ALGORITHMS FOR PARTICLE SWARMS USING GLOBAL CONTROL: AGGREGATION, MAPPING, COVERAGE, FORAGING, AND SHAPE CONTROL

A Thesis

Presented to

the Faculty of the Department of Electrical and Computer Engineering

University of Houston

in Partial Fulfillment

of the Requirements for the Degree

Master of Science

in Electrical Engineering

by
Arun Viswanathan Mahadev
May 2017

ALGORITHMS FOR PARTICLE SWARMS USING GLOBAL CONTROL: AGGREGATION, MAPPING, COVERAGE, FORAGING, AND SHAPE CONTROL

Arun Viswanathan Mahadev

Approved:

Chair of the Committee
Dr. Aaron T. Becker, Assistant Professor
Department of Electrical and Computer Engineering

Committee Members:

Dr. Nikolaos V. Tsekos, Associate Professor
Department of Computer Science

Dr. Miao Pan, Assistant Professor
Department of Electrical and Computer Engineering

Dr. Suresh K. Khator, Associate Dean,
Cullen College of Engineering

Dr. Badrinath Roysam, Professor and Chair,
Electrical and Computer Engineering

Acknowledgements

My thesis advisor Dr. Aaron T Becker of the Department of Electrical and Computer Engineering at the University of Houston has always been the light guiding the way in all my efforts during my masters study. I have the deepest gratitude for him for allowing me to work on this project and for guiding me whenever I faced challenges along the way.

I would like to take this opportunity to thank my co-authors: Dominik Krupke, Dr. Sándor P. Fekete from Department of Computer Science, TU Braunschweig Shiva Shahrokhi, and Jarrett Lonsford from the University of Houston. The work presented here was possible because of their continuous support and contribution to this project.

This material is based upon work supported by the National Science Foundation under Grant No. [IIS-1553063] and [IIS-1619278] and all of us are thankful to the NSF for supporting our research.

Finally, I would like to thank my first teachers, my parents M.Viswanathan and Rajalakshmi Viswanathan. Their love and support is the driving force for all of my successes. Thank you.

ALGORITHMS FOR PARTICLE SWARMS USING GLOBAL CONTROL: AGGREGATION, MAPPING, COVERAGE, FORAGING, AND SHAPE CONTROL

An Abstract

of a

Thesis

Presented to

the Faculty of the Department of Electrical and Computer Engineering

University of Houston

in Partial Fulfillment

of the Requirements for the Degree

Master of Science

in Electrical Engineering

by
Arun Viswanathan Mahadev
May 2017

Abstract

Targeted drug delivery is a promising technique to reduce the side effects of drugs by delivering them in concentrated doses using large swarms (10^{16}) of controllable microbots only targeting bad or infected tissue. A promising way to control small steerable microbots is by using a global control field such as the magnetic gradient of an MRI machine. In this work we develop benchmark algorithms for performing aggregation of microbots using global control. Using our findings we develop algorithms for a novel approach of mapping tissue and vascular systems without the use of harmful contrast agents in an MRI. In our work we consider a swarm of particles in a 1D, 2D, and 3D grids that can be tracked and controlled by an external agent thus building a map. We present algorithms for controlling particles using global inputs to perform: (1) Mapping, i.e., building a representation of the free and obstacle regions of the workspace; (2) Foraging, i.e., ensuring that at least one particle reaches each target location;and (3) Coverage, i.e., ensuring that every free region on the map is visited by at least one particle. Finally we also demonstrate shape control of large swarms using global control by developing an algorithm for position control.

Table of Contents

Acknowledgements **v**

Abstract **vii**

Table of Contents **viii**

List of Figures **x**

1 Introduction **1**

1.1 The Vision of Nano Medicine 1

1.2 Workspace Setup 4

2 Aggregation **6**

2.1 Related Work 6

2.1.1 Localization 6

2.1.2 Robot Rendezvous 7

2.2 Theory 8

2.2.1 Conditions for Aggregation 8

2.2.2 Optimal Aggregation for Small Particles 9

2.2.3 Greedy Strategy to Aggregate Small Particles 10

2.2.4 Aggregating Large Particles in a Target Region 14

2.2.5 Aggregating Large Particles with Maximal Moves 15

2.3 Results and Inferences 15

3 Mapping **21**

3.1 Related Work 24

3.2 Theory 25

3.2.1 Mapping in 1D 25

3.2.1.1 1D mapping with 1 particle 25

3.2.1.2 1D mapping with n particles 26

3.2.1.3 1D mapping with scan and move costs 27

3.2.2 Mapping in 2D 29

3.2.2.1 2D mapping with 1 particle 29

3.2.2.2 2D mapping with n particles 30

3.2.3 Mapping in 3D 34

3.3 Results and Inferences 34

3.3.1 1D mapping 34

3.3.2 2D mapping 36

4 Shape Control **42**

4.1 Painting a Picture with Global Control 42

4.2 Experiment, Results and Inferences 46

4.2.1 Position Control Algorithm for n Robots 46

4.2.2 Hardware Experiment: Position Control of n Robots 48

5 Conclusion **51**

References **53**

List of Figures

1.1 Drug delivery concept visualization 3

1.2 Types of particles with respect to workspace size 5

1.3 Maximal moves as input 5

2.1 Workspace constraints 9

2.2 Optimal aggregation explained 11

2.3 Optimal aggregation example 11

2.4 Optimal aggregation limitation 12

2.5 Large particles delivered to a target region 14

2.6 Rechability of aggregation using large particles 16

2.7 Optimal algorithm vs greedy algorithm 17

2.8 Convergance speed using the greedy algorithm variants 18

2.9 Performance of greedy algorithm as a function of moves vs freespace . . . 20

3.1 Mapping using global moves: overview 22

3.2 Coverage and foraging in a 2D map 23

3.3 The variables involved in mapping of one a dimensional space 26

3.4 Limitations of greedy algorithm: Alg. 5 33

3.5 Using DFS as opposed to using Alg. 5 33

3.6 Mapping a 3D space 35

3.7 Probability distribution of maximum gap vs number of particles in 1D . . 35

3.8 Coverage and mapping performance in 1D 36

3.9 Performance of Closest Frontier for mapping different types of maps in 2D 37

3.10 Comparing the three algorithms for 100 iterations of n=1000 38

3.11 Comparing mean and standard variation of number of moves taken vs number of particles for the three algorithms 39

3.12 Mapping vs coverage vs foraging 40

3.13 Distribution effects on performance 41

4.1 Recreating art with 862 particles 43

4.2 Workspace setup for n-particle position control 44

4.3 Drift move representation 45

4.4 Moves vs no. of particles for n-particle positioning 47

4.5 No. of moves depends on minimum clearance ϵ 47

4.6 Hardware description 49

4.7 Hardware experiment showing letter 'U' formed using 7 particles 50

Chapter 1

Introduction

1.1 The Vision of Nano Medicine

The potential use of nano medicine as a tool in medical interventions has given rise to possibilities such as nano machines, targeted drug delivery and better contrast agents. The field in nano medicine of interest in our work is active targeting or active manipulation of particles to achieve desired intervention results. Targeted drug therapy is a goal for many interventions, including treating cancers, delivering pain-killers, and stopping internal bleeding. Using concentrated dosage to medicate affected tissue helps in two ways. First, higher concentration of drugs can be administered for treatment. Second, drug side effects on healthy tissue is much lesser since healthy tissue are avoided. Both would lead to faster treatment and recovery.

Chemists, biologists, and roboticists have shown the ability to produce very large populations (10^3-10^{14}) of small scale(10^{-9}-10^{-6} m) robots using a diverse array of materials and techniques [1, 2, 3]. These particles are easily replacable, dispensible and are compatible for use in the human body, hence making them perfect for the application. It is also possible to mass produce particles with a magnetic core and a catalytic surface for carrying medicinal payloads [4, 5]. The challenges of using such tiny particles is that there can be no on-board computation on these particles and they cannot communicate between each other. Particles of such small size have all attributes we see in low Reynolds number. Low Reynolds number here referees to the setting where we observe motion of particles to be similar to that of steady flow. Effects of viscosity dominate effects of inertia because of the small size of vessles and particles themselves and so particles are

more susceptible to magnetic and electrostatic force compared to gravity. We can control the microbot particles by a global external input, such as an applied fluidic flow, electric field and magnetic gradient. In particular, real-time MRI scanning could allow feedback control using the location of a swarm of these particles. Navigating one particle through the vascular network is just be a simple path planning problem. Steering magnetic particles using the magnetic gradient coils in an MRI scanner was implemented in [6, 4]. 3D Maxwell-Helmholtz coils are often used for precise magnetic field control [7]. These are examples where only one or two particles were manipulated. But controlling swarms of particles using a global input is challenging. The particles are under-actuated as all particles move in the same direction vector when commanded. Also complicated maze-like vascular networks have many obstacles which make steering and path planning difficult as shown in Fig. 1.1.

The second potential application of developing control algorithms for globally controlled particles is mapping. In MR imaging, some tissues have poor *contrast*, which means that the boundaries between tissue types cannot be determined. To discover tissue boundaries, particulate solutions of a contrast agent are used to illuminate regions of interest [8]. Drawbacks include that the contrast agent diffuses quickly and must be injected repeatedly during long scans. Additionally, many contrast agents such as gadolinium chelates are toxic, and prolonged exposure causes medical complications [9].

A safer way is to use biocompatible particles with a magnetic core and a catalytic surface for carrying medicinal payloads [4, 5]. An alternative is *superparamagnetic iron oxide microparticles*, 9 μm particles that are used as a contrast agent in MRI studies [7]. These particles can be steered by the global magnetic gradient of an MRI and visualised by the MRI [10], even when the tissues they move through have poor contrast.

A recent survey on challenges related to controlling multiple micro robots (less than 64 robots at a time), [11] shows us how even controlling a small number of particles

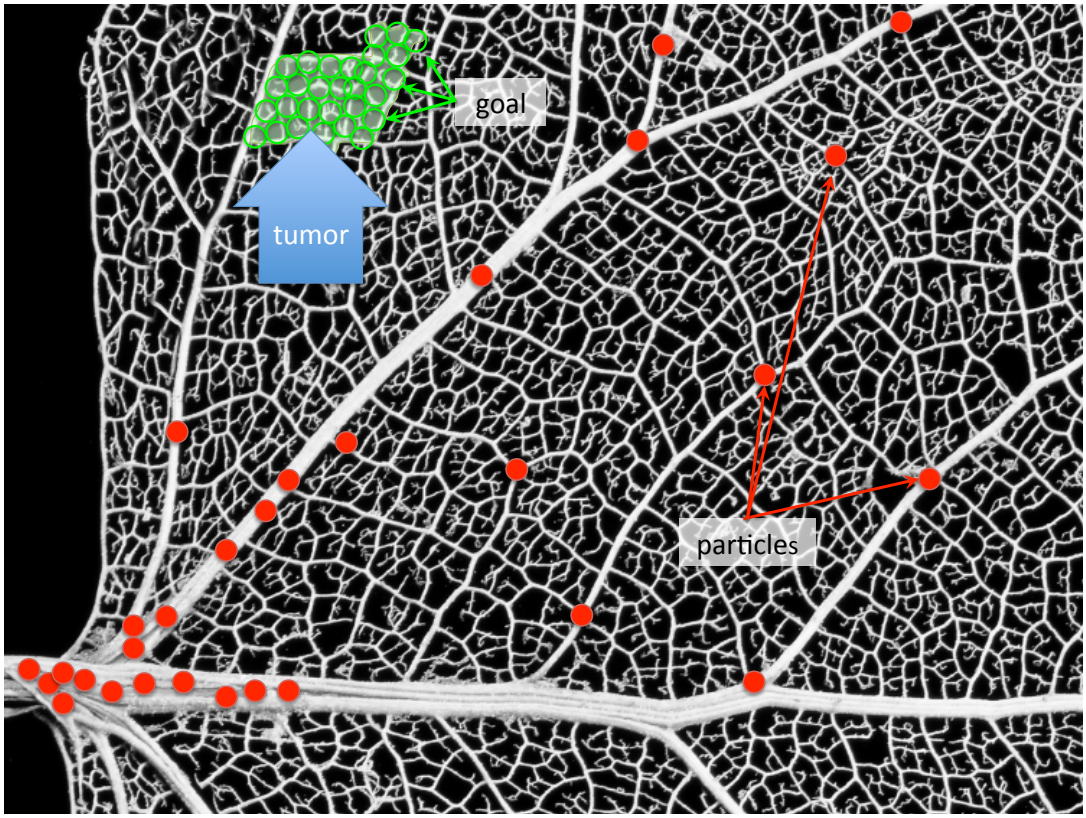


Figure 1.1: Vascular networks are common in biology such as the circulatory system and cerebrospinal spaces. Navigating a swarm using global inputs, is challenging due to the many obstacles.

can be difficult. In our work we tackle the challenges of developing control algorithms for large swarms of globally controlled particles which has inspired a large body of work on global control like [12],[13] and [14]. We draw inferences based on the behavior of particles in mazes and provide theoretical reasoning for understanding the results of our study. Using global control we explore aggregation of large number of dispersed particles in chapter 2 which is an extension of our work published in [15]. We extend these ideas to develop control algorithm for mapping, coverage, foraging discussed in chapter 3 and finally accurate position control using global control in chapter 4. Chapter 3 is an expansion on the work done for [16] and chapter 4 is an expansion on the work done for [17]. Both [16, 17] have been submitted for publication in IROS 2017 and are currently under review.

