Coverage path planning for agricultural ground multi-robots under complex terrain conditions

by

Dania Gisela Martinez Figueroa

B.S., University of Narino, 2019

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering
Carl. R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2023

Approved by:

Co-Major Professor
Dr. Sanjoy Das

Approved by:

Co-Major Professor
Dr. Stephen Welch

Copyright

Abstract

As steep terrain conditions are considered to be too risky for human operators, agricultural tasks must be carried out by multi-robots. This research considers minimum energy path planning by a team of ground robots for full coverage of uneven terrain. Experimental data from a real robot is used to develop a machine learning model to estimate minimum energy straight line paths between pairs of proximally located points. Longer paths are approximated as sequences of line segments. Exemplar-based clustering is used to identify a set of waypoints, which can be used for sensor placement, to ensure full terrain coverage. Treating the waypoints as the vertices of a digraph, optimal cyclic paths for navigation by a team of agricultural robots are determined. The proposed approach for waypoint identification and optimal path discovery can be implemented through local message passing in a sensor network. Comparison with other recently proposed methods, and simulations using real-world elevation data establish the effectiveness of the proposed approach.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I am deeply thankful to my advisor, Dr. Sanjoy Das, whose unwavering support and patience guided me through the challenging journey of my master's degree. His mentorship was invaluable, and I will always be grateful for the dedication he showed in helping me achieve this document. Beyond his role as an advisor, Dr. Das became a cherished friend, embodying the qualities of kindness and generosity.

I extend my heartfelt appreciation to the collaborative efforts of Dr. Welch and Dr. Flippo, whose teamwork and valuable insights significantly enriched this research.

I would also like to express my gratitude to the Department of Electrical and Computer Engineering for providing me with the opportunity to be a part of this incredible university. Special thanks to Dr. Rys, whose genuine care for his students and unwavering support played a pivotal role in our successes.

I am profoundly fortunate to have been surrounded by such dedicated and supportive individuals throughout this academic journey.

Dedication

Dedicated to my mom Maria Eugenia, who watches over me from the sky.

Chapter 1 - Previous work and Research Contributions

KANSAS is one of the most productive agricultural state in the US, specially collecting high yields of wheat, corn and soybeans. However, the northeastern region of the state has not been utilized for cultivation (Middendorf et al., 2009). This is because the region is characterized by the presence of steep escarpments of up to 400 feet (Smith D., 2012), rendering it hazardous for humans to operate conventional farm machinery. Fortunately, rapid technological advancements in robotics have made it possible to apply agricultural ground robots in such terrain conditions (Santos et al., 2020). This makes it feasible to irrigate thousands of acres of arable land that is available in the region.

This research considers the use of a team of mobile robots for various cropping applications in uneven landscapes, such as seed dispersal, watering, fertilizer delivery, crop health inspection, soil monitoring, and pest control. In each such application, all plant sites must be visited by at least one robot – a criterion shall be referred to henceforth as the *total coverage* requirement (Chakraborty et al., 2022). Robot navigation through complex agricultural terrain is associated with significant energy usage (Badgujar et al., 2022). Accordingly, the approach proposed in this research is *energy optimal* – the objective being to minimize the total energy consumed by the robots. The robots in the present class of applications are assigned the same task. Therefore, a *homogeneous* robot team is assumed, where all robots are physically identical to each other. The robots are also equipped with identical batteries for energy storage, whose limited capacities serves as *energy constraints* (Moysiadis et al., 2020).

Taking into account these considerations, the overall objective has been formulated as a *coverage path planning* (CPP) problem for robot navigation (Galceran & Carreras, 2013). A preliminary version of this research that addresses single robot CPP appears in (Martinez-Figueora et al., 2022). In addition to extending that earlier research to multiple robot environments, this manuscript details several other research innovations, which have not been published elsewhere.

Prior work on Multi-Robot Path Planning

Coverage path planning is one of the major classes of planning problems in mobile robots (Tan et al., 2021). Various approaches, including ones that typically rely on graph-theoretic algorithms, as well as methods based on metaheuristics such as genetic algorithms (GA), ant colony optimization (ACO) and particle swarm optimization (PSO) have been explored (Liu et al., 2023).

A spanning tree algorithm is considered in (X. Huang et al., 2020). Tree-based search algorithms such as A* and RRT* have been successfully applied to robotic path planning in [(Le et al., 2018), (Qi et al., 2021),(L. Zhang et al., 2021)]. PSO has been used in (Tang et al., 2020) and (L. Zhang et al., 2021). A hybrid algorithm combining PSO and A* has been considered in (C. Huang et al., 2023). GA approaches have been explored in (Le et al., 2020), and (Mukhamediev et al., 2023). Elsewhere, a GA approach has been used in (Király & Abonyi, 2015). Although not intended for CPP, underlying similarities render this approach suitable for comparison with this research. A hybrid scheme involving ACO has been investigated for CPP applications in (Y. Wu et al., 2022). Reinforcement learning has been used very recently in (M. Zhang et al., 2023).

A significant amount of research has been directed towards either aerial robots/UAVs or heterogeneous robot teams consisting of aerial as well as mobile ground units (Ju et al., 2022). A CPP method for UAVs has been suggested in (Jensen-Nau et al., 2021). Voronoi partitioning of the terrain is first applied to identify a suitable set of waypoints, which are then used to construct robot paths. Although the study in (Jensen-Nau et al., 2021) is directed towards UAVs, owing to underlying similarities, it has been adopted for comparison with the proposed approach. Voronoi partitioning has also been considered for other CPP applications in (Yazici et al., 2014) and (Chi et al., 2022). An approach aimed at deploying UAVs in agricultural application is reported in (Mukhamediev et al., 2023). CPP algorithms for other aerial applications have been studied in (Delmerico et al., 2017), (Almadhoun et al., 2022).

Approaches for total coverage have been investigated for an application involving cleaning robots in (Miao et al., 2018), and for general-purpose 3-D terrain exploration in (C. Wu et al., 2019). As in this research, the approach in (C. Wu et al., 2019) also aims to minimize energy consumption. In other domains where total coverage is not essential, the planning algorithm may seek to maximize coverage within certain operating constraints, such as robotic resilience to adversarial attacks (Ishat-E-Rabban & Tokekar, 2021) and terrain uncertainties (Van Pham et al., 2019). In a comparable fashion to this study, energy usage serves as a constraint in (Yazici et al., 2014). The agricultural weed removal application in (Maini et al., 2022) aims to maximize the quantity of weed detected. Instead of energy usage, this approach considers the total robot trajectory length as a performance metric.

Research Contributions

In the proposed approach, the region is approximated as a discrete set of sample points. The basic approach consists of two stages. First, a clustering algorithm is applied to identify a subset of waypoints from the sample points. Next, Min-sum message passing algorithm is applied to obtain robotic trajectories from the waypoints. In addition to the basic research, a supervised learning method to estimate optimal paths between waypoints from experimental data is proposed. Lastly, a novel method to further refine the waypoints, and thence robotic trajectories using reinforcement learning, has been suggested.

The significant contributions in this research are in the following directions:

(*i*) A new formalism to describe the CPP problem is introduced in Chapter 1 - . Subsequently, a discretized and approximate variant that transforms any specific instance of the problem as one on weighted digraphs, is suggested. Various theoretical performance bounds can be readily established using the proposed framework.

(*ii*) The basic approach suggested as a part of this research is implemented through distributed Message Passing. It can be realized entirely through localized communications within a sensor network (Farinelli et al., 2014). Alternately, it can also be implemented in parallel in a GPU environment (Fioretto et al., 2018). This feature has potential benefit during real-world deployment.

(*iii*) A novel message passing algorithm is proposed to obtain optimal multi-robot trajectories in weighted digraphs, which is related to the multiple traveling salesperson (TSP) (Li et al., 2015) and vehicle routing problems (Kumar & Panneerselvam, 2012), which are known to be in NP-hard. This contribution is of theoretical importance as it can be adopted for various other graph theoretic optimization algorithms, including but not limited to path planning problems in mobile robots.

(*iv*) A novel method to estimate the robot velocity to travel from one location to another with minimum energy is outlined. The method uses data collected from laboratory experiments with the robot designed for this seed delivery application. The experimental data is used to train a set of machine learning regression models (cf. (C. Badgujar et al., 2023)) which deliver robot performance measures (C. Badgujar et al., 2022), (C. Badgujar et al., 2023). The minimum traversal energy is obtained from the model outputs using a continuous optimization algorithm.

The proposed scheme can be readily adopted for other path planning purposes to effectively leverage experimental data.

(*v*) A reinforcement learning algorithm is proposed to further fine-tune the solution obtained from the basic message passing approach. Topology based heuristics are incorporated into the learning algorithm, which are generic enough to be useful for other terrain applications.

Multi-Robot CPP Problem Description

In the context of agricultural operations, a fleet of robots is assigned the critical task of uniformly distributing seeds across a designated area. These robots start and end their journey at the same initial point. The essence of this research lies in finding their trajectories to minimize their collective energy expenditure while ensuring fully coverage. It is necessary to establish a coverage radius for the robots, taking into account their ability to spread seeds while traversing the region. The terrain of the given area is characterized by irregular elevations, resulting in varying energy costs when ascending versus descending hills. This disparity transforms the problem into an asymmetric multi-robot CPP. Additionally, it is essential to maintain a comparable path length for each robot, a constraint known as the *Balance tour condition* which will be elaborated upon in subsequent sections. A comprehensive and detailed description of this problem, including its formal specifications, will be presented in the following chapters.

Chapter 2 - Problem Formulation

Exact Problem Formulation

We consider an arbitrary contiguous tract of land $\mathcal{R}$ that does not contain humanmade structures, geological discontinuities (e.g., rocks, holes, fissures, fault planes); $\mathcal{R}$ shall be referred to as a *region*. Any point $\mathbf{x} \in \mathcal{R}$ is a vector $[x, y, z]^{\mathrm{T}}$, where x and y are its latitudinal and longitudinal coordinates – hereafter referred to as its *horizontal* coordinates, and z is its elevation. Mathematically speaking, $\mathcal{R}$ lies within a closed and bounded Riemannian surface (i.e., a 2-manifold) in $\mathbb{R}^3$ (Petersen, 2016).

Furthermore, region $\mathcal{R}$ is assumed to be soil *isotropic*, so that the work done by a robot to move along a path is dependent only on the geometric features of that path. Soil isotropy is a justifiable assumption in many agricultural tasks, including in the application described here, which is that of seed dispersal in any region identified for irrigation. It must be emphasized that the framework is generalized enough, rendering it feasible for direct use in a very wide variety of agricultural problems.

Let $\mathcal{G}$ be the set of identical robots available for CPP. From any location $\mathbf{x} \in \mathcal{R}$, a robot can deliver seeds up to a maximum horizontal distance ρ, where ρ is its *coverage radius*. Thus, the region covered by a robot located at $\mathbf{x}$ is,

$$\mathcal{R}_{\mathbf{x}}^{\rho} \triangleq \{\mathbf{x}' \in \mathcal{R} | (x - x')^2 + (y - y')^2 \leq \rho^2\}. \tag{1}$$

A curve φ in $\mathcal{R}$ is a sequence of contiguous points in $\mathcal{R}$ that a mobile ground robot can traverse. Let the curve's extreme points be $\mathbf{x}_0, \mathbf{x}_\infty \in \varphi$. If $\mathbf{x}_\infty = \mathbf{x}_0$, then φ is a *cycle* (our preferred term for a closed curve). In this application, all robots begin seed delivery from a common initial location. As every robot $r \in \mathcal{G}$ upon completion of its assigned task must return to $\mathbf{x}_0$, its curve φ^r must be cyclic. Accordingly, the CPP must satisfy the (*initial*) *location condition*,

$$\forall r \in \mathcal{G}: \mathbf{x}_0 \in \varphi^r. \tag{2}$$

Since the region $\mathcal{R}$ must be totally covered, the following *coverage condition* must be satisfied,

$$\mathcal{R} \subseteq \bigcup_{r \in \mathcal{G}} \mathcal{R}_{\varphi^r}^{\rho}. \tag{3}$$

In the above expression, $\mathcal{R}_{\varphi^r}^{\rho}$ is the region covered by the robot r when traversing the continuous curve φ^r. For any curve φ, the set of points $\mathcal{R}_{\varphi}^{\rho}$ can be expressed in a formal fashion as,

$$\mathcal{R}_{\varphi}^{\rho} = \bigcup_{\mathbf{x}\in\varphi} \mathcal{R}_{\mathbf{x}}^{\rho}. \tag{4}$$

The work w_{φ} required for any robot to traverse the curve φ can be obtained by integrating the infinitesimal work $w_{\mathbf{x},d\mathbf{x}}d\mathbf{x}$ along φ, where $w_{\mathbf{x},d\mathbf{x}}$ is the instantaneous work per unit length in the direction of $d\mathbf{x}$. Since the work needed by the robot to return to $\mathbf{x}_0$ from $\mathbf{x}_{\infty}$ is not necessarily equal to w_{φ}, it must not be viewed as a manifold distance in $\mathcal{R}$.

It follows from soil isotropy that $w_{\mathbf{x},d\mathbf{x}}$ depends only upon the robot's instantaneous speed at $\mathbf{x}$ and the slope θ, which is the elevation gradient from $\mathbf{x}$ to $\mathbf{x} + d\mathbf{x}$. As will be addressed later in Chapter 6 - , the robot's optimal speed can be obtained from θ. Ignoring speed, $w_{\mathbf{x},d\mathbf{x}}$ can be rewritten as w_{θ}. Hence,

$$w_{\varphi} = \oint_{\varphi} w_{\theta}\, d\mathbf{x}. \tag{5}$$

Each robot has a maximum energy *capacity*, $w^{\max}$ in the form of on-board batteries. As each robot must return to its initial location $\mathbf{x}_0$ before expending its available energy, the trajectories taken by the robots must comply with the *capacity condition* shown below,

$$\forall r \in \mathcal{G}: w_{\varphi^r} \leq w^{\max}. \tag{6}$$

Allocating the task between the robots in an even manner is a desirable feature in many applications as it prevents encumbering a few robots with disproportionately larger shares of the task. Not only would each onboard equipment be subject to similar wear and tear, but the total time would also be greatly reduced. The *balance tour condition* is,

$$\forall r \in \mathcal{G}: |\varphi^r| \geq \frac{\gamma}{|\mathcal{G}|}. \tag{7}$$

Here $|\varphi^r|$ is the curve length and $|\mathcal{G}|$, the cardinality of the set $\mathcal{G}$ tht is the number of robots. Each robot must move through a distance of at least $|\mathcal{G}|^{-1}\gamma$. The parameter γ can be expressed in distance units; in a later section the balance condition in Eqn. (7) is recast so as to render γ unitless. The *objective function* $\Omega_{\mathcal{R}}$ that must be minimized is the total energy required for seeding region $\mathcal{R}$, which is the sum of the work done by each robot. Hence,

$$\Omega_{\mathcal{R}} \triangleq \sum_{r\in\mathcal{G}} w_{\varphi^r}. \tag{8}$$

Given a region $\mathcal{R}$, a set $\mathcal{G}$ of identical, mobile ground robots with coverage radii ρ and energy (battery) capacities $w^{\max}$, as well as a geometry dependent work measure w, we can completely

define an instance of the problem as the 6-tuple $(\mathcal{R}, \mathcal{G}, \rho, w^{\max}, \gamma, w_{\varphi})$. The overarching aim of this research can be formulated in terms of the following constrained optimization problem.

CPP-1:

Minimize: $\Omega_{\mathcal{R}}$ in Eqn. (8);

w.r.t.: $\varphi^r, r \in \mathcal{G}$.

Subject to: (2), (3), (6), (7).

Lattice Framework

For computational purposes, the region $\mathcal{R}$ is discretized as a set $\mathcal{L}$ of sample points, referred to as a *lattice*. These points are uniformly spaced at (small) intervals of d_0 so that the points adjacent to any interior point $\mathbf{x} \equiv [x, y, z]^{\mathrm{T}}$ have coordinates of the form $[x \pm d_0, y, z']^{\mathrm{T}}$ and $[x, y \pm d_0, z']^{\mathrm{T}}$, where z' is any elevation. The spacing d_0 is the lattice *granularity*. Since $|\mathcal{L}| < \infty$, the points in the lattice can be uniquely indexed in any suitable manner. The set theoretic relationship $\mathbf{x}_i \in \mathcal{L}$ can be expressed more concisely as $i \in \mathcal{L}$.

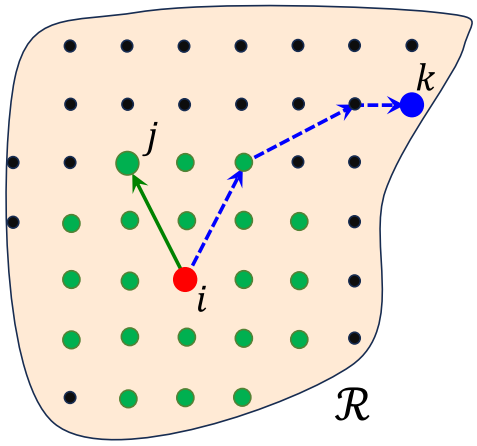


Figure 1. Lattice and Paths.

