a variable transformation such as a Box-Cox transformation.

We can now perform our t-test, where sample independance is assured on account of the simulation random-trialling mentioned previously. We shall use the independant, two-sample t-test as shown below:

4.4 Hypothesis Test

```
# Run independent two-sample, t-tests
ind_t_test_pa = ttest_ind(means.pso,means.aco)
```

```
ind_t_test_pl = ttest_ind(means.pso,means.lawn)
ind_t_test_al = ttest_ind(means.aco,means.lawn)

print('The results of the independent t-tests are:
        \nPSO:ACO -> tt-value = {:4.3f} tp-value = {:4.3f} \n
        PSO:LAWN -> tt-value = {:4.3f} tp-value = {:4.3f} \n
        ACO:LAWN -> tt-value = {:4.3f} tp-value = {:4.3f}'.format(
        ind_t_test_pa[0],ind_t_test_pa[1],
        ind_t_test_pl[0],ind_t_test_pl[1],
        ind_t_test_al[0],ind_t_test_al[1]
        )
)
```

```
The results of the independent t-tests are:
PSO:ACO -> tt-value = 0.315 tp-value = 0.755
PSO:LAWN -> tt-value = -4.577 tp-value = 0.000
ACO:LAWN -> tt-value = -5.051 tp-value = 0.000
```

For completeness, we performed a t-test on our pso and aco samples and got a p-value of ~0.755, indicating that there is no statistically significant difference between the two means. More interestingly though, and of central importance to this work, we have p-values of $< \alpha$ (0.05) for the pso-lawn and aco-lawn tests, which is an indication of statistically significant difference between their means. This means we can assuredly reject our null hypothesis in these scenarios.

Chapter 5

Conclusion

A more robust and flexible approach utilising the Maximum Likelihood Estimation (MLE) of trial durations was considered and premised to be better placed towards stemming this problem in a more *realistic* fashion. Practically, this enhancement would allow for local agents to probabilitically posit how long the trial is expected to run and at which point to determinably *give up* on local or global objectives. Plant configuration was found to be a confounding variable whose effect was mitigated through re-trials.

If it is something other than binary and if you are really worried – run the t-test and then run the Wilcoxon-Mann-Whitney on the same data and see what you see. The t-test is robust with respect to non-normality but if the data gets too extreme the test can fail to detect a difference in mean location when one exists. The Wilcoxon works under all conditions that would be appropriate for a t-test but it does a better job (has higher power) in cases of extreme asymmetry.

An interesting investigation would be to analyse the CPU load of each trial to analyse strategy performance from a computational standpoint.

Optimization of CPP Strategy Parameters. Future work into leaderless, self-organisation strategies with greater swarm sizes. The former can be achieved via agent roaming procedures that incorporate random walk behaviour with swarm cohesion implemented through Leinard-Jones potentials between agents. Contributions include a programmatic framework for the generation of simulation data suitable for biological statistical analysis. The need for benchmark datasets in the field is a noted need and it is hoped that added effort is placed towards the release of standardised benchmarking sets. Established path planners such as Open Motion Planning Library (OMPL) [78] should be considered for enhanced validity in comperative analyses. Further, higher order models of the quadcopter and external disturbances such as wind should be considered to enhance simulator realism. Higher order models are possible employing techniques to generating models learned from flight data [79] whereas robust wind models such as the Dryden wind

turbulence model [2] could be included. Collision detection and obstacle avaoidance with 3D-capable metaheuristic planners. Techniques and approaches to parameter optimisation and social learning such as genetic programming or neural networks are high potential, interest areas that would be considered to augment this work in future. A standardised metaheuristic optimization algorithm library such as [45] and a wider variety state prediction techniques such as probabilistic smoothing. SLAM Mapping of targets. Multi-objective TSP. Enhanced local communication with the ring leds. Implement time estimation techniques to vary holding times. A dataset repository of similarly generated simulation data with a standardised scientific toolkit, an example of which is PySwarms [45]. There is still much to be desired in the terminology and frameworks available in literature. As explained in the introduction of [67], Implement the DPSO in a GPU-based framework so that the computation time can be significantly reduced while keeping the hardware requirement unchanged. A better integrated simulator such as MORSE that utilises Blender as its' 3D modelling engine would allow for more visually realistic scenes. Alternatively, AirSim, a recently released autonomous vehicle simulation platform could be used. This would open the field to novel behaviour modelling approaches via deep learning, a field in recent resurgence due to enhanced GPU computing capabilities and access to swaths of training data. Studies in to the response of the MRS to adaptivity and fault tolerance (Iocchi2001).