1.2 Workspace Setup

In the 2D case a completely connected planar grid *workspace* W is filled with a number of robots (each occupying one cell of the grid). Each unit square in the workspace is either *free*, which a robot may occupy or *obstacle* which a robot may not occupy. Obstacles present do not disconnect the free-space.

The 'robots' in this work are simple particles. We consider two types of particles, small and large, as depicted in Fig. 1.2. If particles are much smaller than the workspace geometry, we call them small. We represent each grid cell as filled if it contains at least one particle and empty otherwise. A cell filled with small particles can combine with another filled cell. If particles are the same size as workspace grid-cells, we call the particles large. Large particles cannot combine. The presence of a large particle in a cell prevents another particle from entering.

All robots are commanded in unison: the valid commands are "Go Up" (u), "Go Right" (r), "Go Down" (d), or "Go Left" (l). We consider two classes of commands, discrete and maximal moves. *Discrete moves*: robots all move in the commanded direction one unit unless they are prevented from moving by an obstacle or a stationary robot. *Maximal moves*: robots all move in the commanded direction until they hit an obstacle or a stationary robot. For maximal moves, we assume the area of W is finite and issue each command long enough for the robots to reach their maximum extent 1.3. A command sequence $\boldsymbol{m}$ consists of an ordered sequence of moves m_k, where each $m_k \in \{u, d, r, l\}$. A representative command sequence is $\langle u, r, d, l, d, r, u, \ldots \rangle$.

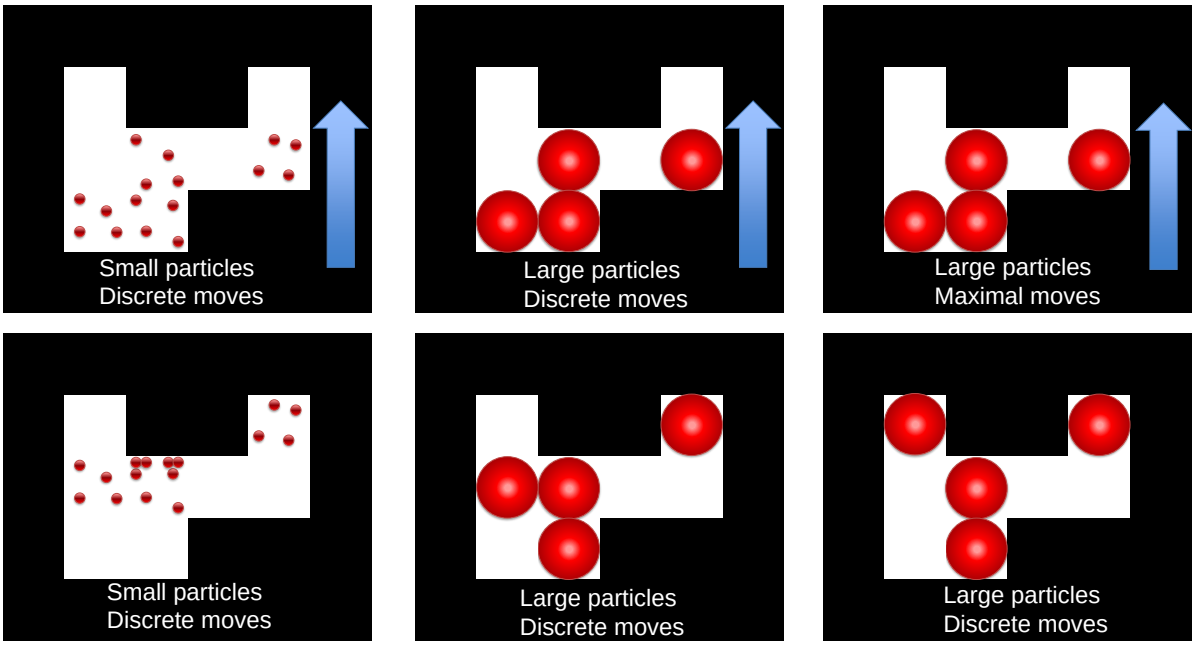


Figure 1.2: If particles are much smaller than the workspace geometry, we call them *small*. If particles are the same size as workspace gridcells, we call the particles *large*. Large particles cannot combine.

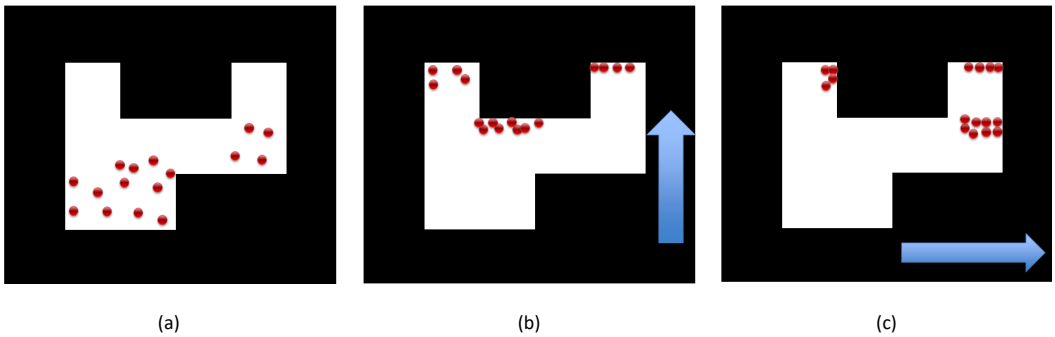


Figure 1.3: Particles distributed in a bounded space (a) are subjected to uniform input to go 'up' (b) until all particles reach their limit, similarly in (c) the input is a uniform 'right' input.

Chapter 2

Aggregation

The old adage *toxicity is a function of concentration* explains that often we can flow a diluted drug through the body without much side effects to the body. Our efforts are focused on building algorithms that can take such a dispersed particle swarm and then aggregating the particles to a location and thus delivering the required payload. Building on the techniques for controlling many simple particles with uniform control inputs presented in [18, 19] we also outline new research problems. We study two notions of aggregating a swarm, in correspondence with particle size with relation to workspace: 1) Small particles: the swarm is aggregated when all particles share the same (x, y) coordinates. 2) Large particles: Here the swarm is aggregated when it forms one *connected component*. A *connected component* is a set of particles P such that for any two particles in P, there is a sequence of neighboring particles that connect them. 2D cells are neighbors if they share an edge, 3D cells are neighbors if they share a face.

2.1 Related Work

The strongest parallels in robotics literature to aggregating particles are particle localization and particle rendezvous. It has applications to drug delivery and micro manufacturing.

2.1.1 Localization

In localization a robot must use the data available to it by means of observation to estimate it's location. For example, a mobile robot with a map of the workspace must

localize itself, using only a compass and a bump-sensor that detects when the robot contacts a wall. Given an environment, finding a localizing sequence is framed as a planning problem with an unknown initial state and an unobservable current state. The solution in [20] was to transform the problem from an unobservable planning problem in state space to an observable problem in a more complex information space. The concepts from [20, 21] have been extended to robots with bounded uncertainty in their inputs [22]. They provided a complete algorithm, but generating an optimal localizing sequences remains an open problem. [20] assumes there is only one robot, but it still gives us clarity on how planning for independent robot systems differ from swarm robot systems. Also related is work on sensorless part orientation, where flat tray is tilted in a series of directions to bring a polygonal part, initially placed at random orientation and position in the tray, to a known position and orientation [23].

2.1.2 Robot Rendezvous

In *robot rendezvous*, planning is focused on finding paths for two or more independent, intelligent agents to meet. The introduction to this concept by Alpern and Gal [24] introduced a wide range of models and methods for this concept as have Anderson and Fekete [25] in a two-dimensional geometric setting. As in our work it is assumed that the topology is bounded and robots have limited onboard computation. This is relevant to maneuvering particles through worlds with obstacles and implementation of strategies to reduce computational burden while calculating distances in complex worlds [26].

The main variation that occurs in this field of research lies on how a successful rendezvous is defined. It can be defined as "The time taken for successful transitive rendezvous between all agents" [26]. Or it can be considered as "aggregation of all agents to a given spot' [27].

2.2 Theory

With discrete inputs and small particles, the problem can be reduced to localizing a sensorless robot in a known workspace. This is similar to work on draining a polygon [28], or localizing a blind robot [20, 21], but with discrete inputs. In Section 2.3, for the small particle problem we present an optimal aggregation algorithm, Alg. 1 in Section 2.2.2 and a greedy aggregation policy Alg. 2 in Section 2.2.3. We also give positive (Section 2.2.4) and negative (Section 2.2.5) results for large particles.

2.2.1 Conditions for Aggregation

Particles initialized in two unconnected regions of a work-space cannot be aggregated. The proof is trivial, since a path does not exist without connectivity. Such a configuration is depicted in Fig. 2.1*a*.

Under *maximal inputs* aggregation is not possible in certain cases where there are bottlenecks. Such a world can be constructed with spaces resembling bottles or fish weirs from which a single particle cannot escape, as shown in Fig. 2.1*b*. If the free space contains at least two such bottles with at least one particle in each, the swarm cannot be aggregated with maximal inputs.

The world must be bounded. Two initially separated particles in an unbounded world without obstacles cannot be aggregated; however with discrete inputs, one obstacle is sufficient as seen in Fig. 2.1*c* and can be inferred from previous work on global control presented in [12].

A swarm of *small particles* can be aggregated on any bounded grid if controlled by discrete global commands. However, there are configurations where sometimes aggregation is impossible with large particles, as seen in Fig. 2.1*d*. Here if we move right, the particle in the bottom right is disconnected and if we move left, the particle on the bot-

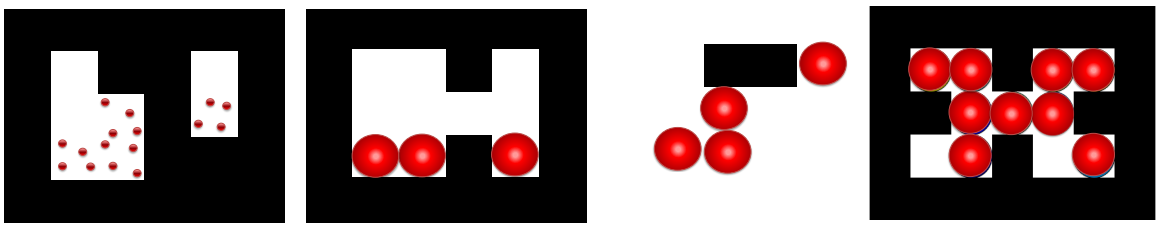


Figure 2.1: Examples of workspaces for which aggregation is not possible. (a) The swarms are in unconnected components (b) Bottle neck environment (c) An unbounded world with a single obstacle. (d) Connected component issues

tom left will be disconnected. So, aggregation is complicated and sometimes impossible in large particle scenario.

2.2.2 Optimal Aggregation for Small Particles

The optimal move sequence to aggregate all particles in a workspace can be determined by a simple strategy of growing and pruning a configuration tree that expands the tree of all possible movement sequences in a breadth-first search manner, and halts when all the particles aggregate at one point. Alg. 1 implements this breadth-first-search technique. It initializes a tree where each node contains the configuration of particle locations $C[p]$, the move that generated this configuration $M[p]$ and a parent configuration pointer $P[p]$. Here C, M, P are the respective complete lists. p is the current iteration pointer and e is the end of list pointer. The root node is $\{C_0, \varnothing, 0\}$, where C_0 is the initial configuration of particle locations. We then construct a breadth-first tree of possible configurations $\{u, r, d, l\}$, pruning configurations that already exist in the tree. We stop when the cardinality at a leaf is one, $|C_i|= 1$, which indicates that the swarm has been aggregated as represented in Fig. 2.2. For a simple case where n=2 and m=5 the optimal move sequence is found by growing and pruning the tree in each level by cutting off nodes which give a previously seen configuration and keeping the nodes which produce a new configuration. Algorithm terminates when a convergence occurs . Here,

the number of moves in the end will be the same as the number of levels expanded. This algorithm produces the optimal path to determine the shortest path length 's' as seen in Fig. 2.3, but requires $O(4^s)$ time to learn and $O(4^s)$ memory and the graph grows exponentially (Fig. 2.4). This leads us to investigate other algorithms which will solve the path with much lower computational time and data.