Circles represent lattice points and $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ are three such points. Points in $\boldsymbol{\mathcal{L}_i}$ with $\boldsymbol{\xi} \in \left[\sqrt{\mathbf{5}}\boldsymbol{d_0}, \sqrt{\mathbf{8}}\boldsymbol{d_0}\right]$ are shown as slightly larger green circles. The direct minimum work path from $\boldsymbol{i}$ to $\boldsymbol{j} \in \boldsymbol{\mathcal{L}_i}$ (solid green) and the indirect trajectory from $\boldsymbol{i}$ to $\boldsymbol{k} \notin \boldsymbol{\mathcal{L}_i}$ (dashed blue) are shown.

Suppose $i, j \in \mathcal{L}$ are any two points with the coordinates $\mathbf{x}_i \equiv [x_i, y_i, z_i]^{\mathrm{T}}$ and $\mathbf{x}_j \equiv [x_j, y_j, z_j]^{\mathrm{T}}$, then the Euclidean distance between their horizontal locations is denoted as $d_{i,j}$ and the slope from i to j, as $\theta_{i,j}$. Whence,

$$\begin{cases} d_{i,j} = ((x_i - x_j)^2 + (y_i - y_j)^2)^{\frac{1}{2}} \\ \theta_{i \to j} = \tan^{-1} \dfrac{z_j - z_i}{d_{i,j}} \end{cases}. \tag{9}$$

It is clear that $d_{j,i} = d_{i,j}$ and $\theta_{j \to i} = -\theta_{i \to j}$.

Every pair of points $\boldsymbol{i, j} \in \boldsymbol{\mathcal{L}}$ that are within a distance parameter $\boldsymbol{\xi}$ are directly connected by an edge. Let $\boldsymbol{\mathcal{L}_i} \subset \boldsymbol{\mathcal{L}}$ be the subset of points in $\boldsymbol{\mathcal{L}}$ that are connected directly to $\boldsymbol{i}$ as illustrated graphically in Figure 1. **Hence $\boldsymbol{\mathcal{L}_i}$ can be defined as,**

$$\mathcal{L}_i \triangleq \{j \in \mathcal{L} | d_{i,j} \leq \xi\}. \tag{10}$$

Henceforth, the lattice $\mathcal{L}$ is treated as fully connected and that it *maximally covers* $\mathcal{R}$, so that no other point in $\mathcal{R}$ can be inserted into $\mathcal{L}$ without violating the uniform spacing arrangement. In other words, the fully connected, uniformly spaced lattice $\mathcal{L}$ maximally covers $\mathcal{R}$ if and only if,

$$\max_{\mathbf{x} \in \mathcal{R}} \min_{i \in \mathcal{L}} \{\mathbf{x} \in \mathcal{R} | (x - x_i)^2 + (y - x_i)^2 \leq \xi\}. \tag{11}$$

In the same manner as in Eqn. (10), the *ρ-neighborhood* $\mathcal{N}_i^{\rho}$ of some $i \in \mathcal{L}$ is defined as the set of all points in $\mathcal{L}$ that are within some distance ρ away from i , i.e.

$$\mathcal{N}_i^{\rho} \triangleq \left\{j \in \mathcal{L} | d_{i,j} \leq \rho\right\}. \tag{12}$$

Since $d_{j,i} = d_{i,j}$, it is evident that $j \in \mathcal{N}_i^{\rho} \Leftrightarrow i \in \mathcal{N}_j^{\rho}$.

It should be noted that in practice, ρ is significantly greater that ξ so that $\mathcal{L}_i \subset \mathcal{N}_i^{\rho}$. Moreover, although both parameters, ξ and ρ are distance thresholds, ρ is a physical quantity that is intrinsic to the robot; specifically, it is the furthest distance the robot can reach to spray seeds. On the other hand, ξ is determined purely from computational considerations. It determines the connectivity of $\mathcal{L}$; lower values of ξ imply less computational load, while also yielding coarser approximations of the proposed approach.

Any point in $\mathcal{L}$ can serve as a *waypoint*. The set of waypoints $\mathcal{C}$ ($\mathcal{C} \subset \mathcal{L}$), is constructed so that points in $\mathcal{L} \backslash \mathcal{C}$ lie within at least one waypoint's ρ-neighborhood. Hence,

$$\mathcal{C} \equiv \left\{k \in \mathcal{L} | \forall i \in \mathcal{L}: \ \left(d_{i,k} = \min_{k' \in \mathcal{C}} d_{i,k'}\right) \Rightarrow d_{i,k} \leq \rho\right\}. \tag{13}$$

In one possible sensor network realization, waypoints may serve as sensor locations so that the steps needed to find optimal robot tours can be implemented in a fully distributed manner. Sensors must be able to communicate across distances of $2\rho + \sqrt{2}d_0$ or more. Alternately, if the region is small enough a sensor may be placed at each point in $\mathcal{L}$ so that waypoint identification can also proceed using distributed message passing. Further details of such a realization are not addressed

in this research. Chapter 3 - , which provides more details of the message passing method, does not address network realization that is deferred to future research.

Problem Approximation

Suppose i and j are a pair of connected points ($j \in \mathcal{L}_i$), then there is a direct path from i to j. Let us assume that the work $w_{i\to j}$ required for a robot to go from i to j is available. An algorithm to estimate the minimum work is described in the next section. The points in $\mathcal{L}$ can be treated as the set of vertices of a graph. The latter's edges are all pairs of directly connected points ($i \to j$). The weight $w_{i\to j}$ of the edge $i \to j$ is obtained in terms of the work that would be required by any robot to go from i to j. Since $w_{i\to j} \neq w_{j\to i}$, the lattice $\mathcal{L}$ and the directed edges is seen to be a weighted digraph.

The route taken by any robot in $\mathcal{G}$ is a sequence of trajectories. The term *trajectory* is reserved in this article to refer to the path from one waypoint to another. If the route includes the path from k to l ($k, l \in \mathcal{C}$) such that $j \notin \mathcal{N}_i^{\rho}$, the trajectory must include intermediate points in $\mathcal{L}$. Such a trajectory is represented as $k \rightsquigarrow l \equiv j_0 \to j_1 \to \cdots j_{P-1} \to j_P$, where $k \equiv j_0$ and $l \equiv j_P$. The edges $j_{p-1} \to j_p$ must be in the lattice so that $j_p \in \mathcal{L}_{p-1}$ ($p = 1,2, \dots, P$).

The work $w_{k\rightsquigarrow l}$ needed by a robot to traverse $k \rightsquigarrow l$ is the sum of the work needed to move along its edges,

$$w_{k\rightsquigarrow l} = \sum_{p=0}^{P} w_{j_{p-1}\to j_p}\,. \tag{14}$$

All intermediate lattice points in the trajectory $k \rightsquigarrow l$ are those that minimize the work $w_{k\rightsquigarrow l}$ so that,

$(j_1 \dots j_{P-1}) =$

$$\underset{\substack{Q, j'_1 \dots j'_{Q-1} \\ \forall q: j'_q \in \mathcal{L}_{j'_{q-1}}}}{\operatorname{arginf}} \left(w_{k\to j'_1} + w_{j'_{Q-1}\to l} + \sum_{q=1}^{Q-1} w_{j'_{q-1}\to j'_q} \right). \tag{15}$$

Thus, $k \rightsquigarrow l$ is the minimum work trajectory.

The Dijkstra's shortest paths algorithm is applied to obtain the intermediate lattice points, where the work $w_{i\to j}$ from i to $j \in \mathcal{L}_i$ is treated as an asymmetric 'distance' (Tao et al., 2021). It has been established that shortest paths through discrete sample points in a manifold are effective

approximations of true geodesic distances [cf. (Bose et al., 2011)]. This is the rationale for the use of the shortest paths algorithm. Moreover, an unsupervised manifold learning algorithm has been proposed in (Tenenbaum et al., 2000) that uses the shortest paths algorithm to approximate geodesic distances. As $w_{k\rightsquigarrow l} \neq w_{l\rightsquigarrow k}$, the RHS in Eqn. (15) must not be interpreted as any measure of distance. Nevertheless, it is safe to assume that the minimum work trajectory $k \rightsquigarrow l$ can be approximated in the same manner.

Let the work done by a robot to go from k to l ($k, l \in \mathcal{C}$) along a continuous curve $\varphi_{k\rightsquigarrow l}$ in $\mathcal{R}$ be denoted as $w_{\varphi_{k\rightsquigarrow l}}(\mathbf{x}_k, \mathbf{x}_l)$. Since the piecewise approximate minimum work $w_{k\rightsquigarrow l}$ is defined according to Eqns. (13) and (14), this research rests on the following *minimum work assumption*,

$$w_{k\rightsquigarrow l} \approx \operatorname*{arginf}_{\varphi_{k\rightsquigarrow l}} w_{\varphi_{k\rightsquigarrow l}}(\mathbf{x}_k, \mathbf{x}_l)\,. \tag{16}$$

Since $\mathcal{L}$ maximally covers $\mathcal{R}$, it follows that $\lim_{d_0\to 0} \mathcal{L} = \mathcal{R}$ and $\lim_{d_0\to 0} \mathcal{N}_k^\rho = \mathcal{R}_{\mathbf{x}_k}^\rho$. Whence, the approximation in Eqn. (16) becomes exact, so that $\lim_{d_0\to 0} w_{k\rightsquigarrow l} = \operatorname*{arginf}_{\varphi_{k\rightsquigarrow l}} w_{\varphi_{k\rightsquigarrow l}}(\mathbf{x}_k, \mathbf{x}_l)$, thereby justifying the assumption.

As the minimum work trajectory $k \rightsquigarrow l$ is approximated via a sequence of intermediate points $j_1 \ldots j_{P-1}$ in $\mathcal{L}$, it is assumed that the region in $\mathcal{R}$ covered by a robot moving along the trajectory $k \equiv j_0 \to j_1 \ldots \to j_{P-1} \to j_P \equiv l$ is equal to the region $\mathcal{R}_{\varphi_{k\rightsquigarrow l}}^\rho$. This is the *coverage assumption*, which is also central to this research. The assumption can be stated mathematically as,

$$\bigcup\nolimits_p \mathcal{R}_{j_{p-1}\to j_p}^\rho \approx \mathcal{R}_{\varphi_{k\rightsquigarrow l}}^\rho. \tag{17}$$

As in Eqn. (16), it can be seen that the approximation in Eqn. (17) becomes exact when d_0 approaches zero. This is the rationale for the coverage assumption.

As the route of each robot $r \in \mathcal{G}$ begins and ends at the same point $\mathbf{x}_0$ (or $k_0 \in \mathcal{L}$), it is a *cyclic tour*, which is denoted as $\mathcal{T}^r$. The tour can be seen as a sequence of waypoints, $\mathcal{T}^r \equiv k_0 \rightsquigarrow k_1 \rightsquigarrow \cdots k_{P-1} \rightsquigarrow k_0$, where each $k_p \in \mathcal{C}$ and P is the tour's cardinality $|\mathcal{T}^r|$ (its formal definition postponed until Chapter 5 -). Under these circumstances, Eqn. (2) reduces to the following equivalent location condition,

$$\forall r \in \mathcal{G}\colon k_0 \in \mathcal{T}^r. \tag{18}$$

Similarly, the capacity condition defined before in Eqn. (7) is reformulated in terms of $\mathcal{T}^r$ as,

$$\forall r \in \mathcal{G}\colon w_{\mathcal{T}^r} \leq w^{\max}. \tag{19}$$

Since $\mathcal{L}$ maximally covers $\mathcal{R}$, as long as d_0 is small enough the condition discussed previously in Eqn. (3) can be replaced with $\mathcal{R} \subseteq \bigcup_{r\in\mathcal{G}} \mathcal{R}^{\rho}_{\mathcal{T}^r}$. Instead, we choose to express the approximate coverage condition in terms of the lattice,

$$\mathcal{L} \subseteq \bigcup\nolimits_{r\in\mathcal{G}} \mathcal{L}^{\rho}_{\mathcal{T}^r}. \tag{20}$$

In Eqn. (20), $\mathcal{L}^{\rho}_{\mathcal{T}}$ is the set of lattice points covered by any tour τ, i.e. $\mathcal{L}^{\rho}_{\mathcal{T}} \equiv \mathcal{R}^{\rho}_{\mathcal{T}} \cap \mathcal{L}$.

The balanced tour condition analogous to Eqn. (7) is,

$$\forall r \in \mathcal{G}\colon |\mathcal{T}^r| \geq \frac{|\mathcal{C}|}{|\mathcal{G}|}\gamma. \tag{21}$$

The parameter $\gamma \in [0,1)$ is a preset lower bound of tour cardinality. Casting the coverage condition in terms of $\mathcal{L}$ allows any instance of the CPP problem to be described as the 6-tuple $(\mathcal{L}, \mathcal{G}, \rho, w^{\max}, \gamma, w_{\mathcal{T}})$, where $w_{\mathcal{T}}$ is the work done in $\mathcal{T}$ via points in $\mathcal{L}$.

The objective function $\Omega_{\mathcal{L}}$ to be minimized is,

$$\Omega_{\mathcal{L}} \triangleq \sum\nolimits_{r\in\mathcal{G}} w_{\mathcal{T}^r}. \tag{22}$$

Based on the assumptions discussed in Eqns. (15) and (16), an approximate version of CPP-1 can be formulated in the following manner.

CPP-2:

Minimize: $\Omega_{\mathcal{L}}$ in Eqn. (22);

w.r.t.: $\mathcal{T}^r, r \in \mathcal{G}$.

Subject to: (18), (19), (20), (21).

Chapter 3 - Min-Sum Message Passing in Factor Graph Models

Min-sum message passing algorithm is an attractive method based on Probabilistic Graphical Models that can be used to solve discrete optimization problems (Ravanbakhsh et al., 2014). As it is the base for solving the multi-robot CPP in this research, a brief review of the fundamentals is presented in this chapter.

Factor Graphs Models

Factor graph models (FGMs) are a class of probabilistic graphical models that can be applied to constrained binary optimization problems (Kschischang et al., 2001). The optimum is reached solely by means of distributed message passing between its nodes. The FGM consists of two classes of nodes: *variable nodes* $i \in \mathcal{V}$ and *function nodes* $k \in \mathcal{F}$, where $\mathcal{V}$ and $\mathcal{F}$ are the sets of indices of all variable and function nodes. Each variable node contains a corresponding binary decision variable $c_i \in \{0,1\}$. Each function node is associated with a function $f_k(c_{i\in\mathcal{V}_k}): \{0,1\}^{|\mathcal{V}_k|} \to \mathbb{R}$, where $i \in \mathcal{V}_k \subseteq \mathcal{V}$. The symbol $c_{i\in\mathcal{V}_k}$ that is the argument to $f_k(\cdot)$ represents all decision variables in $\mathcal{V}_k$. The schematic in Figure 2 shows how messages are passed in such an FGM.

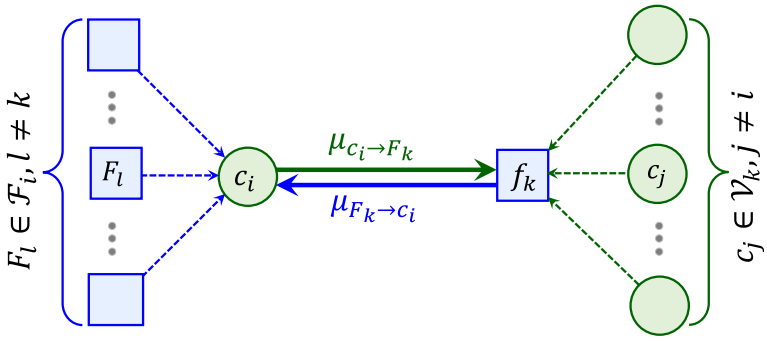


Figure 2. Factor Graph Model.

Solid arrows represent messages $\boldsymbol{\mu_{c_i \to f_k}}$ and $\boldsymbol{\mu_{f_k \to c_i}}$ passed between nodes $\boldsymbol{c_i}$ and $\boldsymbol{f_k}$. Dashed arrows correspond to messages involved in computing $\boldsymbol{\mu_{c_i \to f_k}}$ and $\boldsymbol{\mu_{f_k \to c_i}}$. Only variable nodes in $\boldsymbol{\mathcal{V}_k}$ (green circles) and function nodes in $\boldsymbol{\mathcal{F}_i}$ (blue squares) are shown.