As previously mentioned, this work makes the following contributions to the field:

- A novel approach to comparatively evaluate the performance of GREW-MRS strategies and implementations. The application of statistical testing is also introduced with novel measurement variables proposed for wider consideration. Data generation is addressed with appropriately developed tooling provided.
- A novel and simple simulation pipeline with code made freely available. This works' software repository is made freely available and is scripted as an ARGoS experiment that is dynamically configured with the help of a shell script. It can be found at `https://github.com/wndaiga/swarm_ucl` [62].
- A sample dataset of 10 independantly and randomly seeded simulation trials over a range of Green Wall target/plant numbers (2-50).

It is hoped that this research output proves beneficial to fellow Swarm roboticists towards the advancement of these nascent but rapidly growing and promising fields.

Bibliography

[1] Acm announces recipients of four prestigious technical awards for 2017.

[2] Matlab documentation.

[3] Missionlab v7.0.

[4] The pandas project.

[5] Project jupyter.

[6] Statistical technical papers.

[7] *Swarm Intelligence Market worth 447.2 Million USD by 2030*.

[8] What is hdf5?

[9] When can count data be considered continuous?

[10] Disney makes history with first u.s. light show powered by 300 intel drones, Dec 2016.

[11] Vertical farming market size, share – industry forecast report 2024, Apr 2017.

[12] Chirantan Banerjee. Up , Up and Away ! The Economics of Vertical Farming. 2(1):40–60, 2014.

[13] Levent Bayindir. A review of swarm robotics tasks. *Neurocomputing*, 172:292–321, 2016.

[14] Gerardo Beni. From swarm intelligence to swarm robotics. *Ant Colony Optimization and Swarm Intelligence Proceedings*, 3342(July 2004):1–9, 2005.

[15] A. Bircher, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, and R. Siegwart. Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6423–6430, May 2015.

[16] BMGI. Hypothesis testing roadmap.

[17] Harm Jan Boonstra. *A simulation study of repeated weighting estimation*. Statistics Netherlands, Voorburg [etc.], 2004.

[18] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

[19] Universite Libre De Bruxelles. Ant colony optimization for the traveling salesman problem.

[20] Zuo Llang Cao, Yuyu Huang, and Ernest L. Hall. Region filling operations with random obstacle avoidance for mobile robots. *Journal of Robotic Systems*, 5(2):87–102, 1988.

[21] Howie Choset. Coverage for robotics - A survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):113–126, 2001.

[22] Carlos A. Coello Coello and Margarita Reyes-Sierra. Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. *International Journal of Computational Intelligence Research*, 2(3):287–308, 2006.

[23] Erik Cuevas, Miguel Cienfuegos, Daniel Zaldívar, and Marco Pérez-cisneros. A swarm optimization algorithm inspired in the behavior of the social-spider. *Expert Systems With Applications*, 40(16):6374–6384, 2013.

[24] Carmelo Di Franco and Giorgio Buttazzo. Energy-aware coverage path planning of UAVs. *Proceedings - 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC 2015*, pages 111–117, 2015.

[25] Marco Dorigo and G Di Caro. The Ant Colony Optimization Meta-Heuristic. *New Ideas in Optimization*, 2:11–32, 1999.

[26] Marco Dorigo, Dario Floreano, Luca Maria Gambardella, Francesco Mondada, Stefano Nolfi, Tarek Baaboura, Mauro Birattari, Michael Bonani, Manuele Brambilla, Arne Brutschy, and et al. Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20(4):60–71, 2013.

[27] Marco Dorigo and Luca Maria Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *{IEEE}{T}rans. on {E}volutionary {C}omputation*, 1(1):53–66, 1997.

[28] Marco Dorigo and Erol Şahin. Guest editorial. *Autonomous Robots*, 17(2/3):111–113, 2004.

[29] John R.G. Dyer, Christos C. Ioannou, Lesley J. Morrell, Darren P. Croft, Iain D. Couzin, Dean A. Waters, and Jens Krause. Consensus decision making in human crowds. *Animal Behaviour*, 75(2):461–470, 2008.

[30] Gilberto Echeverria, Nicolas Lassabe, Arnaud Degroote, and Severin Lemaignan. Modular open robots simulation engine: Morse. *2011 IEEE International Conference on Robotics and Automation*, 2011.