Algorithm 1 Optimalaggregating(W, C_0)

$p \leftarrow 1$
$\{C[p], M[p], P[p]\} \leftarrow \{C_0, \varnothing, 0\}$ ▷ initialize
$e \leftarrow 1$
while $|C[p]| > 1$ **do** ▷ more than 1 unique position
    for $m = \{u, d, r, l\}$ **do**
        $C_{temp} \leftarrow \text{ApplyMove}(C[p], m)$
        if $C_{temp} \notin C$ **then** ▷ add node to list
            $e \leftarrow e + 1$
            $\{C[e], M[e], P[e]\} \leftarrow \{C_{temp}, m, p\}$
        end if
    end for
    $p \leftarrow p + 1$ ▷ get next configuration
end while
$path \leftarrow \{\}$ ▷ construct optimal path
while $P[p] > 1$ **do**
    Append $M[p]$ to $path$
    $p \leftarrow P[p]$
end while
$path \leftarrow \text{Reverse}[path]$

2.2.3 Greedy Strategy to Aggregate Small Particles

Two small particles in a finite and connected polyomino can be aggregated with discrete movement by simply repeatedly moving one particle onto another in the shortest way. The corresponding procedure AGGREGATEAB is described in Alg. 2. By iteratively aggregating any two disjoint particles, the distinct positions of the particle swarm can be reduced until all particles are at the same position. The two particles can be chosen with different methods and we implement the following methods:

1. Closest pair of particles - choose a pair of particles with the minimum distance between them.

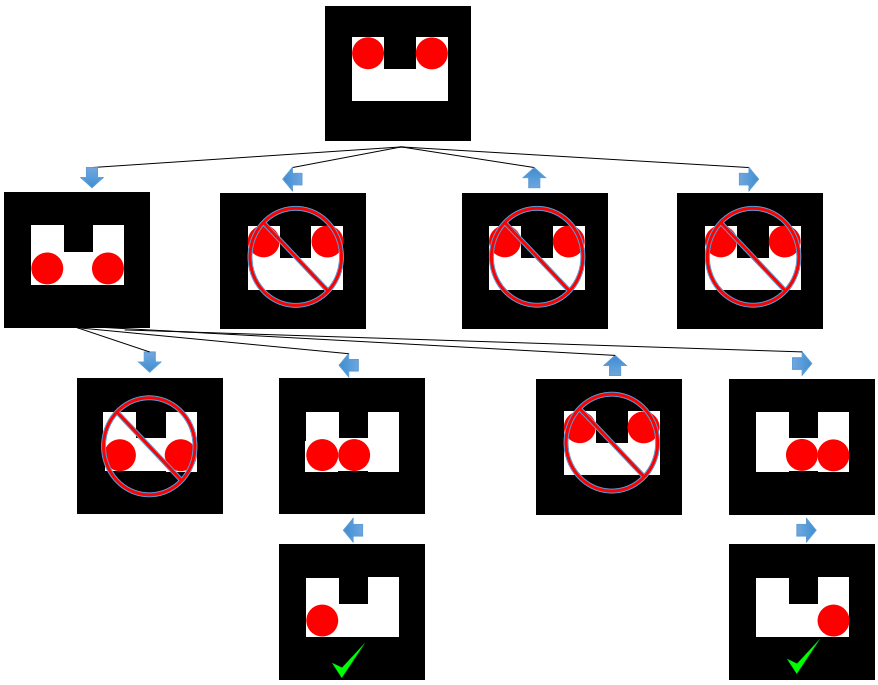


Figure 2.2: Optimal algorithm grows and prunes tree of configurations. Growth happens in nodes which show new configuration.

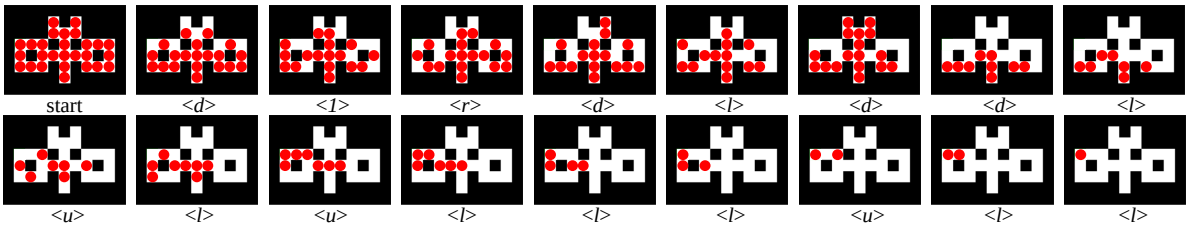


Figure 2.3: The optimal solution for a world with 27 free spaces, which required expanding 423,440 nodes with an optimal path (shown) taking 17 moves.

2. Furthest pair of particles - choose a pair with maximum distance between them.

3. Connect to first - choose the first two particles while searching for particles in the workspace from top left to bottom right.

4. Random combinations - choose any two particles.

5. First to last - choose the first particle and last particle, i.e., the leftmost top and rightmost bottom particles, respectively.

AGGREGATEAB can be called to implement any of these methods.

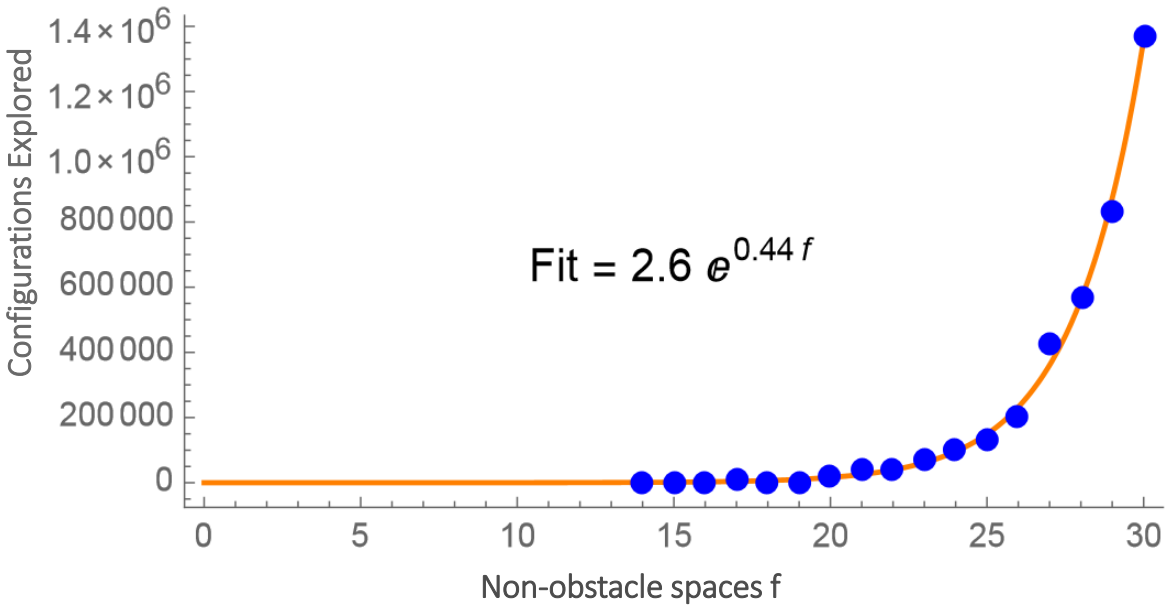


Figure 2.4: A workspace with 30 free spaces required 1.6 million nodes before finding the optimal solution.

Algorithm 2 Aggregating two particles that can overlap

Require: a can reach b, Polyomino is bounded

1: **procedure** AGGREGATEAB(a: Particle, b: Particle)
2: **while** $dist(a, b) \neq 0$ **do**
3: Let $\mathcal{C} \in \{u, d, l, r\}^N$ be the shortest control sequence that moves a onto $pos(b)$
4: Execute $\mathcal{C}$
5: **end while**
6: **end procedure**

Theorem 1. AGGREGATEAB *aggregates two particles in a polyomino with* $O(n^3)$ *discrete control commands, where* n *equals the polyomino's height times its width.*

Proof. The distance between a and b equals the length of $\mathcal{C}$. After execution of $\mathcal{C}$, the distance has not increased as a is now on the previous position of b and b has at most moved $|\mathcal{C}|$ units from it. If b had a collision during the execution of $\mathcal{C}$, the distance is even less as at least one command did not result in a move of b. As $dist(a, b) \in O(n)$, only $O(n)$ loop iterations with collisions are needed to aggregate a and b. Obviously, $|\mathcal{C}| \in O(n)$ and hence every loop iteration executes at most $O(n)$ commands. With every iteration without collision, the positions of a and b change each by $pos(b) - pos(a)$. This

difference only changes if b had a collision, therefore the particles move in the same direction with every collision-free iteration. After $O(n)$ collision-free iterations of the loop, b must have a collision, as the polyomino is finite. This results in $O(n^2)$ commands to reduce the distance by at least one and thus $O(n^3)$ commands suffice to aggregate a and b. □

Theorem 2. AGGREGATEAB *has a computational complexity of* $O(n^3)$.

Proof. The shortest control sequence $\mathcal{C}$ can be calculated in $O(n)$ time by a simple breadth-first-search. Under the assumption that a command can be executed in $O(1)$, one loop iteration has a computational complexity of $O(n)$. With $O(n^2)$ loop iterations (see proof of Theorem 1), this results in an overall complexity of $O(n^3)$. □

Theorem 3. *A particle swarm of size* $O(m)$ *can be aggregated with* $O(m.n^3)$ *discrete control commands and a computational complexity of* $O(m.n^3)$ *where* n *equals the polyomino's height times its width.*

Proof. Select two disjunct particles and execute AGGREGATEAB. This reduces the size of distinct positions in the particle swarm by one. After $O(m)$ executions, there is only one position left and the particle swarm is aggregated. □

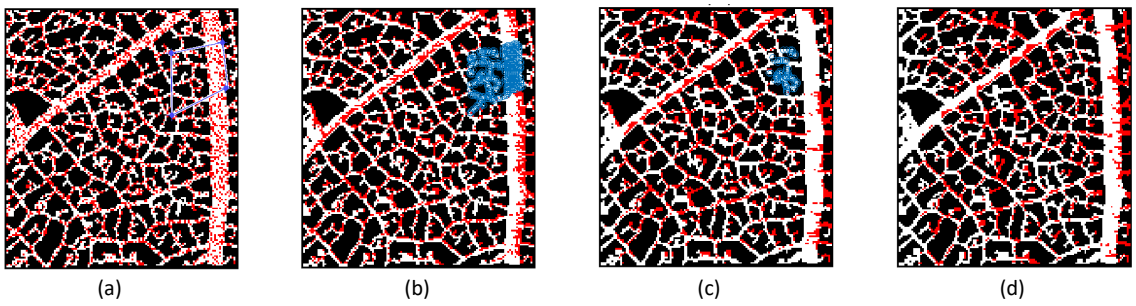


Figure 2.5: We define a target stick target(a). A modified aggregation algorithm is used to deliver particles to the sticky target which metabolizes the particles as they reach the target as seen in (b),(c) and (d).

2.2.4 Aggregating Large Particles in a Target Region

In the previous two subsections, the particles are relatively small, allowing several to be aggregated in the same location anywhere in the environment. But when the particles are relatively large, they may block each other's way, making the motion control trickier. We can still deliver a swarm of particles to a defined target region(Fig. 2.5(a)) by making use of discrete moves, assuming that particles are metabolized once they reach the target region (Fig. 2.5(b),(c),(d)), i.e., the target is "sticky".

This implies that they stay within the target region once they get there, and that they do not block each other within that region.

Theorem 4. *For a sticky target region and large particles within an environment of diameter D, a particle swarm of size $O(m)$ can be aggregated with $O(m.D)$ discrete control commands.*

Proof. The proof is straightforward by induction. Moving one particle to the target region takes at most D moves, which leaves all other particles within distance D. □

Note that the extent of the environment is critical. If we are dealing with k particles within an environment of size $m \times m$, then we get the following.

Corollary 5. *For a sticky target region and large particles, a particle swarm within an environment of size $m \times m$ can be aggregated with $O(m^3)$ discrete control commands.*

If all particles of the swarm are relatively close to the target region, the complexity can be stated differently.

Corollary 6. *For a sticky target region and large particles, a particle swarm of size $O(k)$ that fills a square environment around a target region can be aggregated with $O(k^{3/2})$ discrete control commands.*

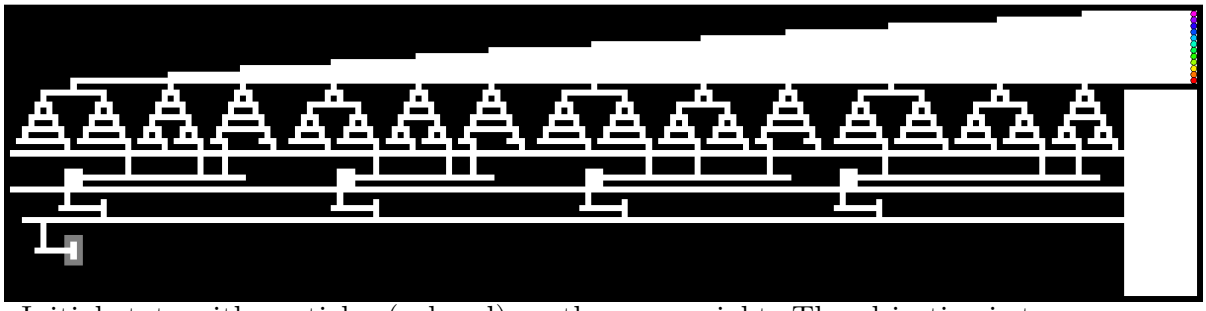
2.2.5 Aggregating Large Particles with Maximal Moves

Our results rely on being able to limit the extent of the motion, i.e., having *discrete moves*. If that is not the case, i.e., in the case of *maximal moves*, in which each particle moves until it is stopped by an obstacle or another stopped particle, the problem becomes considerably harder and may indeed be intractable. As we showed in previous work [29], deciding whether even a *single* particle can be delivered to a target region (Fig. 2.6) is already an NP-hard problem; this implies not only that finding a solution is computationally hard, but that there are instances in which no solution exists.