Min-Sum Message Passing Algorithm

Computations in FGMs are carried out entirely through iterative message passing between variable and function nodes. All such messages are represented using the symbol μ, with appropriate

arguments and with subscripts indicating direction of message flow. The outgoing message from variable node $c_i \in \mathcal{V}$ to function node $F_k \in \mathcal{F}_i$ where $\mathcal{F}_i \subseteq \mathcal{F}$ is obtained as the sum of all incoming messages to c_i excluding that from F_k,

$$\mu_{c_i \to F_k}(c_i) = \sum_{F_l \in \mathcal{F}_i \setminus k} \mu_{F_l \to c_i}(c_i)\,. \tag{23}$$

The message from each function node $F_k \in \mathcal{F}$ to its adjoining variable nodes $c_i \in \mathcal{F}_k$ is obtained through tabular binary minimization,

$\mu_{F_k \to c_i}(c_i) =$

$$\min_{c_{j \in \mathcal{V}_k \setminus i}} \left\{ F_k\left(c_{j \in \mathcal{V}_k}\right) + \sum_{c_{j \in \mathcal{V}_k \setminus i}} \mu_{c_j \to F_k}(c_i) \right\}. \tag{24}$$

It has been shown that such message passing algorithms converges to exact solutions in polytrees and provide reasonable approximations in generic digraphs (Murphy et al., n.d.).

In the present seeding application, it suffices for the nodes to pass the difference between $\mu(1)$ and $\mu(0)$ as scalar messages, so that the actual messages sent and received are given by,

$$\begin{cases} \mu_{c_i \to f_k} = \mu_{c_i \to f_k}(c_i = 1) - \mu_{c_i \to f_k}(c_i = 0) \\ \mu_{f_k \to c_i} = \mu_{f_k \to c_i}(c_i = 1) - \mu_{f_k \to c_i}(c_i = 1) \end{cases}. \tag{25}$$

The quantities in the RHS of Eqn. (25) are obtained from Eqns. (23) and (24).

Iterative *min-sum message passing* of the scalar messages in Eqn. (25) would lead to an optimum $\mathbf{c}^* \in [0,1]^{|\mathcal{V}|}$ given as,

$$[c_1, \dots, c_i, \dots c_{|\mathcal{V}|}]^{\mathrm{T}^*} \triangleq \operatorname*{argmin}_{c_{i \in \mathcal{V}}} \sum_{k \in \mathcal{F}} F_k\left(c_{i \in \mathcal{V}_k}\right). \tag{26}$$

In (Givoni & Frey, 2009), message passing within an underlying FGM has been shown to be equivalent to the distributed exemplar clustering algorithm in (Frey & Dueck, 2007). Such message passing algorithms have been extended in (Ravanbakhsh et al., 2014) to solve a variety of NP-hard combinatorial problems. It has been demonstrated in (Ravanbakhsh et al., 2014) that such algorithms run in polynomial time for large scale instances of the symmetric TSP as well as min-max clustering problems. More recently, in (Dai et al., 2022) min-sum message passing has been applied to the disjoint shortest paths problem in weighted digraphs, where convergence towards the exact optimum has been established.

Chapter 4 - Way-point Selection Using Exemplar Clustering

Previously in Chapter 2 - , the set of waypoints $\mathcal{C}$ ($\mathcal{C} \subset \mathcal{L}$) was formally defined. These waypoints in $\mathcal{C}$ are identified from $\mathcal{L}$ through affinity propagation (Frey & Dueck, 2007), a clustering algorithm based on message passing. The update rules are obtained from the FGM in (Givoni & Frey, 2009). Message passing is restricted to pairs of points i, k that lie in the ρ-neighborhoods of one another. Otherwise, this approach is a straightforward implementation of affinity propagation. Nevertheless, and only for the sake of completeness, the main message passing steps in (Givoni & Frey, 2009) are described below. A schematic of the FGM is shown in Figure 3.

Any point $i \in \mathcal{L}$ can pick the point $k \in \mathcal{L}$ as its waypoint so long as $k \in \mathcal{N}_i^{\rho}$; this relationship denoted as $k = \hbar(i)$. The associated Boolean variable $c_{i,k} \in \mathcal{V}$, where $\mathcal{V}$ is the FGM's set of variable nodes is given as,

$$c_{i,k} \triangleq \begin{cases} 1, & k \in \mathcal{N}_i^{\rho}, k = \hbar(i) \\ 0, & k \in \mathcal{N}_i^{\rho}, k \neq \hbar(i) \\ \text{undefined}, & k \notin \mathcal{N}_i^{\rho} \end{cases}. \tag{27}$$

There are three types of function nodes, represented as $f_{i,k}^{D}(\cdot) \in \mathcal{F}_D$, $f_i^{I}(\cdot) \in \mathcal{F}_I$, and $f_k^{E}(\cdot) \in \mathcal{F}_E$, with the maps $f_{i,k}^{D}: [0,1] \to \mathbb{R}$, $f_i^{I}: [0,1]^{|\mathcal{N}_i^{\rho}|} \to \mathbb{R}$, and $f_k^{E}: [0,1]^{|\mathcal{N}_k^{\rho}|} \to \mathbb{R}$. Function nodes in $\mathcal{F}_D$ make up the objective function of the message passing algorithm. Function nodes in $\mathcal{F}_I$ and $\mathcal{F}_E$ ensure convergence towards valid solutions. The set of all function nodes in the FGM is $\mathcal{F} = \mathcal{F}_D \cup \mathcal{F}_I \cup \mathcal{F}_E$.

For every pair i, k, the function node $f_{i,k}^{D}(c_{i,k})$ is defined as,

$$f_{i,k}^{D}(c_{i,k}) \triangleq \begin{cases} 0, & i \neq k, c_{i,k} = 0 \\ w_{i \to k}, & i \neq k, c_{i,k} = 1 \\ p_k, & i = k \end{cases}. \tag{28}$$

The quantity p_k in Eqn. (28) is the *preference* of point k. In an approximate sense, p_k reflects how well suited k is to be a waypoint in $\mathcal{C}$. The vector of all such preferences, denoted as $\mathbf{p} = [p_k]_{k \in \mathcal{L}} \in \mathbb{R}^{|\mathcal{L}| \times 1}$, may be fixed at $\mathbf{p} = p_0 \mathbf{1}_{|\mathcal{L}|}$, where p_0 is some constant. However, as described in the next section, the preference vector may also be iteratively incremented to further fine-tune an existing solution.

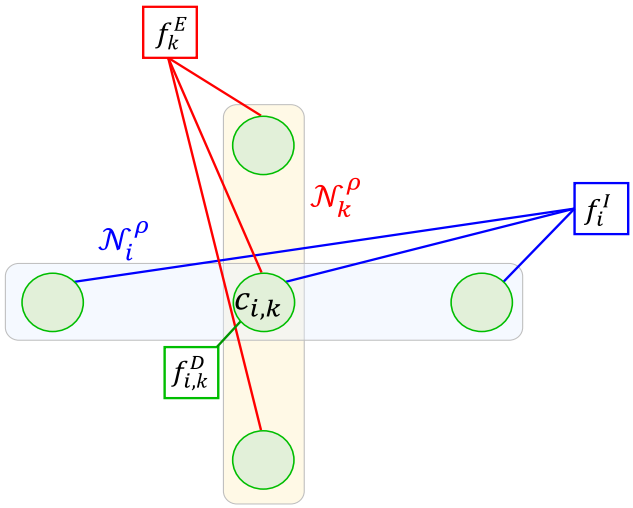


Figure 3. Waypoint Selection FGM.

Variable nodes are shown as green circles. The variable node $\boldsymbol{c_{i,k}}$ with the three function nodes associated to it are shown as squares. The larger shaded rectangular boxes represent the $\boldsymbol{\rho}$-neighborhoods of the points $\boldsymbol{i}$ and $\boldsymbol{k}$.

Function nodes in $\mathcal{F}_I$ impose the *1-in-N consistency condition* (where 'N' is the cardinality of a ρ-neighborhood), which is the requirement that every point $i \in \mathcal{L}$ must select exactly one point $k \in \mathcal{N}_i^\rho$ as its waypoint $\hbar(i)$. The function $f_i^I(\cdot)$ that does so is,

$$f_i^I\left(c_{i,k\in\mathcal{N}_i^\rho}\right) \triangleq \begin{cases} \infty, & \sum_{k\in\mathcal{N}_i^\rho} c_{i,k} \neq 1 \\ 0, & \text{otherwise} \end{cases}. \tag{29}$$

Function nodes in $\mathcal{F}_E$ are included in the FGM in order to satisfy the *waypoint consistency condition*, that is, any point $k \in \mathcal{L}$ can be the waypoint for other points in its ρ-neighborhood if and only if it is its own waypoint. Accordingly, $f_k^E(\cdot)$ is defined in the manner below,

$$f_k^E\left(c_{i\in\mathcal{N}_k^\rho,k}\right) \triangleq \begin{cases} \infty, & c_{k,k} = 0, \sum_{i\in\mathcal{N}_k^\rho} c_{i,k} = 1 \\ 0, & \text{otherwise} \end{cases}. \tag{30}$$

It can be seen from Eqns. (28) and (29) that any invalid assignment of variables in $\mathcal{V}$ would make some $f_i^I(\cdot) = \infty$ or $f_k^E(\cdot) = \infty$, so that the sum $\sum_{f\in\mathcal{F}} f(\cdot) = \infty$. Clearly an assignment that violated a consistency condition is not a minimum as defined in Eqn. (11). The min-sum message passing algorithm, implemented as shown earlier in Eqns. (23), and (24), leads away from such invalid assignments.

Upon convergence of min-sum message-passing, the set of waypoints can be obtained as,

$$\mathcal{C} \leftarrow \{k \in \mathcal{L} | c_{k,k} = 1\}. \tag{31}$$

As message passing is limited to nodes that are in the ρ-neighborhoods of one another, it is obvious that each point i is assigned waypoint $\hbar(i) \in \mathcal{N}_i^{\rho}$ where $\mathcal{N}_i^{\rho}$ is defined in Eqn. (11). In other words, $\mathcal{C}$ is such that for every $i \in \mathcal{L}$, the inequality $d_{i,\hbar(i)} \leq \rho$ is met. It immediately follows that $\mathcal{C}$ satisfies the neighborhood criterion defined in Eqn. (12).

The min-sum message-passing algorithm should ideally converge to a global minimum of $\sum_{i,k\in\mathcal{V}} f_{i,k}^{D}(c_{i,k})$, allowing the optimal set of waypoints to be defined as,

$$\mathcal{C}^* \triangleq \operatorname*{arginf}_{\mathcal{C}} \sum_{i\in\mathcal{L}} d_{i,\hbar(i)} \,. \tag{32}$$

However due to the presence of cycles in the underlying FGM, there are no strong theoretical guarantees that the set $\mathcal{C}$ of waypoints obtained by Eqn. (30) coincides with $\mathcal{C}^*$. Consequently, waypoint selection should be viewed as a heuristic algorithm. The lattice point k_0, which is the initial and terminal location of all robots in $\mathcal{G}$, is inserted into $\mathcal{C}$.

Chapter 5 - Multi-robot Cyclic Path Planning problem

With only one robot ($|\mathcal{G}| = 1$), the proposed message passing algorithm described here can be regarded as an extension of the polynomial time algorithm used in (Ravanbakhsh et al., 2014) to obtain cyclic tours in symmetric graphs, where the Held-Karp (HK) constraints (Held & Karp, 1970) are used to ensure tour validity. In this research, the HK constraints are generalized and extended to be useful in weighted digraphs as well as with multiple robots ($|\mathcal{G}| > 1$). Next, additional constraints are incorporated to satisfy capacity condition in Eqn. (18). The FGM that implements these constraints is described next.

A robot's cyclic tour is must only include trajectories between waypoints that are within mutual distances of ζ. The ζ-*neighborhood* of any waypoint k is defined accordingly,

$$\mathcal{C}_k^{\zeta} \triangleq \{l \in \mathcal{C}\backslash\{k\} | d_{k\rightsquigarrow l} \leq \zeta\}. \tag{33}$$

The algorithmic parameter ζ in Eqn. (33) is such that $\zeta > \rho$. When ζ is large enough, every pair of waypoints are neighbors, i.e., for any waypoint k, $\mathcal{C}_k^{\zeta} = \mathcal{C}$. On the other hand, a value to ζ that is close to unity helps reduce memory requirements, with little effect on the outcome's optimality. The FGM is constructed from an underlying digraph whose vertices are the waypoints in $\mathcal{C}$ and whose directed edge connect each waypoint k to every other waypoint in its ζ-neighborhood. When ζ is large enough (or unused), the sets $\mathcal{E}_k^-$ and $\mathcal{E}_k^+$ of incoming and outgoing edges to/from k are,

$$\begin{cases} \mathcal{E}_k^- \triangleq \{l \rightsquigarrow k | k, l \in \mathcal{C}, k \neq l\} \\ \mathcal{E}_k^+ \triangleq \{k \rightsquigarrow l | k, l \in \mathcal{C}, k \neq l\} \end{cases}. \tag{34}$$

Each trajectory $l \rightsquigarrow k$ in Eqn. (34) is an incoming edge to k and each $k \rightsquigarrow l$ is an outgoing edge from it.

Cyclic tours were discussed in Chapter 2 - . Such a cyclic tour $\mathcal{T} \equiv k_0 \rightsquigarrow k_1 \rightsquigarrow \cdots k_{P-1} \rightsquigarrow k_0$ where $k_0, \ldots k_{P-1} \in \mathcal{C}$ and each $k_{p-1} \rightsquigarrow k_p$ is a minimum work path. The tour can be viewed as an ordered subset of the set of all incoming edges or all outgoing in the underlying digraph. With $k_P \equiv k_0$, we express this formally as,

$$\mathcal{T} \triangleq \begin{cases} \left\{k_{p-1} \rightsquigarrow k_p | k_p \in \mathcal{E}_{k_{p-1}}^+\right\}_{p=1,\ldots,P} & \text{or,} \\ \left\{k_p \rightsquigarrow k_{p+1} | k_{p+1} \in \mathcal{E}_{k_p}^-\right\}_{p=0,\ldots,P-1}. \end{cases} \tag{35}$$

When $|\mathcal{G}| > 1$, the valid solution must include a cyclic tour $\mathcal{T}^r$ for each robot $r \in \mathcal{G}$. The union $\mathcal{T}^{\mathcal{G}}$ of all such tours is,

$$\mathcal{T}^{\mathcal{G}} \triangleq \bigcup_{r \in \mathcal{G}} \mathcal{T}^r . \tag{36}$$

For conciseness, wherever $|\mathcal{G}| = 1$ the symbol $\mathcal{T}$ without superscript is used for the cyclic tour.

Validity Constraints

For $\mathcal{T}^{\mathcal{G}}$ to be valid it must include for each $k \in \mathcal{C}$, only one incoming edge as well as only one outgoing edge. The initial point k_0 is the only exception to this rule. Since k_0 is common to all robot tours, it must have $|\mathcal{G}|$ outgoing edges as well as $|\mathcal{G}|$ incoming edges. We formalize these observations in terms of the *in-degree constraints* and the *out-degree constraints* for each $k \in \mathcal{C}$ that are expressed as below,

$$\begin{cases} \forall k \in \mathcal{C} \backslash k_0 : \begin{cases} \left|\mathcal{T}^{\mathcal{G}} \cap \mathcal{E}_k^-\right| = 1, \\ \left|\mathcal{T}^{\mathcal{G}} \cap \mathcal{E}_k^+\right| = 1; \end{cases} \\ \left|\mathcal{T}^{\mathcal{G}} \cap \mathcal{E}_{k_0}^-\right| = |\mathcal{G}|, \\ \left|\mathcal{T}^{\mathcal{G}} \cap \mathcal{E}_{k_0}^+\right| = |\mathcal{G}|. \end{cases} \tag{37}$$

There are two pairs of degree constraints in Eqn. (37), each consisting of an in-degree and an out-degree constraint. The first pair applies to all waypoints except k_0. The second pair of constraints pertains only to k_0 and is needed for the initial condition in Eqn. (28) to hold.

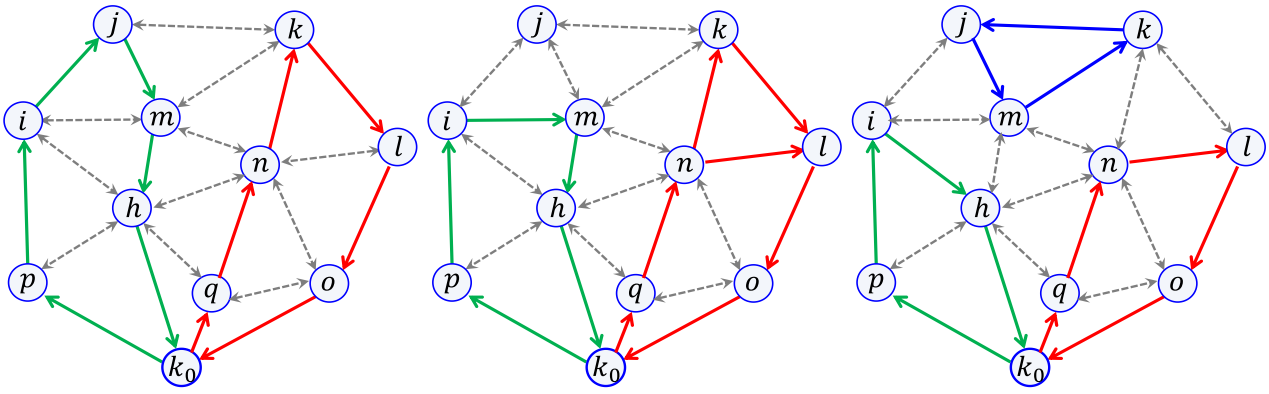


Figure 4. Degree Constraints.