[31] Stefan Enderle, Hans Utz, Stefan Sablatnög, Steffen Simon, Gerhard Kraetzschmar, and Günther Palm. Miro: Middleware for autonomous mobile robots. *Citeseer Ist Psu Eduenderle01Miro Html*, pages 297–302, 2001.

[32] Andy P. Field and Graham Hole. *How to design and report experiments*. Sage, 2012.

[33] J. Fredslund and M.J. Mataric. A general algorithm for robot formations using local sensing and minimal communication. *IEEE Transactions on Robotics and Automation*, 18(5):837–846, 2002.

[34] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013.

[35] Amit Garg, Pawan Gill, Parveen Rathi, and Dr K K Garg. An insight into swarm intelligence. 2, 01 2009.

[36] Luuk Graamans, Esteban Baeza, Andy van den Dobbelsteen, Ilias Tsafaras, and Cecilia Stanghellini. Plant factories versus greenhouses: Comparison of resource use efficiency. *Agricultural Systems*, 160:31–43, 2018.

[37] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, Feb 2007.

[38] Jose Alfredo Guerrero and Yasmina Bestaoui. UAV path planning for structure inspection in windy environments. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 69(1-4):297–311, 2013.

[39] R h. Renssen, A h. Kroese, and A J. Willeboordse. Aligning estimates by re-peated weighting. 01 2001.

[40] Hmartiro. Cpp implementation of a kalman filter, Jan 2015.

[41] Dave Hollander. Bluetooth mesh - what a difference a year makes, Jul 2018.

[42] Steve Holt. Is vertical farming really the future of agriculture?, Jul 2018.

[43] Icdts. Discrete particle swarm optimization for the traveling salesman problem.

[44] Luca Iocchi, Daniele Nardi, and Massimiliano Salerno. Reactivity and Deliberation: A Survey on Multi-Robot Systems. pages 9–32, 2001.

[45] Lester James V. Miranda. PySwarms: a research toolkit for Particle Swarm Optimization in Python. *The Journal of Open Source Software*, 3(21):433, 2018.

[46] John H. Mc.Donald. Handbook of Biological Statistics. *Sokal & Rohlf (1981)*, pages 41–58, 2008.

[47] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[48] Fatemeh Kalantari, Osman Mohd Tahir, Ahmad Mahmoudi Lahijani, and Shahaboddin Kalantari. A Review of Vertical Farming Technology: A Guide for Implementation of Building Integrated Agriculture in Cities. *Advanced Engineering Forum*, 24(October):76–91, 2017.

[49] Graham Kalton and Ismael Flores-Cervantes. Weighting methods. *Journal of Official Statistics*, 19(2):81–97, 2003.

[50] J Kennedy and R Eberhart. Particle swarm optimization. *Neural Networks, 1995. Proceedings., IEEE International Conference on*, 4:1942–1948 vol.4, 1995.

[51] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 3:2149–2154.

[52] Scheutz M Kramer J. Development environments for autonomous mobile robots: A survey. *Autonomous Robots*, 22(2):101–132, 2007.

[53] Thomas S. Kuhn. *The structure of scientific revolutions*. The University of Chicago Press, 2015.

[54] Séverin Lemaignan, Marc Hanheide, Michael Karg, Harmish Khambhaita, Lars Kunze, Florian Lier, Ingo Lütkebohle, and Grégoire Milliez. Simulation and hri recent perspectives with the morse simulator. *Simulation, Modeling, and Programming for Autonomous Robots Lecture Notes in Computer Science*, page 13–24, 2014.

[55] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Auton. Robots*, 4(4):333–349, October 1997.

[56] Martin Macas, Martin Saska, Lenka Lhotska, Libor Preucil, and Klaus Schilling. Path Planning for Formations of Mobile Robots using PSO Technique. (January), 2009.

[57] Maria Manso and João Castro-Gomes. Green wall systems: A review of their characteristics. *Renewable and Sustainable Energy Reviews*, 41:863–871, 2015.

[58] Maja J Matarić. Designing and Understanding Adaptive Group Behavior. *Adaptive Behavior*, 4(1):51–80, 1995.

[59] Neil Mattson. Viability of indoor urban agriculture is focus of research grant | cornell chronicle.

[60] Olivier Michel. Cyberbotics Ltd. Webots™: Professional Mobile Robot Simulation. *International Journal of Advanced Robotic Systems*, 1(1):5, 2004.