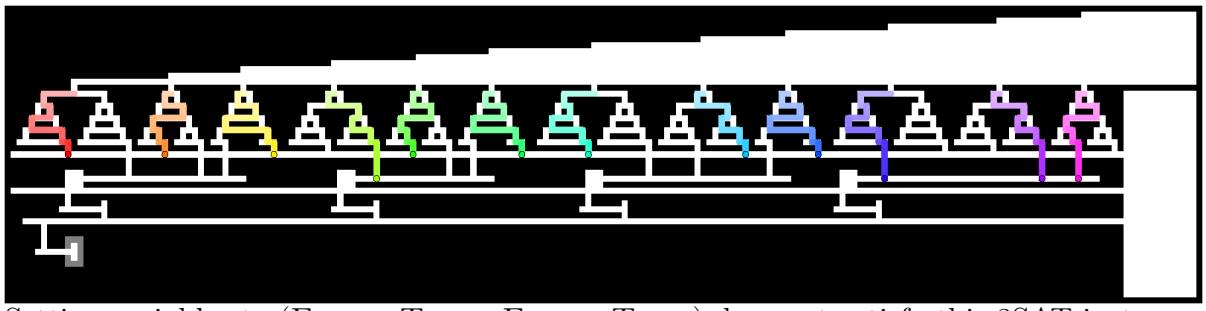
2.3 Results and Inferences

Our first experiment compares the optimal algorithm (Alg. 1) versus three varients of the greedy algorithm (Alg. 2). Fig. 2.7 compares the number of moves required to converge for the optimal strategy (Alg. 1) versus the greedy strategy for small worlds ranging from 5 free spaces to 30 free spaces.

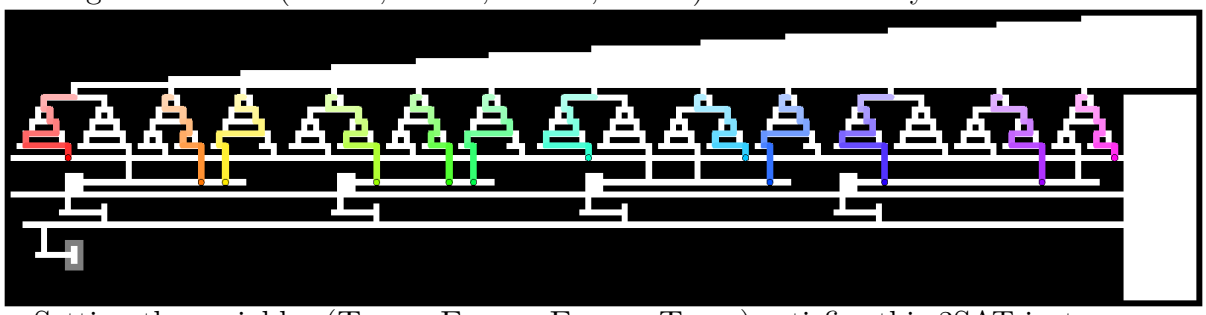
Workspaces with more spaces were not considered because the optimal algorithm requires a lot of time due to the exponential time to free space relationship. This plot shows that the optimal algorithm requires approximately half the moves of the greedy



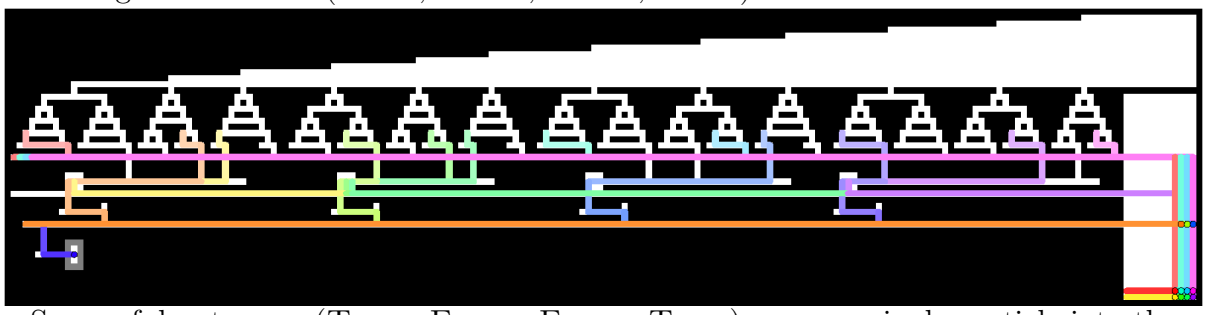
Initial state with particles (colored) on the upper right. The objective is to move one particle into the grey target rectangle at lower left.



Setting variables to (FALSE, TRUE, FALSE, TRUE) does not satisfy this 3SAT instance.



Setting the variables (TRUE, FALSE, FALSE, TRUE) satisfies this 3SAT instance.



Successful outcome. (TRUE, FALSE, FALSE, TRUE) moves a single particle into the target region.

Figure 2.6: Combining 12 variable gadgets, four 3-input OR gadgets, and a 4-input AND gadget to realize the 3SAT expression $(\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee x_3)$.

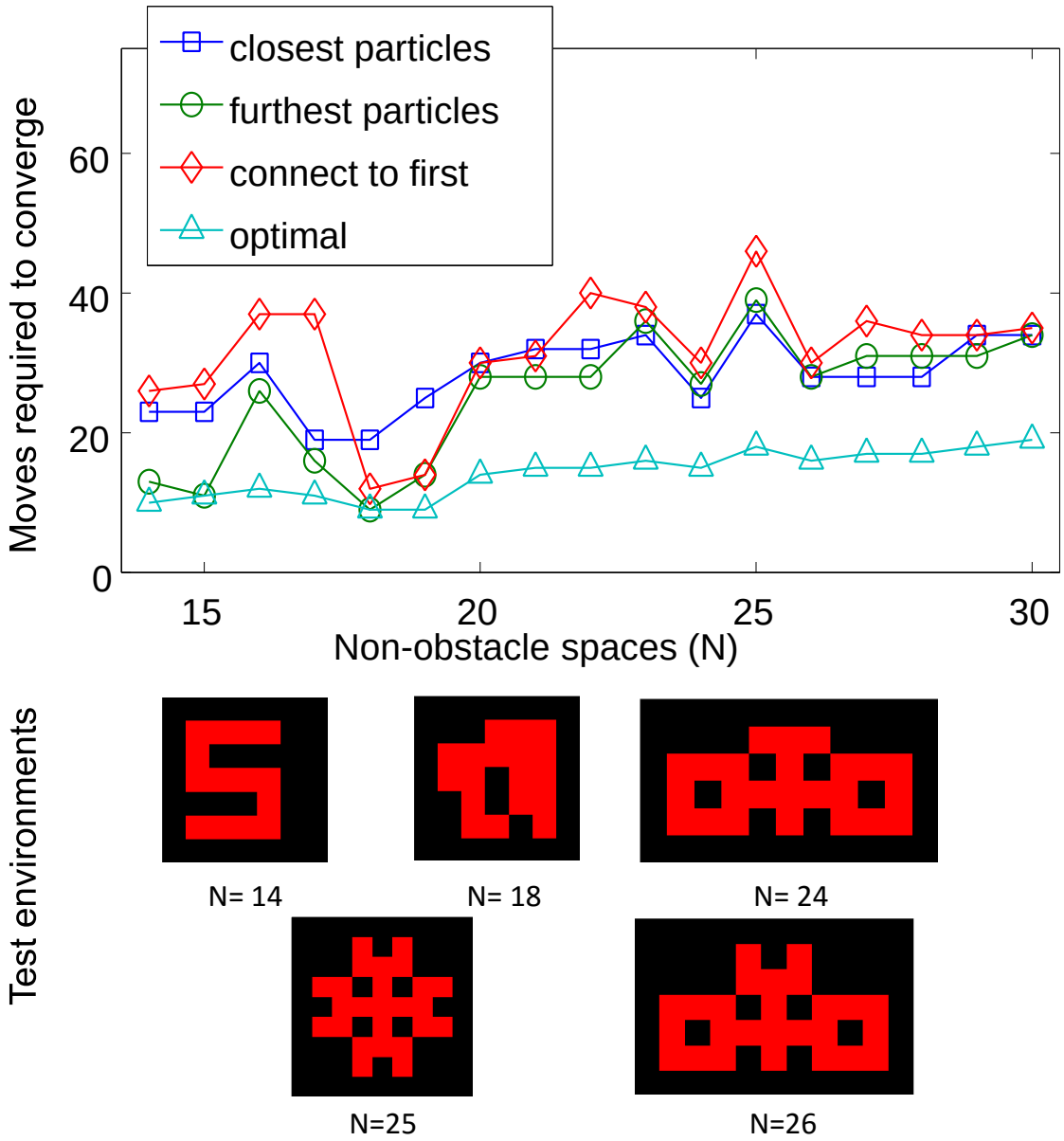


Figure 2.7: The greedy strategy requires 1.95 as many moves as the optimal strategy in a test with 17 different test environments. Below the plot are examples of some of the test environments used.

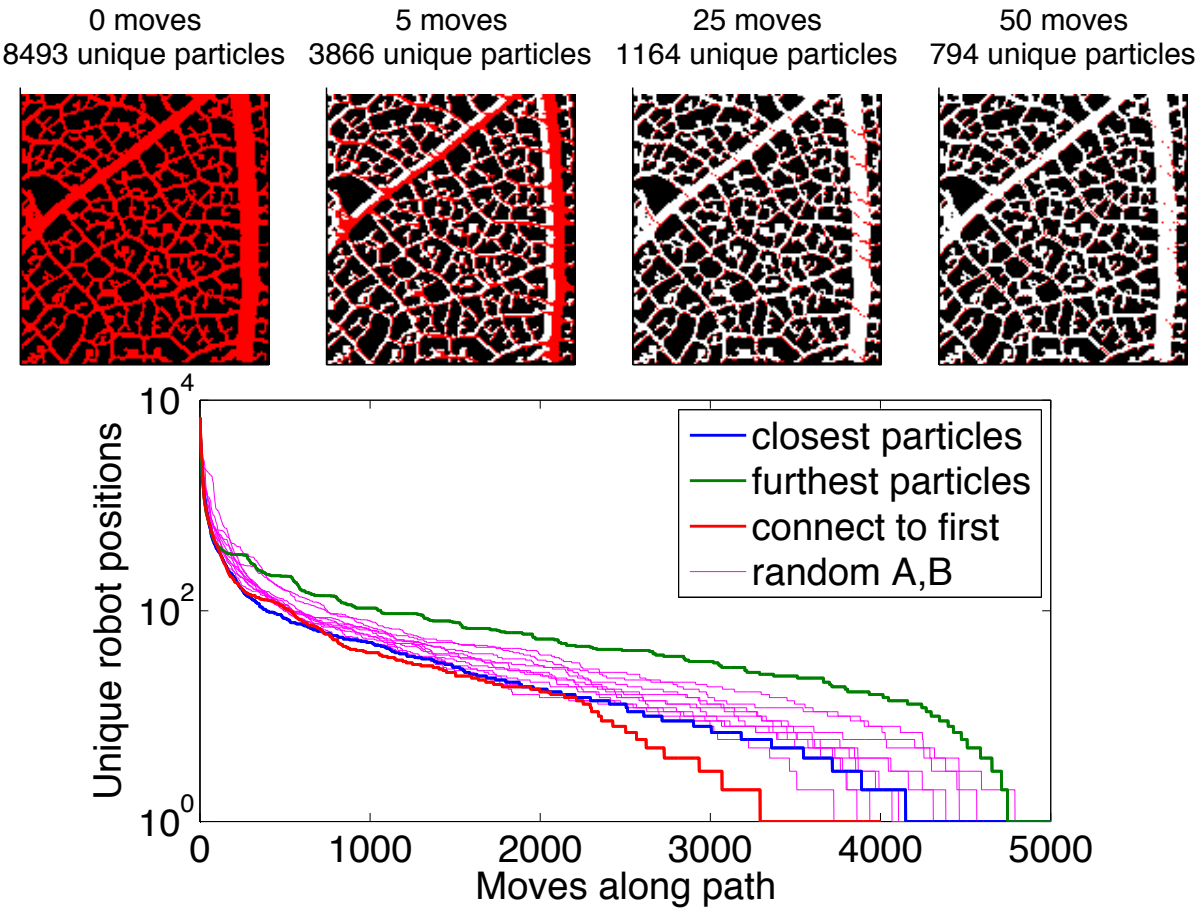


Figure 2.8: Aggregating the furthest particles (green) performs poorly. Connecting the two closest nodes (blue) is better, but both strategies are beat by the strategy that chooses the first two (the top-most, left-most) particles each iteration.

algorithms. The plot has an upward moving trend in general as the number of free spaces increases, but there are local minimums corresponding to easier configurations, which leads to downward spikes in the plot. The number of moves taken to completely aggregate particles also depends on the complexity of the workspace and does not completely depend on the number of free spaces.