Examples pertain to two robots. Cycles are highlighted using solid arrows, each with a separate color (red, green, blue). Dashed arrows represent potential edges of the tour. The figure shows a valid solution with two tours (left), an invalid solution with degree constraint violations at ***j*** and ***n*** (middle), and another such invalid solution that does not violate any degree constraint (right).

Waypoint k_0's degree constraints simplify to $\left|\mathcal{T}\bigcap\mathcal{E}_{k_0}^{-}\right| = 1$ and $\left|\mathcal{T}\bigcap\mathcal{E}_{k_0}^{+}\right| = 1$. These expressions are identical to those of any other waypoint. Therefore when $|\mathcal{G}| = 1$ only the first pair of degree constraints are required in Eqn. (37) that apply uniformly to all waypoints including k_0. In addition, if the edge directions are ignored, Eqn. (34) reduces to the undirected edge set ($\mathcal{E}_k = \mathcal{E}_k^{-} \cup \mathcal{E}_k^{+}$), the constraints in Eqn. (37) reduce to a single one, $|\mathcal{T}\bigcap\mathcal{E}_k| = 1$, which is the original HK degree constraint in (Held & Karp, 1970) that is used in (Ravanbakhsh et al., 2014).

Figure 4 illustrates the use of degree constraints with $|\mathcal{G}| = 2$. Figure 4 (left) shows two valid tours satisfying all degree constraints. Figure 4 (middle) shows invalid tours, $\mathcal{T}^1 = k_0 \rightsquigarrow p \rightsquigarrow i \rightsquigarrow m \rightsquigarrow h \rightsquigarrow k_0$ and $\mathcal{T}^2 = k_0 \rightsquigarrow q \rightsquigarrow n \rightsquigarrow k \rightsquigarrow l \rightsquigarrow o \rightsquigarrow k_0$. Waypoint j is not included in either tour producing two constraints violations $\left|\mathcal{T}^1 \cup \mathcal{T}^2\bigcap\mathcal{E}_j^{-}\right|, \left|\mathcal{T}^1 \cup \mathcal{T}^2\bigcap\mathcal{E}_j^{+}\right| = 0$. The presence in $\mathcal{T}^2$ of two outgoing edges from waypoint n yields another constraint violation $|\mathcal{T}^1 \cup \mathcal{T}^2\bigcap\mathcal{E}_n^{+}| = 2$.

Although needed to ensure tour validity, the degree constraints are not sufficient. Figure 4 (right) shows the invalid tours $\mathcal{T}^1$ and $\mathcal{T}^2$ of two robots that do not violate any degree constraint. By themselves both $\mathcal{T}^1 = k_0 \rightsquigarrow p \rightsquigarrow i \rightsquigarrow h \rightsquigarrow k_0$ and $\mathcal{T}^2 = k_0 \rightsquigarrow q \rightsquigarrow n \rightsquigarrow o \rightsquigarrow k_0$ are valid cyclic tours. However they do not provide full coverage due the presence of the *subtour* $\mathcal{S} = m \rightsquigarrow j \rightsquigarrow k \rightsquigarrow m$ that does not include k_0.

Additional constraints are required to identify invalid multirobot tours that do not violate degree constraints. The HK subtour constraint is a viable option only for single cyclic tour in undirected graphs. It must be applied to several instances whose total grows exponentially with the size of the graph. Fortunately, the number of instances can be significantly reduced to be of cubic order (Ravanbakhsh et al., 2014).

Since our underlying graph contains directed edges and in view of the balance condition in Eqn. (21), an alternate approach is proposed in this research. To develop the present approach, the definitions of the sets of incoming/outgoing edges of any waypoint in $\mathcal{C}$ in Eqn. (34) are generalized to subsets of $\mathcal{C}$. Let $\mathcal{S} \subset \mathcal{C}$ be a proper subset of waypoints. The set of all incoming edges to $\mathcal{S}$ and the set of all outgoing edges from it are defined in the manner below,

$$\begin{cases}\mathcal{E}_{\mathcal{S}}^{-} \triangleq \{l \rightsquigarrow k | l \in \mathcal{C}\backslash\mathcal{S}, k \in \mathcal{S}\} \\ \mathcal{E}_{\mathcal{S}}^{+} \triangleq \{k \rightsquigarrow l | l \in \mathcal{C}\backslash\mathcal{S}, k \in \mathcal{S}\}\end{cases}. \tag{38}$$

As the next constraints are meant to prevent the formation of small subtours, we focus on only subsets $\mathcal{S}$ with not more than $|\mathcal{G}|^{-1}|\mathcal{C}|\gamma$ waypoints. For reasons that are explained later, the

constraints need only apply to subsets $\mathcal{S}$ that contain k_0. Since k_0 is common to all robot tours, every robot in $\mathcal{G}$ must exit $\mathcal{S}$ at least once, $\mathcal{T}^{\mathcal{G}}$ must have at least $|\mathcal{G}|$ edges in $\mathcal{E}_{\mathcal{S}}^{+}$. If all robot trajectories in $\mathcal{T}^{\mathcal{G}}$ were cyclic then every robot must also reenter $\mathcal{S}$, must also have an equal number of edges in $\mathcal{E}_{\mathcal{S}}^{-}$. However, as the message passing algorithm is not initialized to only cyclic tours, at an intermediate stage, the possibility that $\mathcal{T}^{\mathcal{G}}$ contains acyclic tours must also be taken into account. To maintain validity, $\mathcal{T}^{\mathcal{G}}$ must meet the *balanced subtour constraint* below,

$$|\mathcal{S}| \leq \frac{|\mathcal{C}|}{|\mathcal{G}|}\gamma, k_0 \in \mathcal{S}\colon \left|\mathcal{T}^{\mathcal{G}} \cap \mathcal{E}_{\mathcal{S}}^{-}\right| + \left|\mathcal{T}^{\mathcal{G}} \cap \mathcal{E}_{\mathcal{S}}^{+}\right| \geq 2|\mathcal{G}| - 1. \tag{39}$$

The degree constraints in Eqn. (37) and the balanced subtour constraint in Eqn. (39) are necessary and sufficient for $\mathcal{T}^{\mathcal{G}}$ to be the union of $|\mathcal{G}|$ cyclic subtours that satisfy Eqn. (21)

Chapter 6 - Work Optimal Path Estimation

Estimation Steps

As there exists a directed edge from every point $i \in \mathcal{L}$ to all other points $j \in \mathcal{L}_i$, the minimum work $w_{i\to j}$ required by a robot was estimated with the help of two separate support vector regression (SVR) models. The SVRs shared a common input space in $\mathbb{R}^3$ that comprised of the following three fields: (*i*) the mean slope θ ($\theta \equiv \theta_{i,j}$), (*ii*) the robot's speed v, and (*iii*) the drawbar pull F, which is equal to the force that would be required by a robot to traverse the path $i \to j$ at speed v. The scalar outputs of the models were (*i*) the travel reduction ratio γ, and (*ii*) the power number ν. Physical meanings of all these quantities, which are used to estimate $w_{i\to j}$, can be found in (C. M. Badgujar et al., 2022).

Although γ and ν are uniquely determined from θ, v and F, due to the presence of physical nonlinearities, deriving exact relationships between them is impossible. This is the reason for using nonlinear SVRs to accomplish approximate relationships,

$$\begin{cases} \gamma \approx m_\gamma(\theta, v, F) \\ \nu \approx m_\nu(\theta, v, F) \end{cases}. \tag{40}$$

Training data was obtained from extensive laboratory experiments. The setup involved one of the robots designed for seed dispersal in landscapes characterized by steep slopes. Samples were obtained for various slopes and drawbar pulls, and recording the resulting speeds, travel reduction ratio, and power numbers.

For every pair of lattice points $i, j \in \mathcal{L}$ such that $j \in \mathcal{L}_i$, the horizontal distance $d_{i,j}$ and the slope $\theta_{i,j}$ can be obtained as per Eqn. (9). The optimal work needed by the robot to traverse a unit distance along the path $i \to j$ can be obtained by minimizing the travel reduction ratio with respect to the speed and drawbar pull. In this research, the minimization was carried out iteratively by applying sequential quadratic programming to the first regression model. The optimal speed $v^*_{i\to j}$ and drawbar pull $F^*_{i\to j}$ obtained through this optimization are determined as follows,

$$\left[v^*_{i\to j}, F^*_{i\to j}\right] = \underset{v,F}{\operatorname{arginf}}\, m_\gamma\left(\theta_{i\to j}, v, F\right). \tag{41}$$

The corresponding power number $\nu^*_{i\to j}$ was directly obtained from the next regression,

$$\nu^*_{i\to j} = m_\nu\left(\theta_{i\to j}, v^*_{i\to j}, F^*_{i\to j}\right). \tag{42}$$

The robot's power is the product of its mass M and its instantaneous speed. Under the assumption that the speed remains constant throughout $i \to j$, the minimum work needed $w^*_{i\to j}$ by the robot is,

$$w^*_{i\to j} = M d_{i,j} v^*_{i\to j}. \tag{43}$$

In Eqn. (14) and Eqn. (15), all quantities of the form $w_{i\to j}$ are, in fact, the minimum work $w^*_{i\to j}$ of Eqn. (43). As the two experimental robots available with this article's authors can change their orientations very slowly, the work needed by them to reorient were deemed negligible, and not incorporated in obtaining the shortest trajectories in Eqn. (15).

In the 6-tuple $(\mathcal{L}, \mathcal{G}, \rho, w^{\max}, \gamma, w_{\mathcal{T}})$, the lattice $\mathcal{L}$, the set of robots $\mathcal{G}$, the coverage distance ρ and battery capacity $w^{\max}$ are defined from geographic data and robot mechanics. The parameter γ is also strongly related to the specific cropping task and the available robots. Only some method $w_{\mathcal{T}}$ to estimate minimum work trajectories is needed to fully define an instance of CPP-2. The work optimization method along with the shortest paths algorithm that was described earlier, provides this component of a problem instance.

Chapter 7 - Refinement Using Reinforcement Learning

Waypoint identification and tour optimization are the basic steps of the proposed approach. Due to the presence of cycles in the underlying FGMs of the message passing framework, the output although work optimal, is not guaranteed to represent the global minimum. Ergo, there is the scope to fine-tune the solution for further improvement which we address next. Note that although the basic approach can be deployed online through a distributed network of sensors, the algorithm described here is meant for centralized processing. As discussed previously in Chapter 1 - , an offline implementation of the basic steps has its advantages.

Refining the tours in $\mathcal{T}^{\mathcal{G}}$ uses a form of reinforcement learning. It involves iteratively updating the preference vector $\mathbf{p} = [p_i]_{i\in\mathcal{L}} \in \mathbb{R}^{|\mathcal{L}|\times 1}$. This vector is incremented at the end of every *epoch* i.e. a learning step. At the end of epoch n, the preference vector is represented as $\mathbf{p}^{(n)}$. All scalar preferences are initialized to the same value p_0,

$$\mathbf{p}^{(0)} = p_0\mathbf{1}_{|\mathcal{L}|}. \tag{44}$$

At the beginning of epoch n, a set $\mathcal{S}^{(n)}$ of training samples Each sample consists of a preference vector $\mathbf{p}_s^{(n)}$, a set of waypoints $\mathcal{C}_s^{(n)}$, and a numerical value of the objective defined in Eqn. (27). Ignoring (for simplicity) the subscript $\mathcal{L}$, we denote this objective as $\Omega_s^{(n)}$. Accordingly, $\mathcal{S}^{(n)} = \left\{\mathbf{p}_s^{(n)}, \mathcal{C}_s^{(n)}, \Omega_s^{(n)}\right\}_{s=1,\ldots,S}$. The sample size S remains constant throughout the learning process.

The preference vector at the beginning of epoch n is $\mathbf{p}^{(n-1)}$ from the previous epoch. Each sample indexed s is obtained in the following manner. First, a perturbation is applied to every scalar element in $\mathbf{p}^{(n-1)}$ to get the corresponding value in $\mathbf{p}_s^{(n)}$,

$$p_{i,s}^{(n)} = p_i^{(n-1)} + U(-\varepsilon_p, +\varepsilon_p). \tag{45}$$

The term $U(\cdot)$ represents a uniformly distributed random number lying between the bounds specified in its arguments.

Randomization is the exploratory component of our learning method. Earlier experiments (not reported here) showed that applying normally distributed perturbations were unable to equip the learning algorithm with enough exploration. The preferences of unexplored lattice points would not receive enough perturbation for the next steps of the epoch. Uniformly distributed random perturbations as in Eqn. (44) resulted in a significant overall improvement.

The set of waypoints $\mathcal{C}_s^{(n)}$ is obtained from $\mathbf{p}_s^{(n)}$. The waypoint selection method that was described in Chapter 4 - is used to identify waypoints. The higher the preference $p_{i,s}^{(n)}$ of the lattice point $i \in \mathcal{L}$ the more likely it is to become a waypoint. This is because the preference of a lattice point is an indicator of its suitability to serve as a waypoint.

Optimal cyclic tours of the robots in $\mathcal{G}$ are determined as sequences of the waypoints in $\mathcal{C}_s^{(n)}$. The method outlined previously in Chapter 5 - is used for this purpose. The objective value $\Omega_s^{(n)}$ is the sum of the tours' work requirement.

All S samples in $\mathcal{S}^{(n)}$ are obtained in this manner. It is important to note that as waypoint selection and multi-robot cyclic tours optimization strictly involve deterministic message passing steps, $\Omega_s^{(n)}$ is dependent only on the preference vector $\mathbf{p}_s^{(n)}$. Whereupon it can be seen that $p_{i,s}^{(n)}$ reflects how suitable it is to route robots through i.

If the inclusion of the lattice point i in a $\mathcal{C}_s^{(n)}$ results in a relatively low $\Omega_s^{(n)}$, then its preference must be increased so that $p_i^{(n)} > p_i^{(n-1)}$. Conversely if point $i \in \mathcal{C}_s^{(n)}$ yields a higher $\Omega_s^{(n)}$ then $p_i^{(n)} < p_i^{(n-1)}$. Only those points that are in any $\mathcal{C}_s^{(n)}$ are incremented. We define the $\mathcal{C}^{(n)}$ to be the union of all $\mathcal{C}_s^{(n)}$,

$$\mathcal{C}^{(n)} \triangleq \bigcup_s \mathcal{C}_s^{(n)}. \tag{46}$$

With $\overline{\Omega}^{(n)}$ denoting the sample mean, the preference of each $i \in \mathcal{C}_s^{(n)}$ must be updated in proportion to the difference $\overline{\Omega}^{(n)} - \Omega_s^{(n)}$ so that i receives a positive increment if using it as a waypoint yields an objective value $\Omega_s^{(n)}$ that is lower than the sample average $\overline{\Omega}^{(n)}$. The normalized difference is $1 - \frac{\Omega_s^{(n)}}{\overline{\Omega}^{(n)}}$ which can be scaled up by multiplying it with p_0. Averaging $p_0\left(1 - \frac{\Omega_s^{(n)}}{\overline{\Omega}^{(n)}}\right)$ over all S samples is the *target* preference $q_i^{(n)}$,

$$q_i^{(n)} = \frac{1}{S} \sum_{s, i \in \mathcal{C}_s^{(n)}} p_0 \left(1 - \frac{\Omega_s^{(n)}}{\overline{\Omega}^{(n)}}\right). \tag{47}$$

The point's preference is incremented by only a small fraction η in the direction of the target,

$$i \in \mathcal{C}^{(n)}: p_i^{(n)} = p_i^{(n-1)} + \eta\left(q_i^{(n)} - p_i^{(n-1)}\right). \tag{48}$$

The parameter η $(0 < \eta \ll 1)$ is the learning rate of the learning algorithm. The update rule in Eqn. (48) selectively reinforces the preferences of only those lattice points that lower the total work requirement. If a lattice point i is not selected as a waypoint in any $\mathcal{C}_s^{(n)}$ of the epoch's set of samples $\mathcal{S}^{(n)}$, then its preference is not incremented during that epoch (i.e. $p_i^{(n)} = p_i^{(n-1)}$). Figure 5 shows a diagram of this method.