[61] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. *Proc. of 8th National Conference on Artificial Intelligence/14th Conference on Innovative Applications of Artificial Intelligence*, 68(2):593–598, 2002.

[62] Sylvester Wachira Ndaiga. Thesis swarm project code.

[63] Justin Eugene Noronha. Development of a swarm control platform for educational and research applications. *Thesis*, page 106, 2016.

[64] Edwin Olson, John Leonard, and Seth Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2262–2269, 2006.

[65] Matthew O'Brien, Ronald Arkin, Dagan Harrington, Damian Lyons, and Shu Jiang. Automatic verification of autonomous robot missions. 8810:462–473, 10 2014.

[66] Katia Perini, Marc Ottelé, A L A Fraaij, E M Haas, and Rossana Raiteri. Vertical greening systems and the effect on air flow and temperature on the building envelope. *Building and Environment*, 46(11):2287–2294, 2011.

[67] Manh Duong Phung, Cong Hoang Quach, Tran Hiep Dinh, and Quang Ha. Enhanced discrete particle swarm optimization path planning for UAV vision-based surface inspection. *Automation in Construction*, 81(April):25–33, 2017.

[68] Carlo Pinciroli. On the Design and Implementation of an Accurate , Efficient , and Flexible Simulator for Heterogeneous Swarm Robotics Systems. 2014.

[69] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, Timothy S. Stirling, Álvaro Gutiérrez, Luca Maria Gambardella, and Marco Dorigo. ARGoS: A pluggable, multi-physics engine simulator for heterogeneous swarm robotics. *IRIDIA – Technical Report Series*, (December 2010):1–22, 2011.

[70] Nandini Rathi. Mexico's via verde shows path for indian cities to tackle pollution, Jan 2017.

[71] Gerhard Reinelt. Tsplib. a traveling salesman problem library. 3:376–384, 11 1991.

[72] Erol Sahin. Swarm robotics: From sources of inspiration to domains of application. *Swarm Robotics Lecture Notes in Computer Science*, page 10–20, 2005.

[73] V. Selvi and Dr.r. Umarani. Comparative analysis of ant colony and particle swarm optimization techniques. *International Journal of Computer Applications*, 5(4):1–6, Oct 2010.

[74] Shital Shah, Debadeepta Dey, Chris Lovett, A Kapoor Field Robotics, Service, and Undefined 2018. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *Springer*, pages 1–14.

[75] Helen Shen. Interactive notebooks: Sharing the code.

[76] Samar Sheweka and Nourhan Magdy. The living walls as an approach for a healthy urban environment. *Energy Procedia*, 6:592–599, 2011.

[77] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. *Proceedings. 1987 IEEE International Conference on Robotics and Automation*.

[78] Ioan A. Sucan, Mark Moll, and Lydia E. Kavraki. The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012.

[79] Andrew Symington, Renzo De Nardi, Simon Julier, and Stephen Hailes. Simulating Quadrotor UAVs in Outdoor Scenarios. (Iros):3382–3388, 2014.

[80] Ying Tan and Zhong yang Zheng. Research Advance in Swarm Robotics. *Defence Technology*, 9(1):18–39, 2013.

[81] James R. Taylor, Evan M. Drumwright, and Gabriel Parmer. Making time make sense in robotic simulation. *Simulation, Modeling, and Programming for Autonomous Robots Lecture Notes in Computer Science*, page 1–12, 2014.

[82] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y Ng, Zoubin Ghahramani, and Hugh Durrant-Whyte. Simultaneous Localization and Mapping with Sparse Extended Information Filters. *The International Journal of Robotics Research*, 23(7-8):693–716, 2004.

[83] Huu Khoa Tran and Juing Shian Chiou. PSO-based algorithm applied to quadcopter micro air vehicle controller design. *Micromachines*, 7(9):1–8, 2016.

[84] UNFCCC. Paris Agreement. *Conference of the Parties on its twenty-first session*, (December):32, 2015.

[85] M. Weiser. The computer for the 21st Century. *IEEE Pervasive Computing*, 1(1):19–25, jan 2002.

[86] Xuesong Yan, Can Zhang, Wenjing Luo, Wei Li, Wei Chen, and Hanmin Liu. Solve Traveling Salesman Problem Using Particle Swarm Optimization Algorithm. 9(6):264–271, 2012.

[87] Zhi Yan, Luc Fabresse, Jannik Laval, and Noury Bouraqadi. Team size optimization for multi-robot exploration. In *Proceedings of the 4th International Conference on Simulation, Modeling, and Programming for Autonomous Robots - Volume 8810*, SIMPAR 2014, pages 438–449, Berlin, Heidelberg, 2014. Springer-Verlag.