For small workspaces the best result among the greedy algorithms changes and so we cannot determine which is the best using small workspaces. To further compare the greedy strategies, we tested the algorithms on larger workspaces. The largest workspace in Fig. 2.8 has 8,493 non-obstacle positions. Fig. 2.8 demonstrates that choosing which particles to pairwise aggregate in Alg. 2 has a large impact on convergence time.

We conducted a comparison study between the number of moves and the resulting unique particles. As discussed earlier in Alg. 1 (Section 2.2.2) and Alg. 2 (Section 2.2.3), getting the number of unique particles down to 1 signifies completion of the aggregating algorithm. In the leaf vascular network, the majority of particle aggregation occurs during the first steps, with a long tail distribution to aggregate the final particles, as shown in the top row of images. Fig. 2.8 shows that *connect to first*, discussed in Section 2.2.B, outperforms the other algorithms. This can also be validated by further testing to compare the three greedy algorithms on larger workspaces.

We simulated bounded worlds of varied sizes from 500 free spaces to 8,493 free spaces. The results are represented in Fig. 2.9, based on Fig. 1.1 because biological vasculatures are our goal application. This graph plot has a smoother trend compared to the plot in Fig. 2.7 because the ratio of free space to node complexity is similar for worlds in Fig. 2.9. The important observation is the consistency that *connect to first* performs best. This validates *connect to first* as the best of the compared algorithms. This is good news because unlike the other two techniques, which involve distance calculation between all pairs of particles, there is negligible calculation involved in the *connect to first* algorithm. The data for which two particle are in top-leftmost location is readily available from the row, column indices of the particles.

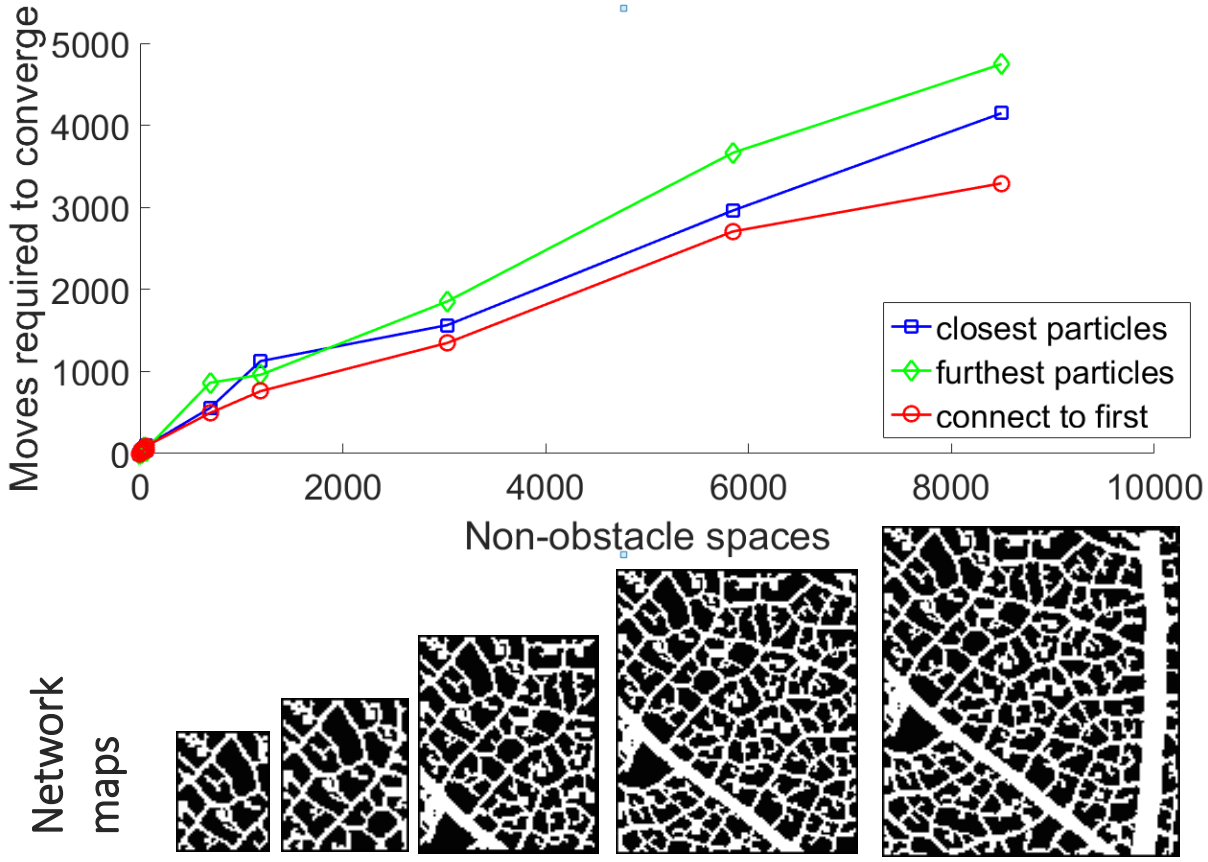


Figure 2.9: Comparing the required number of moves to number of Non-obstacle spaces for aggregation using three of the greedy algorithms in (Alg. 2)

Chapter 3

Mapping, Coverage and Foraging

In our previous work [15] and the previous chapter 2, we provided an algorithm that guarantees the collection of particles. In this work we explore the field of mapping, coverage, and foraging using globally controlled particles. These tasks relate to a large body of previous work from robot navigation, both from theory and practice, which is based on individual control. We propose a novel approach to mapping tissue and vascular systems without the use of contrast agents, based on moving and measuring magnetic particles. To this end, we consider a swarm of particles in 1D, 2D and 3D grids that can be tracked and controlled by an external agent. Control inputs are applied uniformly so that each particle experiences the same applied forces. The algoriths discussed are for performing the three related tasks: (1) *Mapping*, i.e., building a representation of the free and obstacle regions of the workspace; (2) *Coverage*, i.e., ensuring that every free region on the map is visited by at least one particle (Fig. 3.2 (a), (b)). (3) *Foraging*, i.e., ensuring that at least one particle reaches each of a set of desired locations (Fig. 3.2 (c), (d)).

This theoretical discussion is based on discrete 2D workspaces and it is also shown that it scales to the 3D workspace. Fig. 3.1 represents the complete mapping of a workspace using a large number of particles. After 82 moves, the 300 particles have mapped the free cells, while the 4 particles require a total of 2394 moves to fully map the free cells. At the initial step, all particles (red circles) are in free cells (white squares) and are surrounded by blue squares that represent the unknown frontier cells. By commanding the particles to take one step in a particular direction, we can categorize the the frontier cells in this direction as either obstacle or free. If the particle was able to

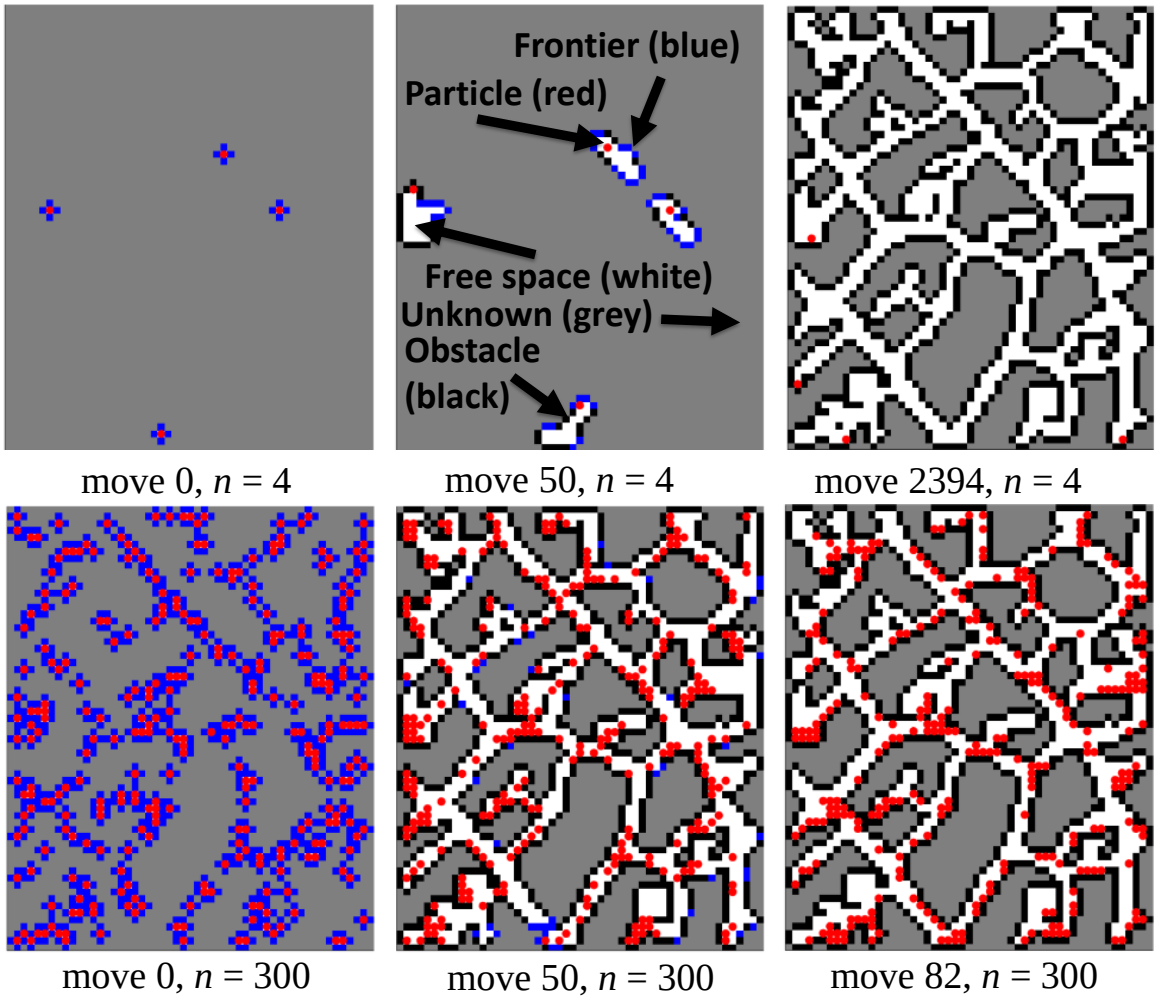


Figure 3.1: Mapping a 2D environment with 1200 free cells using $n = 4$ (top) and $n = 300$ (bottom) particles.

move, that frontier cell is labeled as free, and new frontier cells are added to adjacent areas that have not been mapped. If the particle was unable to move, that frontier cell is labeled as obstacle. The goal is to explore all frontier cells, thereby discovering all connected free cells and the obstacles that surround them.

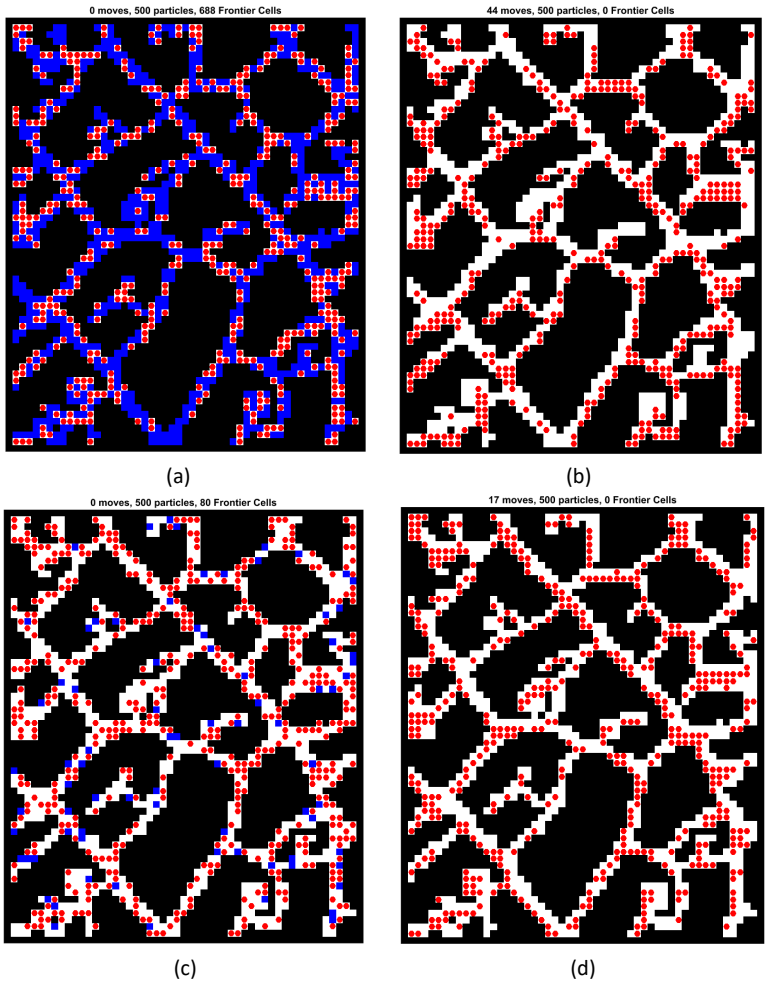


Figure 3.2: Coverage (top) and Foraging (bottom) of a 2D environment with 1200 free cells using $n = 500$ and $n = 500$ particles.