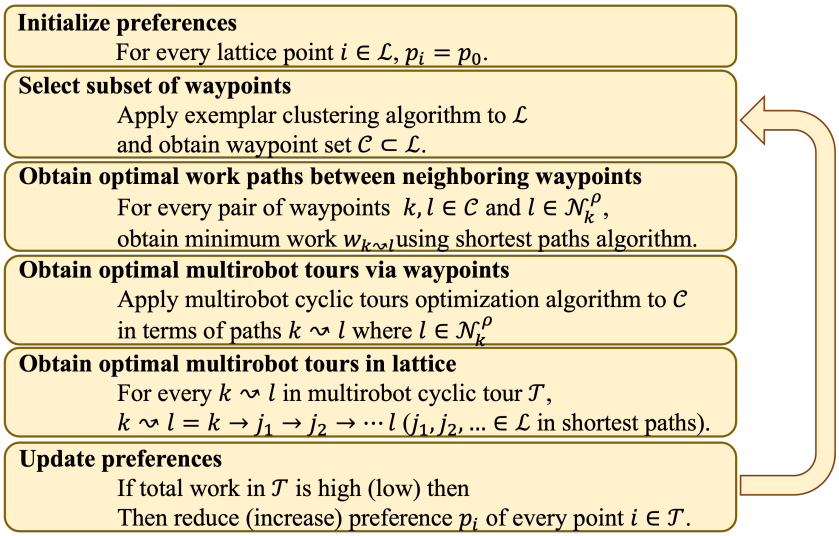


Figure 5. Flowchart of the reinforcement learning process.

In any real-world seeding application, as the robots' coverage radius ρ is usually larger than the lattice granularity d_0 by at least one order of magnitude, only a small fraction of lattice points is selected as waypoints. In order to provide the necessary amount of exploration using Eqn. (45), the reinforcement learning algorithm should go through many epochs of training. This is somewhat wasteful practice, for if the point $i \in \mathcal{L}$ is very suitable to serve as a waypoint, then so should other points in its vicinity. An improvement over the Eqn. (47) is to allow the preferences of lattice points j in the immediate vicinity of i to be incremented. Accordingly,

$$\begin{cases} p_j^{(n)} = p_j^{(n-1)} + \eta_{i,j}\left(q_i^{(n)} - p_j^{(n-1)}\right), \text{ where,} \\ \eta_{i,j} = \eta\left(1 + \dfrac{|z_i - z_j|}{z_0}\right)^{-1} e^{-\frac{d_{i,j}^2}{\sigma}}. \end{cases} \tag{49}$$

Chapter 8 - Observations and outcomes

The results obtained from this research are described in this section. Traction data used to develop the work estimation models (Chapter 6 -) were based on extensive laboratory experiments. Figure 6 shows the setup used in data collection. A total of 1,288 samples were obtained for this purpose. All algorithms in this research were implemented in the MATLAB® programming language.

Figure 6. Experimental Setup.

The laboratory setup for data collection**.**

Work Estimation

After discarding a few experimental outliers, the traction data was randomly divided into 1,032 (80%) training samples, 128 (10%) test samples, and 128 (10%) validation samples. Table I provides the upper and lower limits of the five fields used for work estimation (Section IV.A).

Table I

Field	symbol	θ	v	F	γ	ν
	name	slope	speed	drawbar pull	tr. redn. ratio	power no.
	unit	degrees	meters per min	Newtons	-	-
Limit	lower	-18°	1 m/m	0 N	0.02	0.77
	upper	+18°	5 m/m	1500 N	1.00	20.00

Regression models were developed to implement the function mappings $m_\gamma(\cdot)$ and $m_\nu(\cdot)$ in Eqn. (40). The data used to train the models was obtained from A first set of experiments with several regression models showed that neural networks were able to outperform the others in terms

of performance metrics. Unfortunately, for a few input values the norms of the gradients $\nabla m_\gamma(\cdot)$ and $\nabla m_\nu(\cdot)$ were unacceptably high. Moreover, an analysis of the Hessian $\nabla^2 m_\gamma(\cdot)$, indicated the presence of local minima, thereby rendering neural networks unsuitable for numerical optimization in Eqn. (41). Various methods of regularization were ineffective to alleviate this problem.

Support vector regression (SVR) models [cf.(F. Zhang & O'Donnell, 2020)] were determined to be the most appropriate choice for this study. Two SVRs with Gaussian kernels were trained with the ε-loss function and LASSO weight regularization. All SVR training parameters were set to their default values provided in the MATLAB Machine Learning toolbox.

Preliminary Analysis

The purpose of the first study was to assess the computational time requirement of the message passing algorithms for waypoint selection (Chapter 4 -) and multi-robot cyclic tours optimization (Chapter 5 -). Instead of using regularly placed lattice points, a large number of uniformly distributed random points in two dimensional were generated in the square shaped region $[1, 21]^2$. Figure 7 shows how the execution time increases with sample size. It is significant to note that the message passing approach to obtain optimal cyclic tours from these points best fitted a polynomial time regression model.

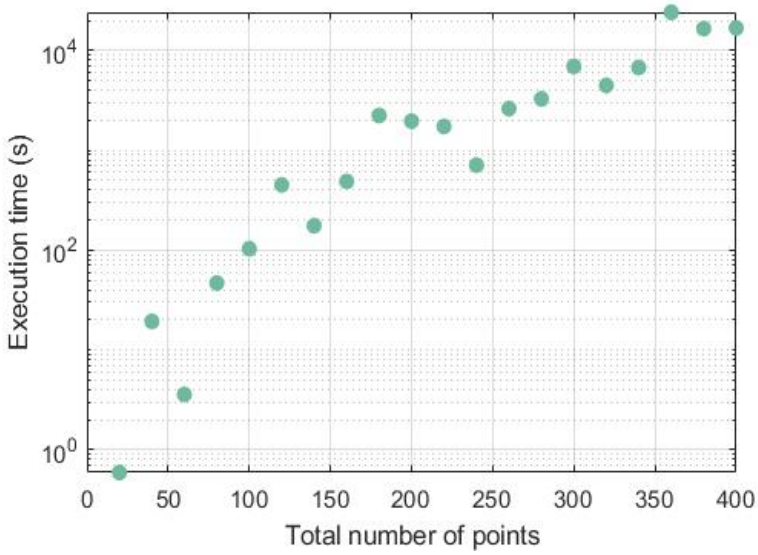


Figure 7. Sample Size vs. Execution Time.

The distribution indicates polynomial time, worst case complexity for sequential message passing.

Chapter 9 - CICO Park Simulations

Elevation data from CICO park, a public park located in the city of Manhattan, Kansas, USA, was used as a platform for the proposed approach. The data was in the form of a 170×110 lattice $\mathcal{L}$ of regularly placed points at intervals of 15 feet along the latitude as well as the longitude, whereupon the lattice granularity $d_0 = 4.572$ m was used everywhere in all results reported in this section. The elevations of the $|\mathcal{L}| = 18700$ points were also available in the geographical data. The area of the rectangular region covered the lattice was 777.24×502.92 m^2.

This elevation data was initially employed to estimate the minimum work employing the methodology outlined in Chapter 6 - resulting in a sparse matrix of dimensions 18700×18700. Subsequently, a set $\mathcal{C}$ was found with a total number of $|\mathcal{C}| = 147$ waypoints with the method proposed in Chapter 4 - for waypoint selection. A smaller matrix of size 147×147, whose elements contained information of the minimum work, was introduced to the Multi-Robot cyclic tour optimization method proposed in this research.

Table II

No. Robots	Max. Iterations	Damp. factor	Bal. factor	Del. factor
$\|\mathcal{G}\|$	-	λ	γ	σ
1	4000	0.8	0.25	0.5
2	2800	0.8	0.5	0.5
3	3200	0.75	0.4	0.25
4	4000	0.8	0.5	0.08
5	4000	0.8	0.4	0.1

To evaluate the proposed method's efficacy, five distinct experiments were conducted, varying the number of robots $|\mathcal{G}|$. Table II presents the parameters deemed appropriate for each test. As the proposed method sends messages iteratively through the factor graph model, a maximum number of iterations needed to be defined by the user. It is essential to emphasize that the algorithm preserves its deterministic nature, as no random components are introduced at any stage. The damping factor λ , is commonly used to prevent oscillations and was set close to the unity. The balance factor γ was set around 0.5. Moreover, the delete factor σ was introduced. This parameter represented the percentage of subtour constraints that could be safely deleted once they were fulfilled during runtime. The rationale behind this parameter was grounded in the understanding

that once a subtour was successfully removed, the corresponding function node's messages became zero. Since the likelihood of the same subtour reappearing was low, deleting these functions enhanced the algorithm's performance without compromising the process.

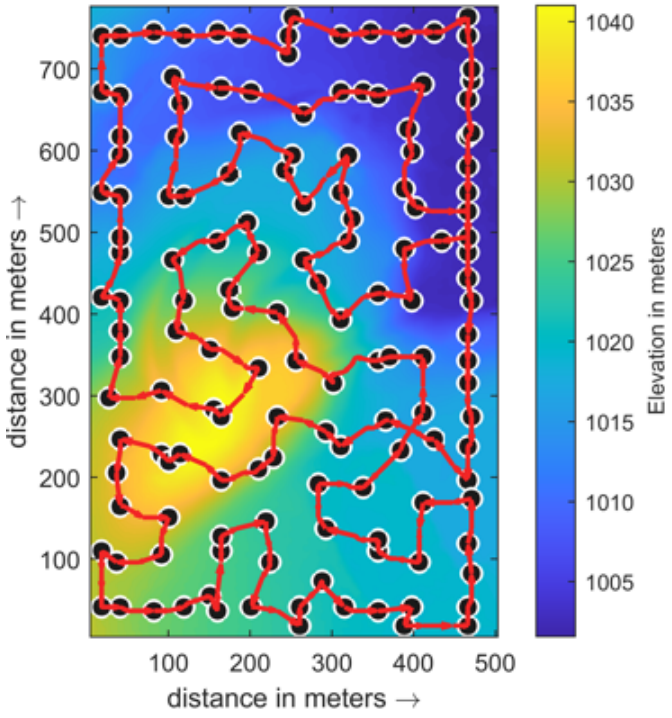
After defining these parameters, the algorithm was implemented on the CICO park dataset. The contour plots depicting the region, for five distinct outcomes corresponding to different values for $|\mathcal{G}|$, are illustrated in Figure 8. It is important to notice that the connection between each pair of waypoints is depicted with a set of small arrows, which represent the Dijkstra's path. Following the reasoning proposed in Chapter 2 - , the trajectory is traced with the help of the shortest path algorithm. The initial location is depicted with a blue cross and was defined to be closed to the center of the region for all the cases. Also, due to the asymmetric character of the problem, arrows were utilized to indicate the specific direction each robot should follow when seeding the field.

Remarkably, the results in Figure 8 showcased well-balanced paths, demonstrating the algorithm's efficiency in optimizing trajectories. Furthermore, waypoints selected through affinity propagation clustering, exhibited an even distribution throughout the region including the elevated and challenging zones.

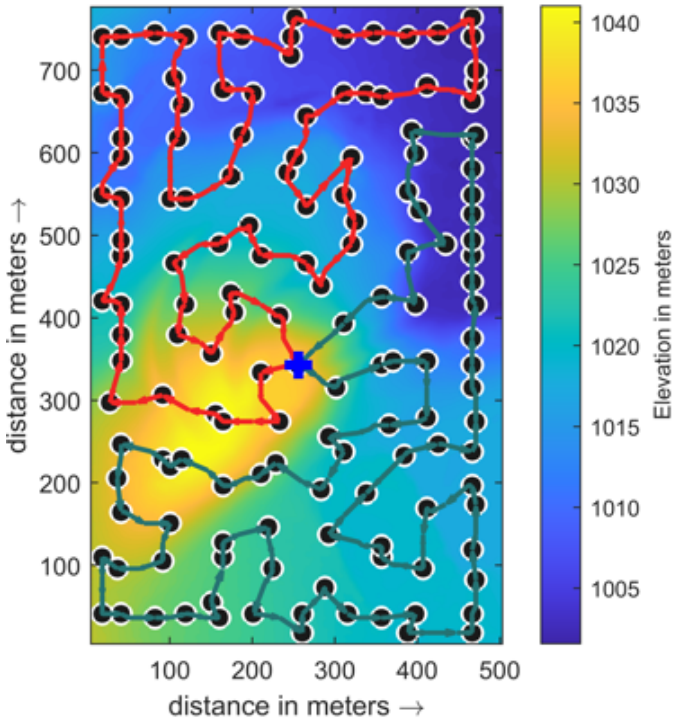
Table III shows the calculated total energy $\Omega_{\mathcal{R}}$ for each experiment, representing the sum of work done by the robots, as discussed in Chapter 2 - . Additionally, the table provides the execution time required by the Multi-robot cyclic tour optimization algorithm, run in Matlab with a Intel® Core i7 processor.

There is an striking finding in the table: the most optimal outcome is achieved with two robots. One possible reason for this intriguing result could be explained by analyzing the area surrounding the initial waypoint in all the experiments depicted in Figure 8. It becomes evident that with an increase in the number of robots, additional edges are created around this specific zone. Consequently, the work executed by one robot becomes redundant due to overlap with the efforts of other robots, leading to an unnecessary consumption of energy. Determining the appropriate number of robots for a given problem is a task out of the scope of this study.

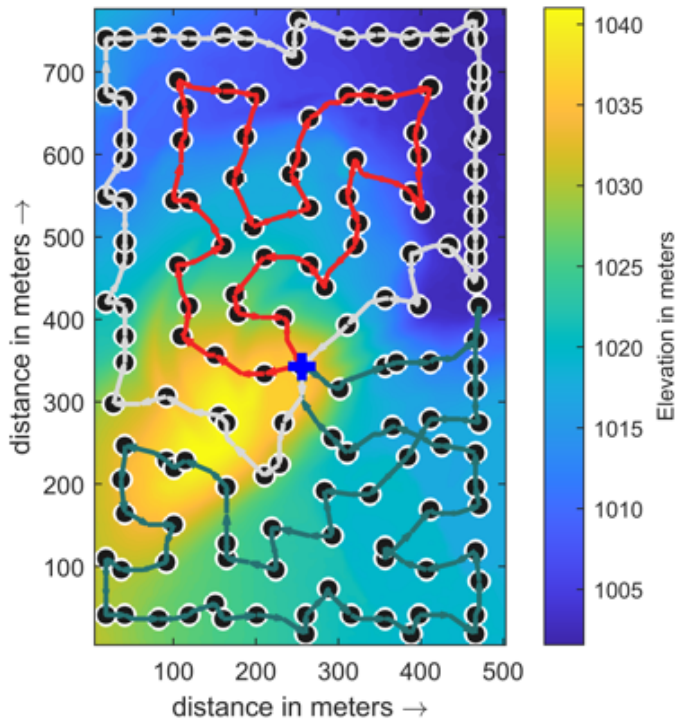
One robot



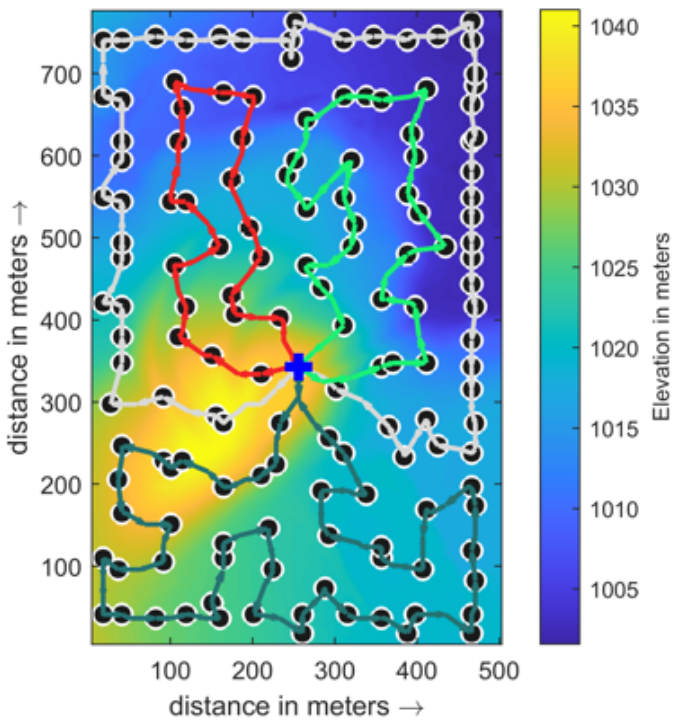
Two robots



Three robots



Four robots



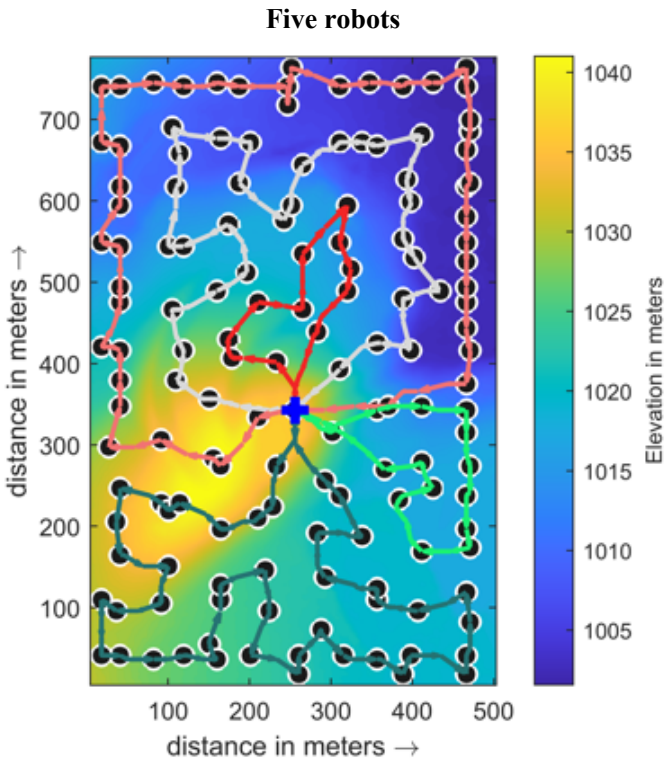


Figure 8. Results with CICO park data.