[88] Guang-Zhong Yang, Jim Bellingham, Pierre E. Dupont, Peer Fischer, Luciano Floridi, Robert Full, Neil Jacobstein, Vijay Kumar, Marcia McNutt, Robert Merrifield, Bradley J. Nelson, Brian Scassellati, Mariarosaria Taddeo, Russell Taylor, Manuela Veloso, Zhong Lin Wang, and Robert Wood. The grand challenges of Science Robotics. *Science Robotics*, 3(14):eaar7650, 2018.

[89] Xin-She Yang. Metaheuristic optimization: Algorithm analysis and open problems. *Experimental Algorithms Lecture Notes in Computer Science*, page 21–32, 2011.

[90] Habib Youssef, Sadiq M Sait, and Hakim Adiche. Evolutionary algorithms, simulated annealing and tabu search: a comparative study. *Engineering Applications of Artificial Intelligence*, 14(2):167–181, 2001.

[91] Ioan A. Șucan, Mark Moll, and Lydia E. Kavraki. The open motion planning library.

Appendix A

Appendix

Parameter	Description	DataType
Type	Indicates the CPP strategy	String
TargetNum	Indicates the total number of targets in simulation	Int
TargetThresh	Indicates the computed target completion threshold	Int
Step	Simulation time step	Int
Completed	Number of targets marked as attended	Int
X	X Position of leader agent	Float
Y	Y Position of leader agent	Float
Z	Z Position of leader agent	Float
RtMProb	Probability that the leader agent will move from rest to move state	Float
RtLProb	Probability that the leader agent will move from rest to land state	Float
MinimumHold	Minimum simulation time that agents must wait in hold state	Int
LaunchStep	Minimum simulation time that leader agent must wait before launching (only used in lawn strategy)	Int
InitialRtMProb	Initial probability that the leader agent will move from rest to move state	Float
RtMDelta	Delta value used to modify the probability that the leader agent will move from rest to move state	Float
InitialRtLProb	Initial probability that the leader agent will move from rest to land state	Float
RtLDelta	Delta value used to modify the probability that the leader agent will move from rest to land state	Float
MinimumRest	Minimum simulation time that agents must wait in rest state	Int
InitialMinimumHold	Initial minimum simulation time that agents must wait in hold state	Int
MaximumHold	Maximum simulation time that agents can wait in hold state	Int
GlobalReach	Mean distance to wall that the collective must maintain	Float
ProximityThresh	Minimum positional distance error	Float
Attitude	Height above targets that leader agent must maintain	Float
SwarmParticles	PSO parameter to set the number of particles	Int
SwarmSelfTrust	PSO parameter to set the particles' current solution weight value	Int
SwarmPastTrust	PSO parameter to set the particles' past solution weight value	Int
SwarmGlobalTrust	PSO parameter to set the particles' swarm solution weight value	Int
SwarmAnts	ACO parameter to set the number of ants in colony	Int
MappingMean	Mapping, gaussian distribution mean	Float
MappingStdDev	Mapping, gaussian distribution standard deviation	Float
MappingSeed	Mapping, gaussian distribution seed	Int
RtMMin	Rest to move, uniform integer distribution min	Int
RtMMax	Rest to move, uniform integer distribution max	Int
RtMSeed	Rest to move, uniform integer distribution seed	Int
RtLMin	Rest to land, uniform integer distribution min	Int
RtLMax	Rest to land, uniform integer distribution max	Int
RtLSeed	Rest to land, uniform integer distribution seed	Int
ACOSeed	ACO Optimizer seed value	Int
TaskCompletedMin	Task completion, uniform integer distribution min	Int
TaskCompletedMax	Task completion, uniform integer distribution max	Int
TaskCompletedSeed	Task completion, uniform integer distribution seed	Int
TargetShuffleMin	Target shuffle, uniform integer distribution min	Int
TargetShuffleMax	Target shuffle, uniform integer distribution max	Int
TargetShuffleSeed	Target shuffle, uniform integer distribution seed	Int
NaiveMapping	Target mapping scheme (not fully implemented)	Int
VStep	Vertical step used in lawn strategy	Int
HStep	Horizontal step used in lawn strategy	Int
ArgosSeed	Argos seed (configures the placement of targets)	Int

Table A.1: CSV Dataset Parameters