This Chapter is arranged as follows. After a review of recent related work in Sec. 3.1, we introduce the algorithms to perform mapping, coverage, and foraging and also discuss the complexity in Sec. 2.2. In Sec. 3.3 we discuss the performance of the algorithms on parameters which determine efficiency based on different environments, particle distribution and completion speed. We conclude this chapter by summarizing the results.

3.1 Related Work

Coverage using one robot is a canonical robotics problem [30]. It has been studied in-depth for many applications including lawn mowing, harvesting, floor cleaning, 3D printing, robotic painting, and others.

Coverage means the robot has passed within one robot radius of every location in the workspace. Coverage with a swarm of robots is key for a range of applications because swarms have higher fault tolerance and reduce completion time. Correspondingly, it has been studied from a control-theoretic perspective in both centralized and decentralized approaches. For examples of each, see [31] and [32].

Previous methods focus mostly on extending single robot coverage techniques to multi-robot systems. Solving coverage for synchronous multi-robots using on-line coverage techniques such as the boustrophedon technique of subdividing the 2D space into cells as in [33] focuses on moving the robot teams in unison until they identify obstacles in their path. Once that happens, the team divides into smaller teams that continue the search in the smaller cells. The robots mentioned have similarity with our particles in the sense that robots try to move in the same direction as long as possible. In our problem of interest through the particles always move in the same direction. The frontier cells exploration mentioned in [34] is an algorithm that is highly explorative as target locations to expand are selected using information from each robot. Many algorithms have been developed after this pioneering work, and it showed how algorithms for single robot could be expanded to multi-robot systems. The explorative bias of the technique allows it to define target cells of high priority to explore.

Of particular relevance for the content of our work is Fekete et al. [35, 36] for online searching by an autonomous robot in an unknown environment, where both moving and measuring incur individual costs, and Fekete et al. [37] for an (offline) setting that studies the closely related bicriteria version of covering with travel cost. For another recent work

on mapping and coverage by a swarm of robot with limited information and capabilities, see [38].

However, all these approaches assume a level of intelligence and autonomy in individual robots that exceeds the capabilities of many systems, including current micro- and nano-robots. Current micro- and nano-robots, such as those in [11, 39, 40] cannot have onboard computation. Thus, we will be referring to them as *particles*.

3.2 Theory

In this section we analyze the mapping in 1D, 2D and 3D environments usiing discrete global inputs.

3.2.1 Mapping in 1D

We begin with the single particle case, then proceed to the n particle case.

3.2.1.1 1D mapping with 1 particle

We consider a particle initialized randomly in a linear map that has m free-spaces. To *map* this region the particle is moved in a choose direction, until it registers a boundary, then the direction is swithched and particle is moved to the other boundary.

Without loss of generality, assume the particle always starts going left, and label the free-space from 1 to m left to right. If the initial position is 1, the particle tries to move 1 unit to the left, but is stopped by the boundary. The particle then moves $m-1$ moves to the right. The final m^{th} move right results in a collision with the right wall, and thus mapping requires $m+1$ moves. This is the minimum number of moves. The worst case is if the particle starts at m, requiring $2m$ moves: m moves to the left and m moves to the right.

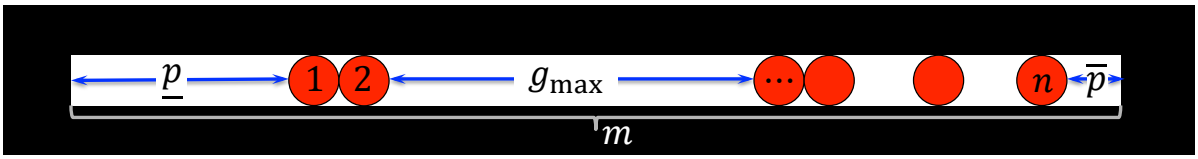


Figure 3.3: Exploring a 1D environment of size m with n particles. Here $m = 20$ and $n = 6$. $\underline{p}$ =4, $\bar{p}$ =1 and $\bar{g}$ =7.

The expected number of moves for one particle to cover a 1D area of length m is

$$\frac{1}{m}\sum_{i=1}^{m}(i+m) = \frac{3m+1}{2}. \tag{3.1}$$

3.2.1.2 1D mapping with n particles

Let $\mathbf{p}$ be the list of positions of n particles uniformly distributed from $[1, m]$. As shown in Fig. 3.3, the number of moves to discover the left and right boundaries is bounded by the maximum and minimum particles $\underline{p} = \min(\mathbf{p})$, $\overline{p} = \max(\mathbf{p})$, requiring moving left $\underline{p}$, followed by a move of $m - (\overline{p} - \underline{p})$ right. When $n = m$, the algorithm requires 2 moves, one left, one right. The minimum time with $n \in [2, m-1]$ occurs with $\underline{p}$ at 1 and $\overline{p}$ at m, requiring 3 moves, 1 left followed by 2 moves to the right.

The maximum $2(m-n+1)$ occurs when the particles are arranged from $m-n$ to m, requiring $m-n+1$ moves to the left, followed by $m-n+1$ moves to the right.

This is drawing without replacement n times from the set $[1, m]$. The minimum is distributed between 1 and $m-n$, the maximum is distributed between n and m.

The expected number of moves to reach both boundaries for n particles in 1D is

$$\frac{1}{\binom{m}{n}}\sum_{\underline{p}=1}^{m-n+1}\sum_{\overline{p}=\underline{p}+n-1}^{m}\binom{\overline{p}-\underline{p}-1}{n-2}\left(2\underline{p}+m-\overline{p}+1\right)$$

$$= \frac{3(1+m)}{1+n}. \tag{3.2}$$

To fully map the area from 1 to m requires that every position from 1 to m be

visited by at least one particle. This time is dominated by the maximum gap $\overline{g}$. The total number of moves is then $2\underline{p} + m - \overline{p} + 1 + \max\left(\overline{g} - \left(\underline{p} + m - \overline{p}\right), 0\right)$.

If all the particles are unit size, there are $m - n$ spaces, and these can be located before, between, or after the n particles in $n+1$ locations giving $\binom{m-n}{n+1}$ possible configurations. The largest gap can be calculated exactly using a recurrence equation [41], but a tight bound when $m > n \log n$ is $\frac{m-n}{n+1} + \Theta\left(\sqrt{\frac{(m-n)\log(n+1)}{n+1}}\right)$ [42].

3.2.1.3 1D mapping with scan and move costs

When controlling particles with an MRI as in [43], the MRI machine iterates between imaging and applying gradient forces to move the particles. This section examines 1D mapping when scanning the workspace and moving the particles a unit distance have associated costs. The objective is to minimize a linear combination of costs for moving and measuring; however, the precise respective coefficients may be subject to change, or even unknown in advance turning this into a *bicriteria problem*, in which both parameters need to be within a bounded ratio of those in an optimal solution. For simpler notation, we write (a, b) for a schedule that involves a unit steps and b scans.

For a more detailed analysis, assume that the left boundary is located D units to the left of the leftmost particle. (This analysis can be applied in both directions.) The theoretically optimal, yet elusive, solution requires scanning the workspace to map particle locations, moving $D+1$ units to the left, then scanning to detect that the leftmost particle has only moved D units and thus has encountered the wall, for a total cost of $(D + 1, 2)$ for the schedule.

We can achieve a schedule with $D + 1$ steps by scanning after each step, for a total cost of $(D+1, D+2)$; while this is optimal with respect to steps, the involved scan cost is large compared to the optimum. At the expense of increasing the number of steps we can reduce the number of scans by successively doubling the number of steps between scans,

i.e., performing the i^{th} scan after 2^i steps, resulting in total cost at most $(2D+2, \log_2 D)$; replacing the base of 2 by an arbitrary constant k, we get $(k \times (D+1), \log_k D)$. This is within a constant of the optimal. On the other hand, moving a sufficiently large number M of steps (known to satisfy $M \geq D$) before performing the second scan yields $(M, 2)$, which is optimal with respect to scan cost, but bad in terms of the cost for motion.

Balancing the competitive factor for both parameters can be achieved as follows: perform the ith scan at position i^i. This yields a simultaneous competitive factor of $O(\log D / \log \log D)$ for *both* parameters.

Theorem 7. *The hyperexponential search sequence i^i yields a best possible simultaneous competitive factor of $O(\log D / \log \log D)$ for* both *parameters of the bicriteria search problem.*

Proof. Let us first consider the number of scans. If the boundary is properly detected in step $j+1$, the particle must have encountered it between steps j and $j+1$, i.e. $j^j \leq D < (j+1)^{j+1}$. Now we can employ the Lambert W function, which is the inverse function of $f(x) = xe^x$; note that

$$\log x = W(\log x) \cdot e^{W(\log x)},$$

so

$$\log \log x = (\log W(\log x) + W(\log x)),$$

and therefore

$$W(\log x) \in \Theta(\log \log x).$$

This implies that $j \leq \log D / W(\log D)$ (the inverse of j^j), hence $\Theta(\log D / \log \log D) +$ 1 scans have been made, while the optimum are 2 scans.

The moved distance is $(j+1)^{j+1}$, while the optimum is $D \geq j^j$. Hence, we get the ratio

$$\frac{(j+1)^{(j+1)}}{j^j} = (j+1)\frac{(j+1)^j}{j^j}, \tag{3.3}$$

where $0 \leq \frac{(j+1)^j}{j^j} \leq \mathrm{e}$ for $j > 0$.

Because $j \leq \log D/W(\log D)$, we obtain

$$\frac{(j+1)^{(j+1)}}{j^j} \leq \mathrm{e}\frac{\log n}{W(\log n)} + \mathrm{e} \tag{3.4}$$

for $j > 0$. This is again within a factor of $\Theta(\log D/\log\log D) + 1$ of the optimum. □

3.2.2 Mapping in 2D

3.2.2.1 2D mapping with 1 particle

The shortest path for mapping with 1 particle is a version of depth-first search that halts when all frontier cells have been explored. As long as the all the free cells are connected, depth-first-search (DFS) is the optimal solution to mapping. Even if the environment is known in advance like in the case of coverage and foraging, the problem is NP-hard as can be shown by a trivial reduction to Hamiltonian paths in grid graphs [44]. One can easily show that a simple DFS guarantees an optimal ratio of 2: the depth-first tree has $m - 1$ edges and each edge is traversed at most twice. Any path that covers m fields needs to traverse least $m - 1$ edges, and hence the depth-first-search is at most twice as long as an optimal coverage path.

For showing that no algorithm can perform better one needs only a simple 1-dimensional graph that goes to the left and to the right. If the algorithm chooses to go arbitrarily to one side, we can make it do a long walk of length m and then return it just for a single field on the other side ($2m + 1$ vs. $2 + m$). If the algorithm decides to switch the direction after some time after arbitrary zig-zags (of increasing cost) of cost z (center to one side to other side) we decide that there is a single field on both sides. The algorithm now needs to go one additional time from one side to the other and back (cost $> z$) while the optimum cost would have been $\leq z + 3$. If the algorithm switches from the second form to the first, the first argument still applies.

Most previous work on grid graph exploration focused on exploration tours, i.e., after exploration one has to go back to the start position. If the environment is known in advance, this equals the traveling salesman problem and a polynomial-time approximation scheme is known [45]. If the environment is unknown, as it is in the mapping case, the best achievable competitive ratio is 2 in general grid graphs (achieved by depth first search) and 7/6 for simple grid graphs (4/3 achieved by smartDFS [46]).

3.2.2.2 2D mapping with n particles

The problem with mapping with n particles is also to identify which move sequence guarantees the shortest path in the worst case.

We describe three algorithms of increasing complexity for 2D mapping. If we implement a random move algorithm as described in Alg. 3, at each step the particles all move in the same randomly selected direction until there are no frontier cells left on the map. $MoveType$ is a vector that holds the four possible move types. The map $\mathbf{M}$ is a matrix the size of the work space. Each cell of $\mathbf{M}$ holds one of five values that denote: *Particle*, *Frontier*, *Unknown*, *Freespace* and *Obstacle*. At each step FRONTIER returns the locations of frontier cells in $\mathbf{M}$ and $\mathbf{r}$ has the list of particle locations. The *move* is implemented to update the map $\mathbf{M}$ and the particle locations $\mathbf{r}$ by calling MOVE&UPDATE. Alg. 3 requires minimal computation and is probabilistically complete, so eventually the swarm maps the free cells [47]. However, this method of mapping is inefficient, resulting in long mapping times, especially with small numbers of particles in large, torturous maps with many turns.