Contour plots are depicted along with the results provided by the algorithm. The paths for multiple robots are drawn with different color arrows. The initial point is depicted with a blue cross.

Table III

No. Robots $\lvert\mathcal{G}\rvert$	Total Energy $\times 10^7$N	Execution time (s)
1	1.5491	275.13
2	**1.5462**	331.23
3	1.5981	767.71
4	1.6002	3328.47
5	1.6622	14783.40

Chapter 10 - Comparison with Related Works

Comparison with a Genetic optimization algorithm

In Chapter 1 - , a widely recognized approach, Genetic Algorithms, was briefly mentioned. GA is a method commonly found in the literature. Of particular significance to this research is the work by (Király & Abonyi, 2015), which introduced a technique for determining a route path with minimal cost in the context of a supply system. Their approach involved a multi-chromosome technique designed to solve the multiple Traveling Salesman Problem (mTSP). The objectives outlined in their study closely parallel the goals of this research. Consequently, this paper holds relevance, serving as a valuable reference point. The insights gleaned from (Király & Abonyi, 2015) provided a solid foundation for evaluating the performance of the multi-robot cycle path planning algorithm (hereafter referred as proposed *method*) through comparative experiments, especially with CICO park data. By drawing upon their methodology and results, this research gained essential benchmarks for assessing the effectiveness of this research

The formulation of (Király & Abonyi, 2015) work, categorized as an Asymmetric Multiple Traveling Salesman Problem with Time Windows (mTSP), was directly applied to an industrial context. In that framework, where the central depot served as the starting point, various trucks were tasked with supplying essential materials to city nodes within the vicinity. Each salesman's route began and ended at the central depot. The optimization problem statement incorporated constraints such as maximum tour length per salesperson and truck capacity. However, one of their constraints, the time window at each location, was disregarded in this experiment as it didn't align with the requirements for coverage path planning in CICO park.

For sake of completeness, a brief explanation of the genetic algorithm utilized by (Király & Abonyi, 2015) is next presented. In their formulation, each chromosome represented an individual salesman's tour. The gene sequence within the chromosome delineated the trajectory to be traversed by each truck. The algorithm started with an initial population of randomly generated individuals, and the fitness function computed the total route length. Individuals with the smallest fitness values were selected for generating new individuals, employing complex operators to sustain the evolutionary process. Finally, once the algorithm converged, the final tour was provided by the survival individuals.

The objective of the proposed experiment was to find optimal paths for CICO park with the work of (Király & Abonyi, 2015) to get a benchmark for a fair comparison with the proposed method. As the message passing algorithm for multi-cycle tour was already tested in CICO park, in Chapter 9 - , the rest of this subsection is dedicated to implement (Király & Abonyi, 2015) method with CICO park data and then compare the results. To facilitate this task, their code was configured with appropriate parameters like population size and number of iterations, obtained from their webpage and implemented in MATLAB® programming language. One critical modification was necessary: ensuring the number of salesmen remained fixed, preventing the algorithm from varying this variable during compilation. Consequently, two versions of the code (V1, V2) were created, both utilized in the experiment.

After defining parameters and constraints for a fair comparison, the mTSP algorithm underwent 100 trials. The resulting total energy values from each method were tabulated in Table IV (the results for the proposed method are exactly the same as in Table III (they were retyped in this table just for easy comparison). Notably, the proposed method consistently demonstrated lower energy consumption across all scenarios. These findings underscore the superior performance and efficiency of our proposed approach in addressing the specific challenges presented by the CICO park coverage path planning problem.

Curiously, the mTSP Genetic Algorithm performed optimally with V2 for one robot, while our proposed method indicated the most efficient configuration involved two robots. Spatial plots illustrating the routes for one and two robots for both methods are presented in Figure 9, with additional plots available but not included here for brevity.

Table IV

Total Energy $\times 10^7$N			
No. Robots $\lvert\mathcal{G}\rvert$	mTSP GA V1 best out of 100 trials	mTSP GA V2 best out of 100 trials	Proposed
1	1.6876	1.5864	1.5491
2	1.5948	1.6139	**1.5462**
3	1.6088	1.6979	1.5981
4	1.6605	1.7608	1.6002
5	1.7261	1.8930	1.6622

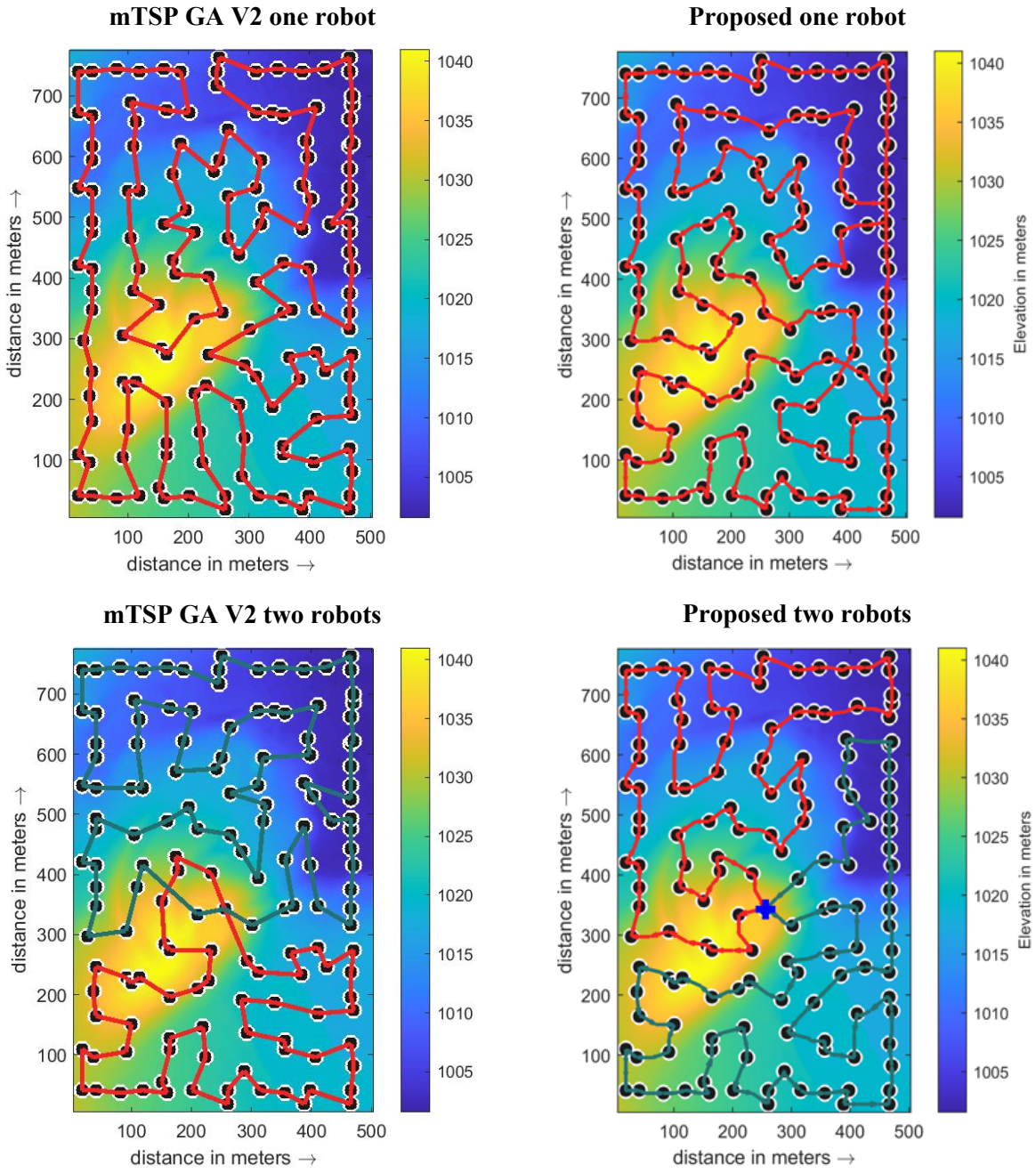


Figure 9. Experiments with mTSP GA V2, and CICO park data.

Contour plots for both methods are presented for comparison purposes. The mTSP GA V2 (left) and proposed method (right).

Comparison with a Cooperative path planning strategy for multi-robots

In this experiment, the proposed approach is compared to the work presented in (Tang et al., 2020),which addresses a similar problem but with a slightly different focus. In their research, a team of robots operating within a limited sensing radius covers an irregular area while avoiding obstacles, aiming to minimize the total cost of full coverage.

Their strategy comprises three main stages: sensor deployment, akin to the waypoint selection proposed in this research; area partition, achieved by clustering sensor points into a predetermined number of sub-regions; and computing optimal paths. The first stage employs the particle swarm optimization algorithm to determine deployed sensors or waypoints. The subsequent stages involve dividing waypoints among the appropriate number of robots and computing optimal paths simultaneously. Area partition utilizes a weighted version of the k-means clustering algorithm, while Genetic Algorithms and the A-star algorithm are employed for finding optimal paths.

Comparing the proposed method with (Tang et al., 2020) is significant for this research as it introduces the challenging aspect of identifying obstacles within the region. While the proposed method had been tested with convex regions thus far, this experiment explores its capability to navigate around obstacles.

The experiment started within a simulated environment created by the authors, featuring three differently shaped prohibited areas that functioned as obstacles. No robot was allowed to traverse these regions. The authors provided their results, which served as a benchmark for our comparison. The simulated area was replicated in MATLAB, and the proposed method was adapted accordingly. The lattice of the region was constructed, ensuring points within prohibited areas were disconnected from other points (in the graph model) as well as among themselves. Subsequently, the waypoint selection and affinity propagation clustering were applied to find the waypoints in the region and divide them into the number of robots. Finally, the proposed method was employed to find optimal paths.

It is crucial to note that in (Tang et al., 2020)'s problem statement, robots started from different locations, and there was no requirement for them to return to their initial points. To ensure a fair comparison, our method was adapted to the single robot version and executed as many times as the number of deployed robots. The method was rigorously tested with 3, 4, and 5 robots, maintaining a sensing radius of $\rho = 10$ m and deploying a total of 85 sensors. This comprehensive

comparison allowed to assess the adaptability and efficacy of our proposed approach in complex environments with obstacles.

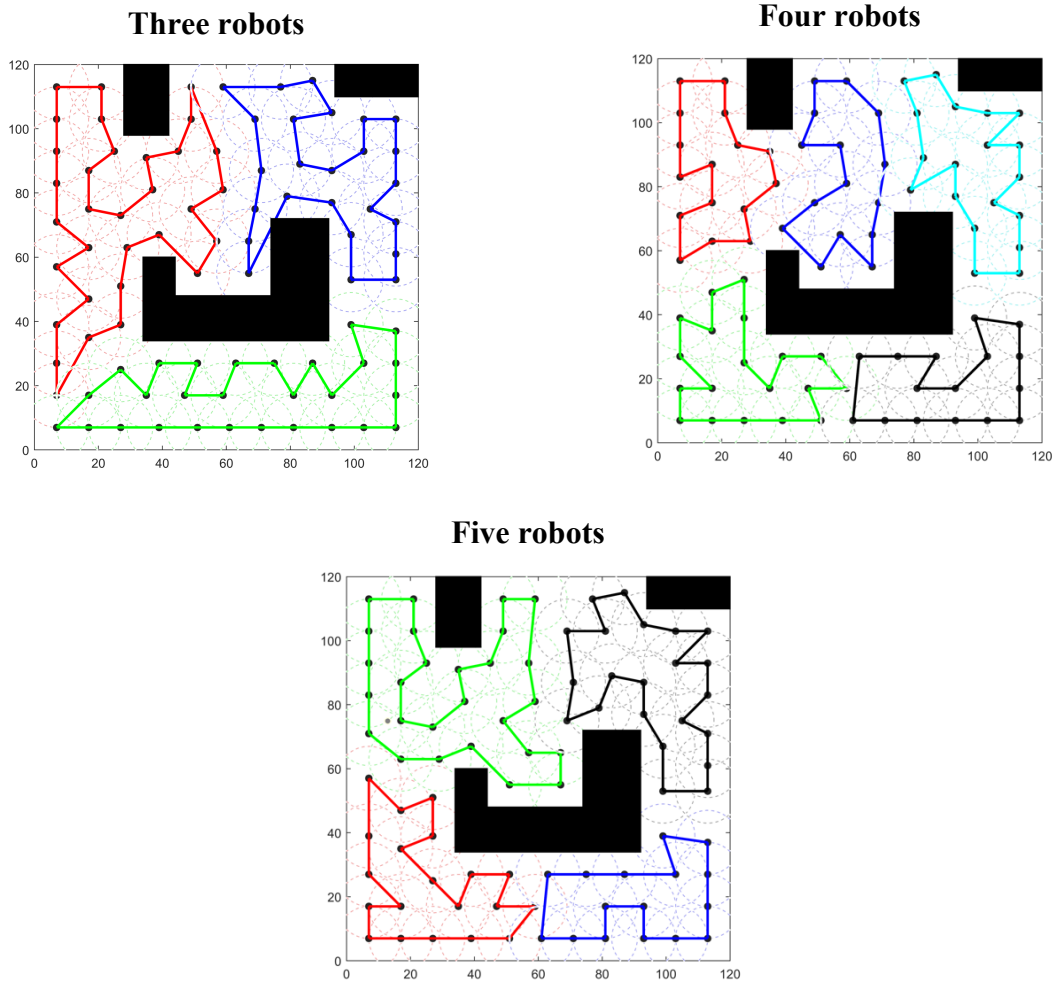


Figure 10. Experiments with the proposed method in the simulated environment proposed by the reference paper.

The squared area of 120x120m contains three prohibited regions or obstacles depicted with black polygons. It was used to test the proposed method for three, four and five robots.

Results shown in Figure 10 provide a visualization of how the simulated environment looks and the solution provided by the proposed method. Certainly, it avoided the prohibited areas and provided reasonably good results. Table V provides quantitative values that helps the comparison with the reference paper (results for (Tang et al., 2020)are the ones reported in their paper). The values for average length path for the proposed method were less than the reference paper, assuring fully coverage even with less total number of sensors.

Table V

No. of Robots	Tang, Zhou, Sun, Di, Xiong 93 sensors [Ref]	Max-Sum Message Passing 85 sensors (proposed)
$\|\mathcal{G}\|$	Avg. Length (m)	Avg. Length (m)
3	362.10	**339.25**
4	266.77	**248.41**
5	207.26	**198.03**

Comparison with Voronoi-based path generation algorithm

For the next experiment, the study under consideration is (Jensen-Nau et al., 2021), presenting a Voronoi-based path generation (VPG) algorithm tailored for an energy-constrained mobile robot. The primary objective outlined in this paper was to strategically distribute waypoints across a designated area to maximize the covered region, all within the confines of limited path-length imposed by energy constraints on the robot.

The algorithm operates by conceptualizing the path as a connected network resembling a mass-spring-damper system. Utilizing Voronoi diagrams, the algorithm simplifies the system dynamics and diminishes the model's complexity. Here's how it functions: a predetermined number of waypoints, computed based on energy constraints, are initially distributed randomly across the area with specified spacing. Thereafter, a Voronoi diagram is constructed considering these waypoints' positions, and centroids of each Voronoi polygon are calculated. Springs connect these centroids to their corresponding waypoints, and each waypoint is linked to its adjacent counterparts. The forces exerted by the centroids and adjacent waypoints through the springs dictate the waypoints' new positions, and the algorithm converges when no further movement is detected.