Algorithm 3 RANDOMMOVES($\mathbf{M}, \mathbf{r}$)

```
1: MoveType = {l, r, u, d}
2: while |FRONTIER(M)|> 0 do
3:     move ← RANDOM(MoveType)
4:     {M, r} ←MOVE&UPDATE(move, M, r)
5: end while
```

A better way to map the world is to deliberately move particles toward frontier cells. We could choose one particle as the *elect* particle and perform motion planning using this particle. In Alg. 4, one of the particles is selected as *elect*. As long as the number of frontiers to be visited is at least one, the algorithm proceeds by generating a $mvSq$ from the current position of the elect particle to the nearest frontier cell. The $mvSq$ is generated by a breadth-first-search (BFS) shortest path algorithm which requires the map $\mathbf{M}$, source *elect*, and the cells FRONTIER($\mathbf{M}$). A representative $mvSq$ is $\langle u, r, d, d, r, u, \ldots \rangle$. The list of moves in $mvSq$ are implemented by iteratively calling the MOVE&UPDATE function for the length of $mvSq$.

Alg. 4 will explore the target frontier cell by the end of $mvSq$. However, often with large-population swarms the whole $mvSq$ need not be implemented. Every time MOVE&UPDATE is called, the nearest frontier is updated and $mvSq$ is also updated because as the particles start to move, the target frontier cell might be explored by a non-elect particle.

Alg. 5 exploits this fact by computing a BFS shortest path from all particles to all frontier cells.

Algorithm 4 ELECTPARTICLE($\mathbf{M}, \mathbf{r}$)

```
1: elect ← RANDOM(r)
2: while |FRONTIER(M)|> 0 do
3:     mvSq ← BFS(M, elect,FRONTIER(M))
4:     for iter := 1 to |mvSq| step 1 do
5:         {M, elect} ←MOVE&UPDATE(mvSq, M, elect)
6:     end for
7: end while
```

In each loop of Alg. 5, all the moves in $mvSq$ are implemented to explore the target frontier cell since it is the shortest possible route to a frontier cell. At time 0, there will be at most $4n$ equally valid destinations that can be visited since cells to the side of each particle not neighboring another particle are frontier cells. Each $mvSq$ guarantees classification of one frontier cell into obstacle or free space. When a frontier

cell is explored it is labelled either free or obstacle. There can be a net gain of at most two frontier cells per particle that encounters a free cell or no new frontier cells if the frontier cell contained an obstacle.

The simulation results in Section 3.3 show that both map complexity and distribution affect the number of moves taken to map the work space. Alg. 3 uses no information from the data except for checking completion. Alg. 4 uses the location and distance data from one particle to map the work space. Alg. 5 improves the performance of mapping by using all the data.

Algorithm 5 ClosestFrontier($\mathbf{M}, \mathbf{r}$)

```
1: while |Frontier(M)|> 0 do
2:     mvSq ← BFS(M, r,Frontier(M))
3:     for iter := 1 to |mvSq| step 1 do
4:         {M, r} ←Move&Update(mvSq, M, r)
5:     end for
6: end while
```

While Alg. 5 is usually the more reasonable approach in practice than DFS with a single particle, we are only able to show a trivial weaker bound on the corresponding moves. Alg. 5 is not optimal. It can need $\Omega(n^2)$ moves while an optimal strategy only needs $O(n)$. An example can be seen in Fig. 3.4. The greedy strategy Alg. 5 could go right first and then cover the square with a single particle which takes $\Omega(n^2)$ moves while the optimal strategy, which visits the square in parallel using all n particles, only needs $O(n)$ moves. In some scenarios Alg. 5 can perform worse than DFS with a single particle, e.g., in Fig. 3.5.

Theorem 8. *Alg. 5 needs at most* $0.5 \cdot m \cdot (m+1)$ *moves where* m *is the number of fields.*

Proof. The distance between a particle and the closest frontier cell can be at most the number of all already visited cells. Hence, the distance for visiting the i^{th} field is bounded by i. The overall number of moves is bounded by $\sum_{i=1,\dots,m} i = 0.5 \cdot m \cdot (m+1)$. □

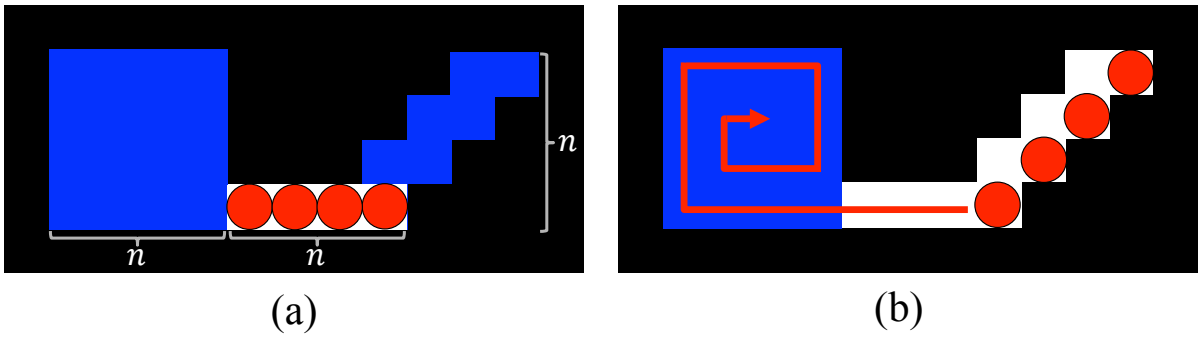


Figure 3.4: For the space as seen in (a) and (b), using an optimal algorithm would require $O(n)$ moves, while the greedy algorithm needs $\Omega(n^2)$

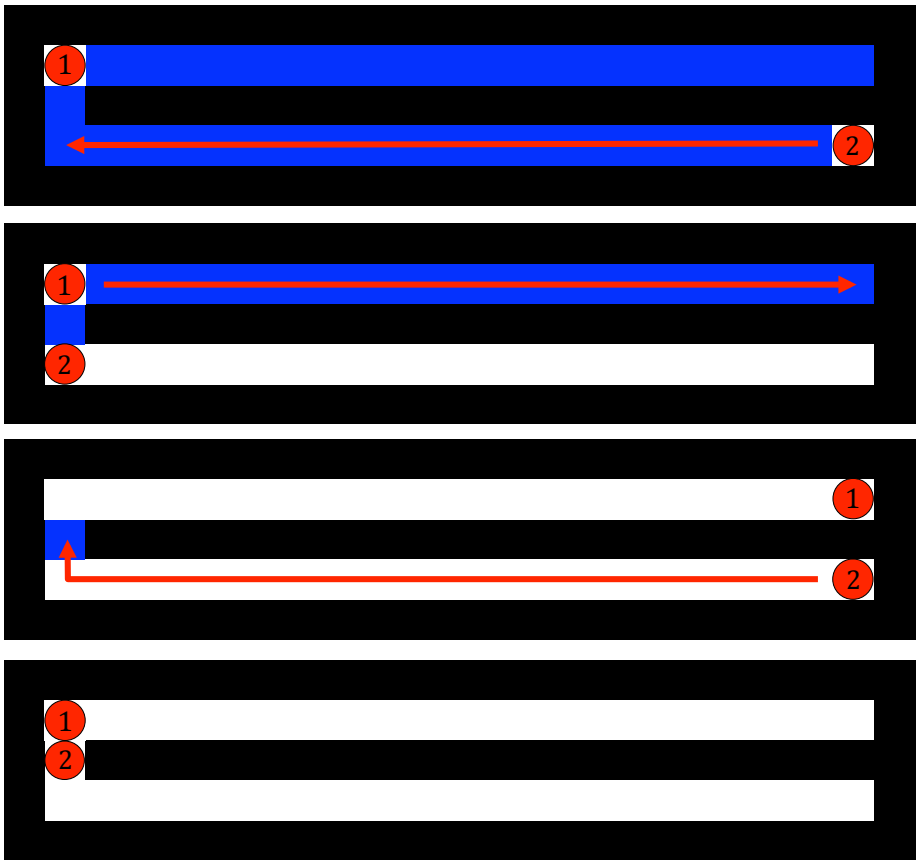


Figure 3.5: In this example, using Alg. 5 is 1.5× worse than DFS using only particle 2.

Theorem 9. *Alg. 5 has a computational complexity in* $O(m^2)$.

Proof. For an environment with m cells, there are at most m iterations. Since the edges in the grid graph are not weighted and each cell only has at most four neighbors, the shortest path from a particle to the frontier cell can be calculated in $O(m)$ time by a simple breadth first search. □

Finally, completion time is also a function of the map geometry. Mapping requires exploring all the free spaces and the boundary of the free spaces. The number of map cells that need to be explored is the $Area + Perimeter - n$. This is minimized by a circular region and maximized by a linear region. For example, a linear region has $m + (2m + 2) - n$ cells to explore, while a square region has $m + (4\sqrt{m}) - n$ cells to explore.

3.2.3 Mapping in 3D

Since we are just going from a four connected scenario in 2D to a six connected scenario in 3D (Fig. 3.6). The two additional directions to explore are the z axis positive (top) and z axis negative (bottom). So, Alg. 5 will apply to 3D case as well.

3.3 Results and Inferences

3.3.1 1D mapping

1D simulations were conducted in Mathematica, with code available at [16]. Fig. 3.7 shows the distributions for the minimum and maximum initial particle locations $\underline{p}$ and $\overline{p}$, the maximum gap $\overline{g}$, and the spread between the minimum and maximum $\overline{p} - \underline{p}$ for 1,000,000 Monte Carlo trials. The expected gap between the first particle and the boundary $\underline{p}$ is 90.94. The expected gap between the last particle and the boundary $\overline{p}$ is 90.98. The expected maximum gap is $\overline{g}$ is 273.9.

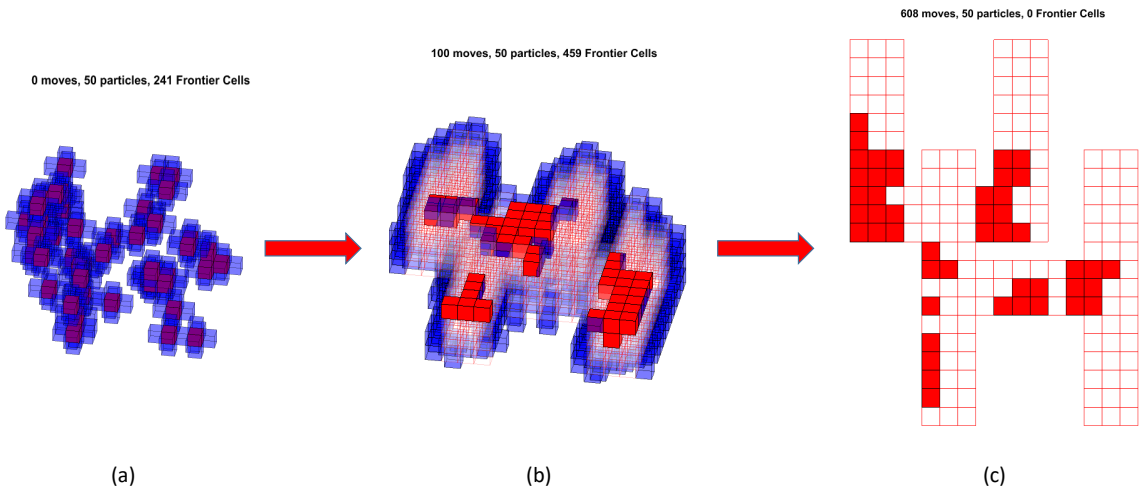


Figure 3.6: Mapping a 3D space of m=720 using Alg. 5 with n=50. In (a) the particles are initialized with six frontiers around them. In (b) the map is gradually built similar to the 2D case. (c) Shows the fully mapped space from top view

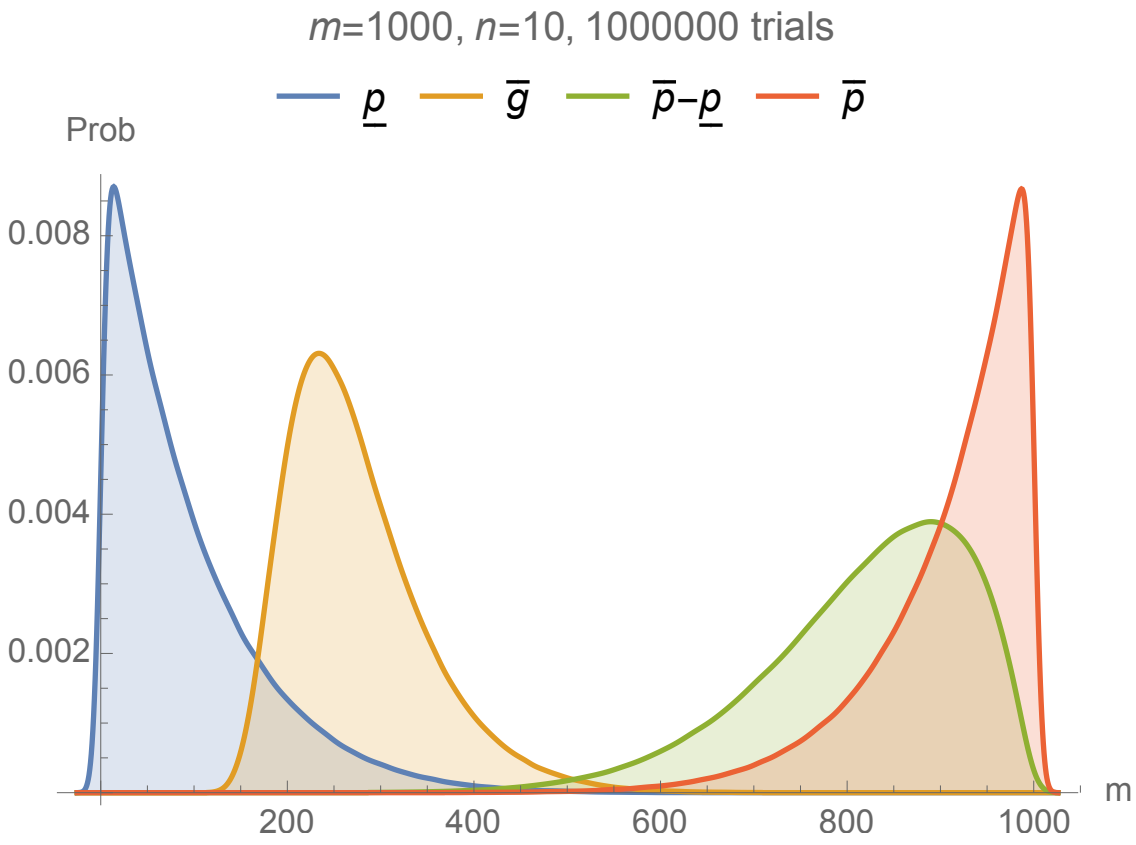


Figure 3.7: With a connected 1D freespace m= 1000 and n=10 particles, the distributions for the gap before the first $\bar{p}$ and after the last $\underline{p}$ gaps are symmetric. The maximum gap $\bar{g} \approx 250$.