In contrast to the proposed method, this work is not designed for multi-robot use. Nevertheless, it is of particular interest to this research due to its coverage and path planning integration. Notably, it abstains from area decomposition into grids and avoids direct optimization techniques. Thus, it offers a unique opportunity to assess and compare two methods with entirely different approaches side by side.

For this experiment, both methods were applied to an area resembling the geometry of CICO park. It is vital to acknowledge that the VPG algorithm is not suited for asymmetric problems.

Consequently, elevation data was omitted, and our proposed method was deployed in its symmetric version. The objective was to cover the entire area of CICO park, with a maximum path length of 11,500m and a coverage radius of 60m. Preserving a fixed separation distance of 38m, both methods were tasked with utilizing 300 waypoints to accomplish the coverage task

Table VI

	Proposed (symmetric)	VPG (Best out of 50 trials)
Number of waypoints	300	300
Covered Area (m^2)	**390889.54**	390459.67
Coverage percentage	100%	99.88%
Total Time (min)	1171.2	4166.97

In Table VI, the results of the experiment are presented. Under identical conditions, slightly more coverage of the area was achieved by the proposed method. Figure 11 displays plots of the CICO park area, showcasing the distribution and paths generated by both methods. The covered area is depicted in green, highlighting the proposed method's ability to cover the entire region, whereas the VPG method leaves small areas in white (left border, almost imperceptible). It's noteworthy that the VPG algorithm overlooks the energy expenditure associated with the robot's return, hence the start and end locations are indicated in yellow.

Additionally, the proposed method demonstrates a remarkably uniform distribution, whereas the VPG solution appears intricate and convoluted. This intricacy could pose challenges for real-world applications, especially when programming a ground robot, indicating a potential drawback in the VPG approach.

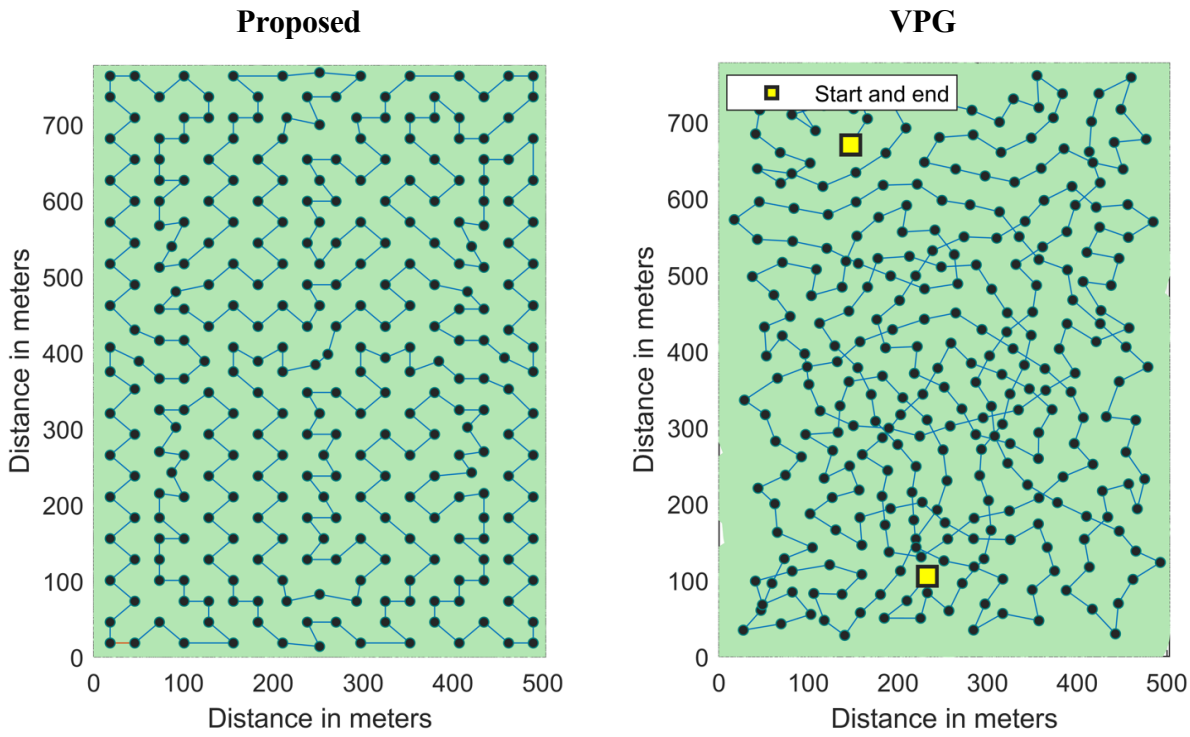


Figure 11. Experiments with VPG and propose method in CICO park area.

These plots show the results for both algorithms with the CICO park area. Proposed shows a quite uniform distribution of the waypoints while VPG's path is intricated and lefts small areas in blank.

Chapter 11 - Results with Reinforcement Learning

AS the CICO park data did not contain a large enough unevenness for fine-tuning to be effective, the performance of the reinforcement learning algorithm discussed in Chapter 7 - were evaluated using two artificially generated landscapes, A and B. The lattices had the same granularity d_0 the CICO park data, and the elevations were scaled so that the slopes between any pair of points within the lattice neighborhoods were within the range given in Table I. The learning algorithm was applied only to single robot routes. The preferences were initialized to $p_0 = -3500|\mathcal{L}|$. There were $S = 10$ samples per epoch and the number of epochs were $N_{\max} = 40$ (landscape A) and $N_{\max} = 50$ (landscape B). The learning rate was initialized to $\eta = 0.05$ and lowered linearly so that at the end of $N_{\max}$ epochs, $\eta = 0$.

The first version of the learning algorithm (RL-1) was implemented without any neighborhood increments. In the second version (RL-2), neighboring points were updated as shown in Eqn. (49). In order to keep the number of independent learning parameters at the minimum possible, the parameters z_0 and σ were set to $z_0 = d_0$ and $\sigma = d_0$.

Figure 12 and 11 illustrate how the two versions of the learning algorithm were able to lower the total work Ω required by the cyclic tour. The plots show the sample median and minimum values ($\bar{\Omega}^{(n)}$ and $\min_s \Omega_s^{(n)}$) in each epoch n ($n = 1,2, \ldots N_{\max}$). Figure 12 shows results that were obtained for landscape A, while Figure 13, for landscape B. The plots show the averaged values of 25 independent trials. It can be observed that the second version of reinforcement learning converged faster than the original.

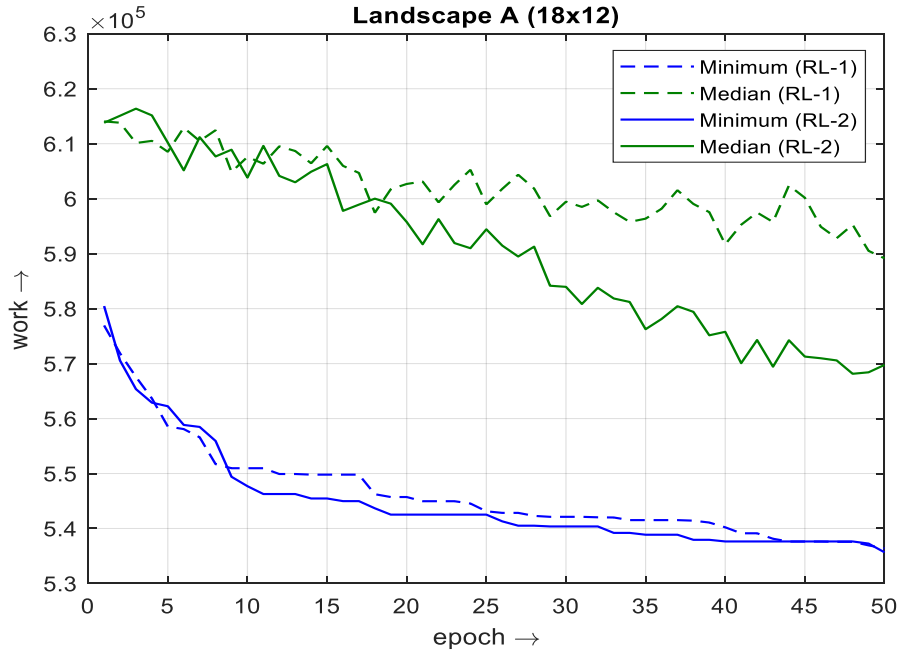


Figure 12. Convergence Plots (Landscape A).

The plots show the performances of RL-1 (dashed lines) and RL-2 (solid lines) for landscape A. Episodes (X-axis) and the corresponding values of the sample median (green) and minimum (blue) (Y-axis) are shown.

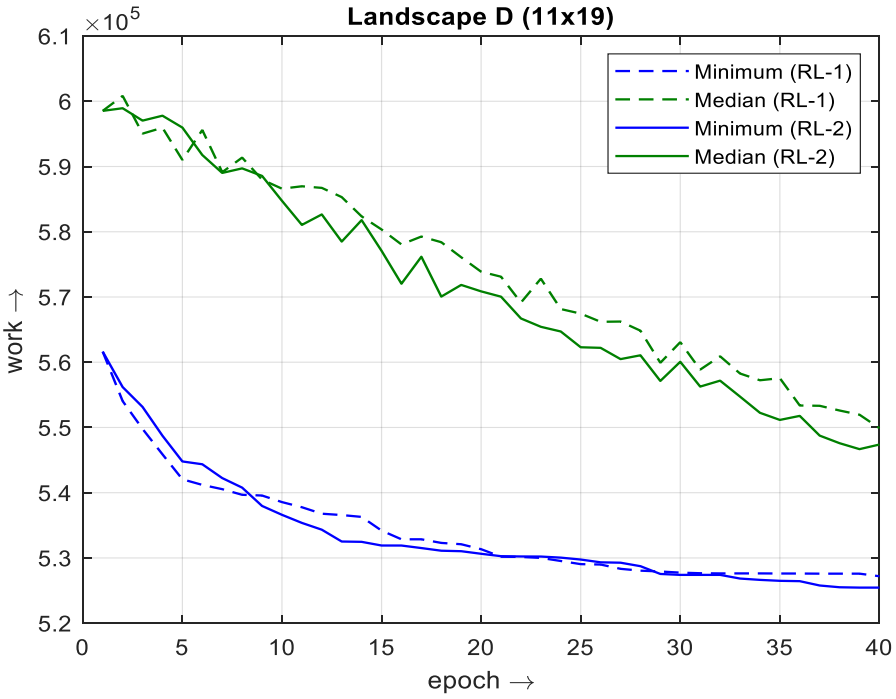


Figure 13. Convergence Plots (Landscape B).

The plots show the performances of RL-1 (dashed lines) and RL-2 (solid lines) for landscape B. Episodes (X-axis) and the corresponding values of the sample median (green) and minimum (blue) (Y-axis) are shown.

Figure 14 shows the sample median value of the number of waypoints ($|\mathcal{C}_s^{(n)}|$) as learning progressed in RL-2. In spite of the updates applied to the preferences at the end of each epoch, there was no noticeable difference in the number of waypoints.

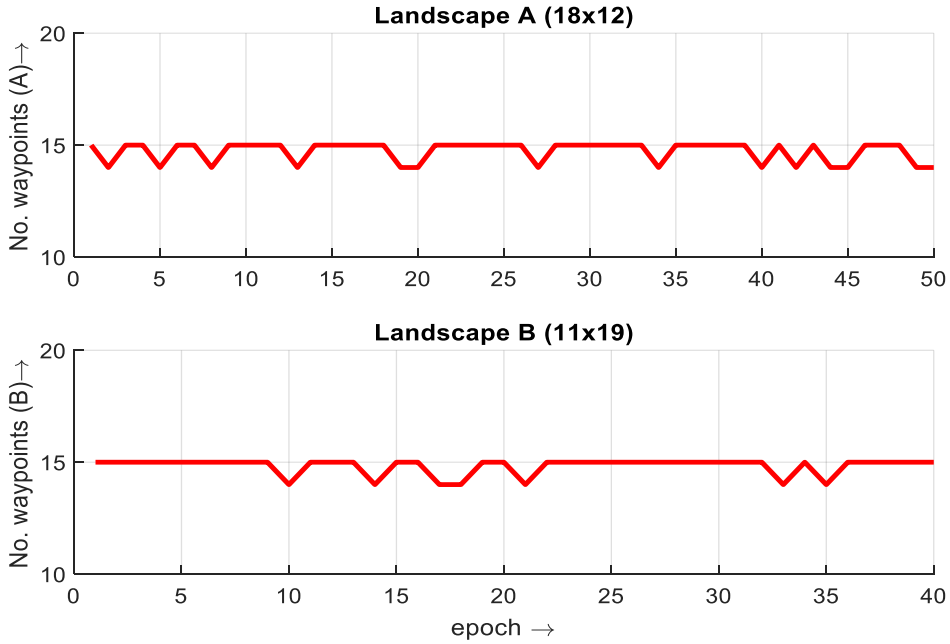


Figure 14. Number of Waypoints.

The plots show the number of waypoints in each training episode. Episodes (X-axis) and median number of waypoints are shown when RL-2 is applied to landscape A (top) and landscape B (bottom).

Figure 15 shows the cyclic tour in landscape A with the least amount of work from all 25 trials. The total work for this tour was $\Omega \approx 52.16 \times 10^4$ Figure 16 shows such a tour with landscape B, which required a total work of $\Omega \approx 52.16 \times 10^4$.

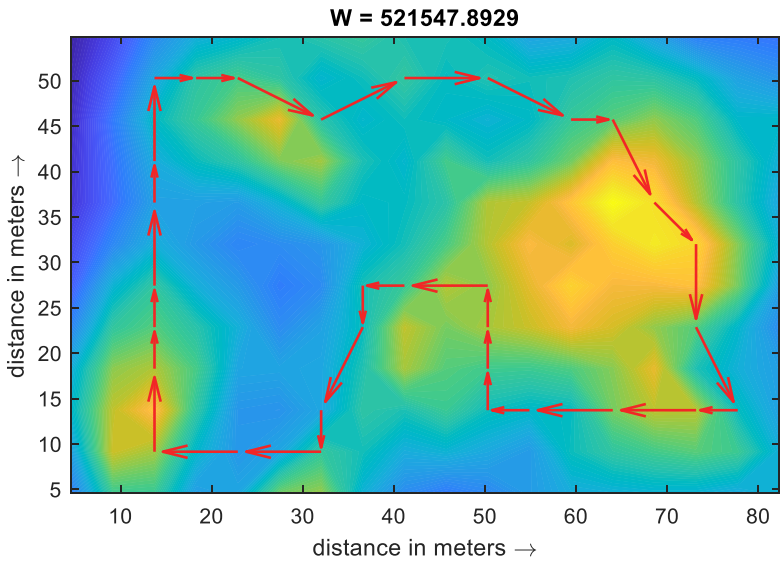


Figure 15. Optimal Tour (Landscape A).

The figure shows the optimal cyclic tour obtained by learning algorithm RL-2 for a single robot.

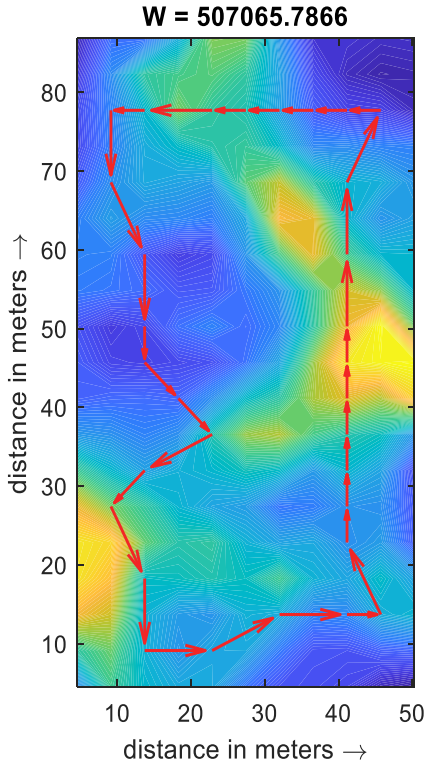


Figure 16. Optimal Tour (Landscape B).

The figure shows the optimal cyclic tour obtained by learning algorithm RL-2 for a single robot.

Chapter 12 - Conclusions and Future Research

In this study, a successful formulation for Coverage path planning applications was introduced. The combination of exemplar clustering for waypoint selection and a message passing algorithm for multi-robot cyclic tour optimization yielded significant success. This novel deterministic approach demonstrated remarkable efficacy, handling up to 200 waypoints seamlessly in an irregular terrain. Its results as compared to genetic algorithms (one of the most widely used methods for multi-robot trajectories optimization) showed its power and potential. Besides its remarkable capacity to minimize energy, its implications extend beyond coverage path planning, addressing a challenging binary optimization problem.

The formalism devised for discretizing regions using graph models exhibited exceptional resilience in the presence of obstacles and prohibited areas, as explained in Comparison with Voronoi-based path generation algorithm (Chapter 10 -). Avoiding these areas posed minimal challenges, with minor modifications to the graph model sufficing to isolate points within these zones. This adaptability hints at the method's potential to tackle non-convex areas, broadening its applicability to diverse real-world problems.

Furthermore, our study shed light on the integration of machine learning methods in practical problem-solving. Through adept algorithms and data-driven modeling, terrain representation becomes a reality, opening avenues for innovative applications.

Future Research

In the pursuit of enhancing execution efficiency, particularly when dealing with over 200 waypoints, two additional constraints for the FGM were explored. One of them could exploit the fact that the total number of edges around the initial locations is always fixed. The other could repose the subtour constraint to send message also withing the subtour edges. While not detailed in this document due to short time, these new constraints promise to optimize the multi-robot cyclic tour algorithm further. Preliminary analyses showcased significant improvements in execution time, particularly when the initial location is near the area's corners. These constraints are slated for inclusion in an upcoming journal paper.

Additionally, our discretization method proved highly effective when presence of non-convex areas, even though some experiments were omitted from this study due to time limitations. These omitted experiments revealed the simplicity of the discretization process. While the method

exhibited remarkable flexibility, further studies are essential to substantiate this claim, paving the way for broader applications in intricate real-world scenarios.

References

Almadhoun, R., Taha, T., Seneviratne, L., & Zweiri, Y. (2022). Multi-Robot Hybrid Coverage Path Planning for 3D Reconstruction of Large Structures. *IEEE Access*, *10*, 2037–2050. https://doi.org/10.1109/ACCESS.2021.3139080

Badgujar, C., Das, S., Figueroa, D. M., & Flippo, D. (2023). Application of Computational Intelligence Methods in Agricultural Soil–Machine Interaction: A Review. *Agriculture*, *13*(2), 357. https://doi.org/10.3390/agriculture13020357

Badgujar, C., Flippo, D., & Welch, S. (2022). Artificial neural network to predict traction performance of autonomous ground vehicle on a sloped soil bin and uncertainty analysis. *Computers and Electronics in Agriculture*, *196*, 106867. https://doi.org/10.1016/j.compag.2022.106867

Badgujar, C. M., Flippo, D., Brokesh, E., & Welch, S. (2022). Experimental Investigation on Traction, Mobility, and Energy Usage of a Tracked Autonomous Ground Vehicle on a Sloped Soil Bin. *Journal of the ASABE*, *65*(4), 835–847. https://doi.org/10.13031/ja.14860

Bose, P., Maheshwari, A., Shu, C., & Wuhrer, S. (2011). A survey of geodesic paths on 3D surfaces. *Computational Geometry*, *44*(9), 486–498. https://doi.org/10.1016/j.comgeo.2011.05.006

Chakraborty, S., Elangovan, D., Govindarajan, P. L., ELnaggar, M. F., Alrashed, M. M., & Kamel, S. (2022). A Comprehensive Review of Path Planning for Agricultural Ground Robots. *Sustainability*, *14*(15), 9156. https://doi.org/10.3390/su14159156

Chi, W., Wang, J., Ding, Z., Chen, G., & Sun, L. (2022). A Reusable Generalized Voronoi Diagram-Based Feature Tree for Fast Robot Motion Planning in Trapped Environments. *IEEE Sensors Journal*, *22*(18), 17615–17624. https://doi.org/10.1109/JSEN.2021.3054888

Dai, G., Guo, L., Gutin, G., Zhang, X., & Zhang, Z.-B. (2022). Iterative Message Passing Algorithm for Vertex-Disjoint Shortest Paths. *IEEE Transactions on Information Theory*, *68*(6), 3870–3878. https://doi.org/10.1109/TIT.2022.3145232

Delmerico, J., Mueggler, E., Nitsch, J., & Scaramuzza, D. (2017). Active autonomous aerial exploration for ground robot path planning. *IEEE Robotics and Automation Letters*, *2*(2), 664–671. https://doi.org/10.1109/LRA.2017.2651163

Farinelli, A., Rogers, A., & Jennings, N. R. (2014). Agent-based decentralised coordination for sensor networks using the max-sum algorithm. *Autonomous Agents and Multi-Agent Systems*, *28*(3), 337–380. https://doi.org/10.1007/s10458-013-9225-1

Fioretto, F., Pontelli, E., Yeoh, W., & Dechter, R. (2018). Accelerating exact and approximate inference for (distributed) discrete optimization with GPUs. *Constraints*, *23*(1), 1–43. https://doi.org/10.1007/s10601-017-9274-1

Frey, B. J., & Dueck, D. (2007). Clustering by Passing Messages Between Data Points. *Science*, *315*(5814), 972–976. https://doi.org/10.1126/science.1136800

Galceran, E., & Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, *61*(12), 1258–1276. https://doi.org/10.1016/j.robot.2013.09.004

Givoni, I. E., & Frey, B. J. (2009). A Binary Variable Model for Affinity Propagation. *Neural Computation*, *21*(6), 1589–1600. https://doi.org/10.1162/neco.2009.05-08-785

Held, M., & Karp, R. M. (1970). The Traveling-Salesman Problem and Minimum Spanning Trees. *Operations Research*, *18*(6), 1138–1162. https://doi.org/10.1287/opre.18.6.1138

Huang, C., Zhao, Y., Zhang, M., & Yang, H. (2023). APSO: An A*-PSO Hybrid Algorithm for Mobile Robot Path Planning. *IEEE Access*, *11*, 43238–43256. https://doi.org/10.1109/ACCESS.2023.3272223

Huang, X., Sun, M., Zhou, H., & Liu, S. (2020). A multi-robot coverage path planning algorithm for the environment with multiple land cover types. *IEEE Access*, *8*, 198101–198117. https://doi.org/10.1109/ACCESS.2020.3027422

Ishat-E-Rabban, M., & Tokekar, P. (2021). Failure-resilient coverage maximization with multiple robots. *IEEE Robotics and Automation Letters*, *6*(2), 3894–3901. https://doi.org/10.1109/LRA.2021.3067275

Jensen-Nau, K. R., Hermans, T., & Leang, K. K. (2021). Near-Optimal Area-Coverage Path Planning of Energy-Constrained Aerial Robots with Application in Autonomous Environmental Monitoring. *IEEE Transactions on Automation Science and Engineering*, *18*(3), 1453–1468. https://doi.org/10.1109/TASE.2020.3016276

Ju, C., Kim, J., Seol, J., & Son, H. Il. (2022). A review on multirobot systems in agriculture. *Computers and Electronics in Agriculture*, *202*, 107336. https://doi.org/10.1016/j.compag.2022.107336

Király, A., & Abonyi, J. (2015). Redesign of the supply of mobile mechanics based on a novel genetic optimization algorithm using Google Maps API. *Engineering Applications of Artificial Intelligence*, *38*, 122–130. https://doi.org/10.1016/j.engappai.2014.10.015

Kschischang, F. R., Frey, B. J., & Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, *47*(2), 498–519. https://doi.org/10.1109/18.910572

Kumar, S. N., & Panneerselvam, R. (2012). A Survey on the Vehicle Routing Problem and Its Variants. *Intelligent Information Management*, *04*(03), 66–74. https://doi.org/10.4236/iim.2012.43010

Le, A. V., Nhan, N. H. K., & Mohan, R. E. (2020). Evolutionary algorithm-based complete coverage path planning for tetriamond tiling robots. *Sensors (Switzerland)*, *20*(2). https://doi.org/10.3390/s20020445

Le, A. V., Prabakaran, V., Sivanantham, V., & Mohan, R. E. (2018). Modified a-star algorithm for efficient coverage path planning in tetris inspired self-reconfigurable robot with integrated laser sensor. *Sensors (Switzerland)*, *18*(8). https://doi.org/10.3390/s18082585

Li, J., Zhou, M., Sun, Q., Dai, X., & Yu, X. (2015). Colored Traveling Salesman Problem. *IEEE Transactions on Cybernetics*, *45*(11), 2390–2401. https://doi.org/10.1109/TCYB.2014.2371918

Liu, L., Wang, X., Yang, X., Liu, H., Li, J., & Wang, P. (2023). Path planning techniques for mobile robots: Review and prospect. In *Expert Systems with Applications* (Vol. 227). Elsevier Ltd. https://doi.org/10.1016/j.eswa.2023.120254

Maini, P., Gonultas, B. M., & Isler, V. (2022). Online Coverage Planning for an Autonomous Weed Mowing Robot with Curvature Constraints. *IEEE Robotics and Automation Letters*, *7*(2), 5445–5452. https://doi.org/10.1109/LRA.2022.3154006

Martinez-Figueora, D., Das, S., Badgujar, C., Flippo, D., & Welch, S. J. (2022). A Distributed Approach for Robotic Coverage Path Planning Under Steep Slope Terrain Conditions. *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, 970–977. https://doi.org/10.1109/SSCI51031.2022.10022252

Miao, X., Lee, J., & Kang, B. Y. (2018). Scalable coverage path planning for cleaning robots using rectangular map decomposition on large environments. *IEEE Access*, *6*, 38200–38215. https://doi.org/10.1109/ACCESS.2018.2853146

Middendorf, G., Becerra, T. A., & Cline, D. (2009). Transition and Resilience in the Kansas Flint Hills. *Online Journal of Rural Research & Policy*, *4*(3). https://doi.org/10.4148/ojrrp.v4i3.109

Moysiadis, V., Tsolakis, N., Katikaridis, D., Sørensen, C. G., Pearson, S., & Bochtis, D. (2020). Mobile robotics in agricultural operations: A narrative review on planning aspects. *Applied Sciences (Switzerland)*, *10*(10). https://doi.org/10.3390/app10103453

Mukhamediev, R. I., Yakunin, K., Aubakirov, M., Assanov, I., Kuchin, Y., Symagulov, A., Levashenko, V., Zaitseva, E., Sokolov, D., & Amirgaliyev, Y. (2023). Coverage Path Planning Optimization of Heterogeneous UAVs Group for Precision Agriculture. *IEEE Access*, *11*, 5789–5803. https://doi.org/10.1109/ACCESS.2023.3235207

Murphy, K. P., Weiss, Y., & Jordan, M. I. (n.d.). *Loopy Belief Propagation for Approximate Inference: An Empirical Study*.

Petersen, P. (2016). *Riemannian Geometry* (Vol. 171). Springer International Publishing. https://doi.org/10.1007/978-3-319-26654-1

Qi, J., Yang, H., & Sun, H. (2021). MOD-RRT*: A Sampling-Based Algorithm for Robot Path Planning in Dynamic Environment. *IEEE Transactions on Industrial Electronics*, *68*(8), 7244–7251. https://doi.org/10.1109/TIE.2020.2998740

Ravanbakhsh, S., Rabbany, R., & Greiner, R. (2014). *Augmentative Message Passing for Traveling Salesman Problem and Graph Partitioning*. http://arxiv.org/abs/1406.0941

Santos, L. C., Santos, F. N., Solteiro Pires, E. J., Valente, A., Costa, P., & Magalhaes, S. (2020). Path Planning for ground robots in agriculture: a short review. *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 61–66. https://doi.org/10.1109/ICARSC49921.2020.9096177

Smith D. (2012). *Physical Geography of Kansas*.

Tan, C. S., Mohd-Mokhtar, R., & Arshad, M. R. (2021). A Comprehensive Review of Coverage Path Planning in Robotics Using Classical and Heuristic Algorithms. *IEEE Access*, *9*, 119310–119342. https://doi.org/10.1109/ACCESS.2021.3108177

Tang, Y., Zhou, R., Sun, G., Di, B., & Xiong, R. (2020). A Novel Cooperative Path Planning for Multirobot Persistent Coverage in Complex Environments. *IEEE Sensors Journal*, *20*(8), 4485–4495. https://doi.org/10.1109/JSEN.2019.2963697

Tao, J., Zhang, J., Deng, B., Fang, Z., Peng, Y., & He, Y. (2021). Parallel and Scalable Heat Methods for Geodesic Distance Computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(2), 579–594. https://doi.org/10.1109/TPAMI.2019.2933209

Tenenbaum, J. B., Silva, V. de, & Langford, J. C. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, *290*(5500), 2319–2323. https://doi.org/10.1126/science.290.5500.2319

Van Pham, H., Moore, P., & Truong, D. X. (2019). Proposed smooth-STC algorithm for enhanced coverage path planning performance in mobile robot applications. *Robotics*, *8*(2). https://doi.org/10.3390/ROBOTICS8020044

Wu, C., Dai, C., Gong, X., Liu, Y. J., Wang, J., Gu, X. D., & Wang, C. C. L. (2019). Energy-efficient coverage path planning for general terrain surfaces. *IEEE Robotics and Automation Letters*, *4*(3), 2584–2591. https://doi.org/10.1109/LRA.2019.2899920

Wu, Y., Li, M., Li, G., & Savaria, Y. (2022). Persistence Region Monitor with a Pheromone-Inspired Robot Swarm Sensor Network. *IEEE Internet of Things Journal*, *9*(14), 12093–12110. https://doi.org/10.1109/JIOT.2021.3133501

Yazici, A., Kirlik, G., Parlaktuna, O., & Sipahioglu, A. (2014). A dynamic path planning approach for multirobot sensor-based coverage considering energy constraints. *IEEE Transactions on Cybernetics*, *44*(3), 305–314. https://doi.org/10.1109/TCYB.2013.2253605

Zhang, F., & O'Donnell, L. J. (2020). Support vector regression. In *Machine Learning* (pp. 123–140). Elsevier. https://doi.org/10.1016/B978-0-12-815739-8.00007-9

Zhang, L., Zhang, Y., & Li, Y. (2021). Mobile Robot Path Planning Based on Improved Localized Particle Swarm Optimization. *IEEE Sensors Journal*, *21*(5), 6962–6972. https://doi.org/10.1109/JSEN.2020.3039275

Zhang, M., Cai, W., & Pang, L. (2023). Predator-Prey Reward Based Q-Learning Coverage Path Planning for Mobile Robot. *IEEE Access*, *11*, 29673–29683. https://doi.org/10.1109/ACCESS.2023.3255007

Appendix A- MATLAB Code for Min-sum message passing

This section provides a general insight into the core implementation of the min-sum algorithm employed in this study. The code presented here begins by loading matrices derived from the waypoint selection stage, a pivotal step in the multi-robot cyclic path planning problem elucidated in Chapter 5 - . The algorithm operates within a structured framework encompassing two essential while loops.

The first loop iterates until the number of undesired subtours is reduced to zero, aligning with the problem formulation outlined in Chapter 5. The variable "constraint" serves as a marker, indicating when the degree paths satisfy the required degree constraints. Consequently, the inner loop executes until this condition holds true, ensuring that all nodes meet the stipulated degree requirements. Within this loop, the computation of new messages from variables to nodes and vice versa takes place. This inter-node communication serves to update values within the matrix μ, ultimately determining the edges forming the final paths.

Central to this process are two key functions: `newF2V_degree_` and `newF2VSubtour`. These functions play a crucial role in computing new messages and disseminating them across the factor graph. They leverage the equations elucidated in Chapter 3 - , forming the backbone of the algorithm's message propagation mechanism.

Upon the completion of the inner loop, the `addSubtourConstraints` function comes into play. This function is responsible for identifying undesirable subtours and introducing new function nodes designed to subsequently eliminate these undesired paths. The entire algorithmic workflow encapsulated in this section represents a fundamental component of the research methodology.

```
% This code runs multiple min-sum algorithm for multi-robot cyclic tour optimization
with CICO park data

load('147centers.mat')            % loading waypoints from previous stage
⋮
⋮
while numberofSubtours~=0 || constraint =="False"
        T           = 0;
        constraint  = "False";

        while T < Tmax && constraint == "False"

           % Update messages from variables to functions
           indexes       = repmat (boolean(eye(N,N)),1,1,tf);
           Total         = sum(mF2V,3);
           mV2F          = repmat (Total,1,1,tf) -mF2V;
           mV2F(indexes) = inf;
```

```
          % Update messages from functions to variables
          % Degree functions
          newF2V(:,:,2) = newF2V_degree_(mV2F(:,:,2),N,1,R);
          newF2V(:,:,3) = newF2V_degree_(mV2F(:,:,3),N,2,R);

          % Subtour functions
          for i=4: tf
              [ngb,noNgb]     = neighbors{i-4,:};
              newF2V(:,:,i)   = newF2VSubtour(mV2F(:,:,i),N,ngb,noNgb,R);
          end

          mF2V        = lam*mF2V+(1-lam)*newF2V;
          mF2V(:,:,1) = D;

          %Find beliefe matrix
          mu         =  sum(mF2V,3);
       end

       [mV2F,mF2V,newF2V,subtours,numberofSubtours,tf,neighbors] =
addSubtourConstraints(mu,subtours,mV2F,mF2V,newF2V,R,distributionrobots);

       if R==1 && numberofSubtours==1
           numberofSubtours = 0;
       end

   end
```