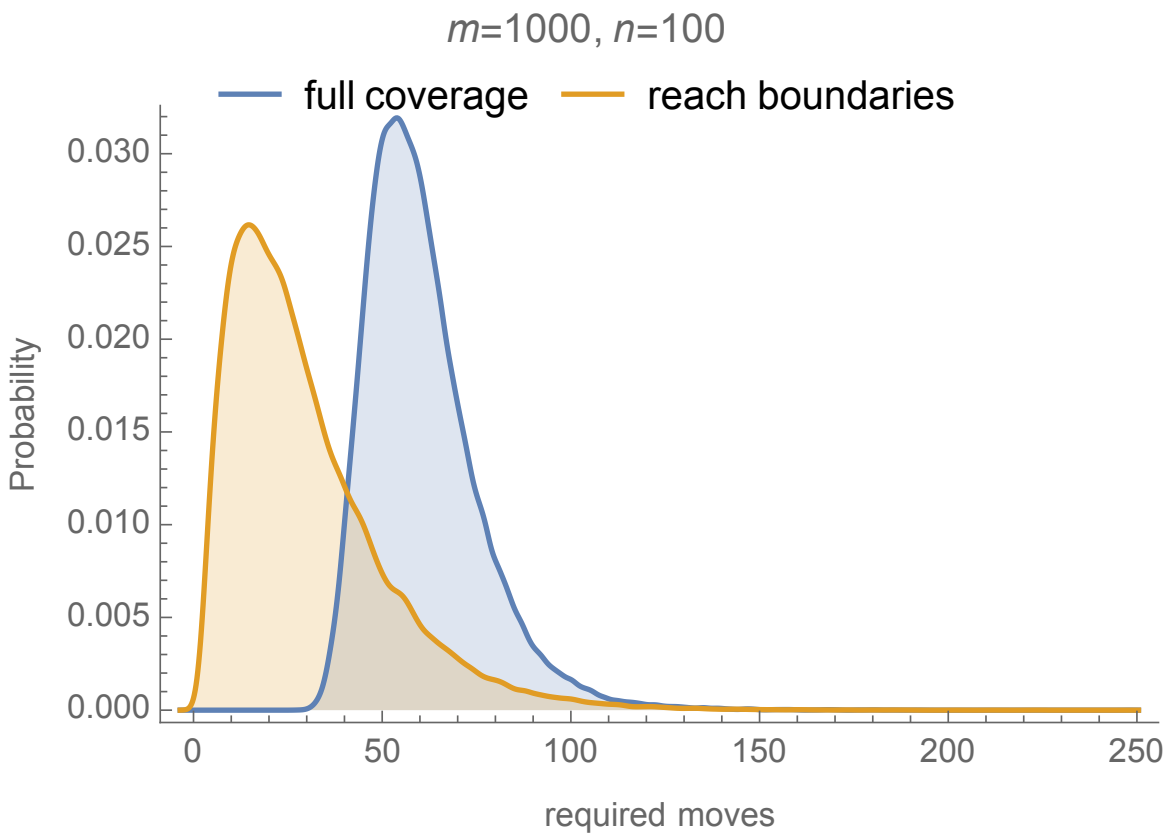


Figure 3.8: Full coverage in 1D with $m = 1000$ and $n = 100$ particles requires 60.7 moves on average, while reaching the boundaries requires only 29.8.

Fig. 3.8 shows that full coverage requires approximately twice the time required to explore the left and right boundaries when $m = 1000$ and $n = 100$.

3.3.2 2D mapping

All simulations used maps with 5000 free cells. Each simulation trial was repeated 100 times. The number of particles ranged from 100 to 5000 by increments of 100. In each run except Fig. 3.13 the particles were placed uniformly randomly throughout the workspace.

The comparison plot Fig. 3.9 between the mapping of four 2D maps *H-tree*, *complex*, *empty rectangle* and *linear* and a 1D map, shows that the H-tree map requires the most

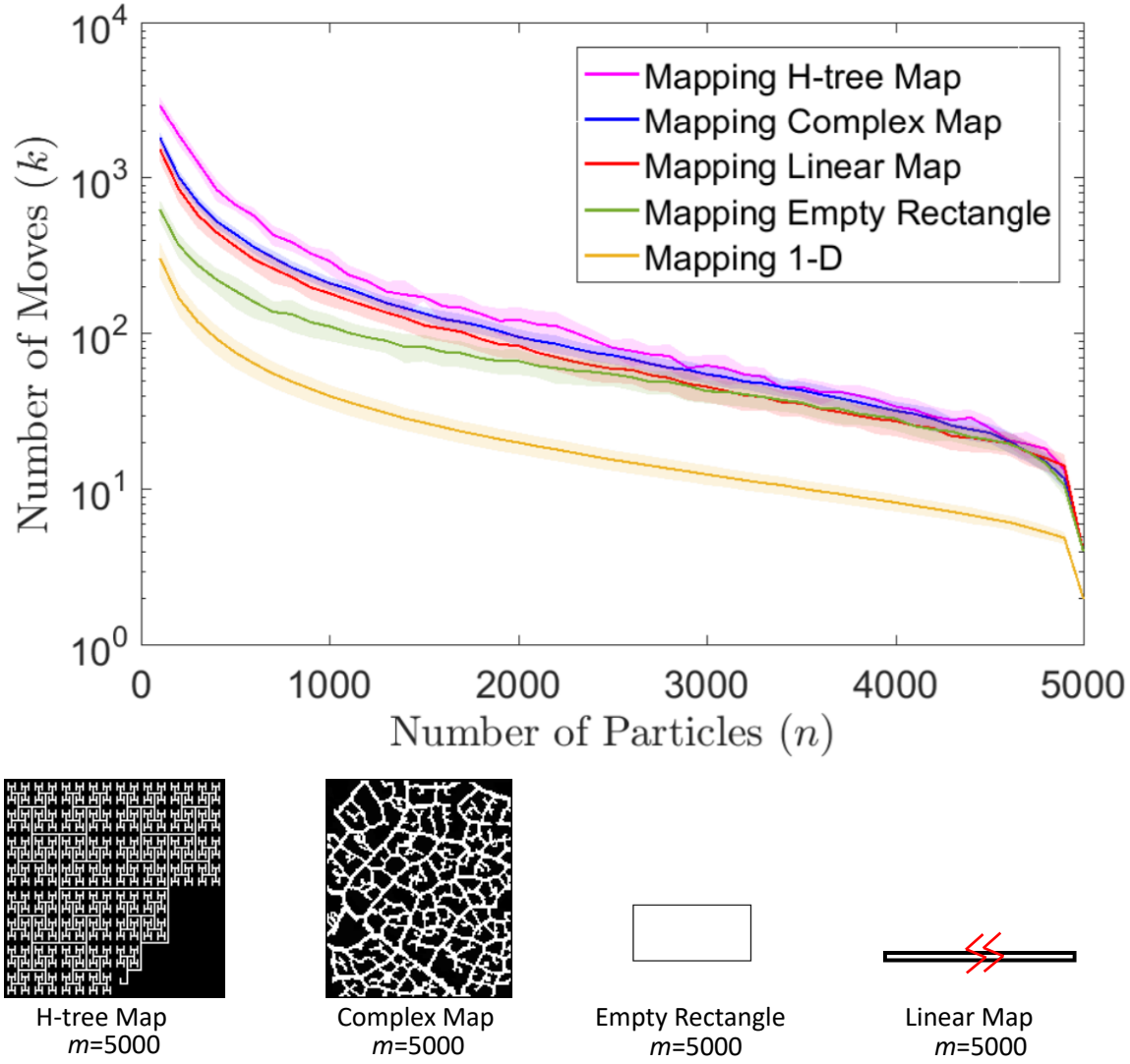


Figure 3.9: Comparison of mapping using the Closest Frontier algorithm on 2D maps of four types and 1D mapping on a linear map. Each map has 5000 free cells.

moves because it has the highest number of turns. In Fig. 3.9 there is an observable difference in moves between the linear and rectangular workspaces. One reason is because the perimeter of the linear map is much larger than the rectangular map. The number of cells to explore is $3m + 2 = 15,002$ for the linear map, but only $m + 2(50 + 100) = 5,300$ for the rectangular map. Only when the number of particles is around 2/3 of the number of free spaces is there an overlap between the moves taken to map the rectangular space and the linear space.

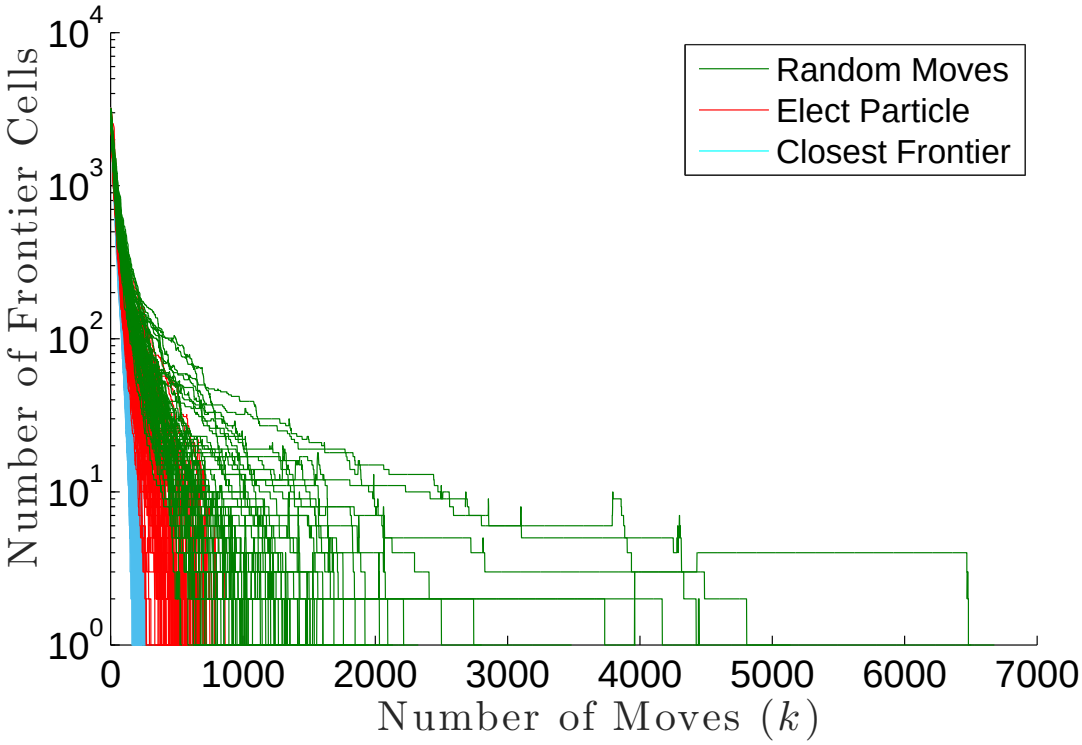


Figure 3.10: Performing mapping on the complex 2D map with $n = 1000$ particles. Random moves requires 1683 moves, Elect particle requires 578 moves and Closest Frontier requires 215 moves on average

The difference between algorithms is highlighted in Fig. 3.10, which shows the number of frontier cells as a function of the number of moves commanded. All tests used $n = 1000$ particles. Elect Particle requires on average twice as many moves as Closest Frontier and Random Moves requires ten times as many moves as Closest Frontier. The deviations for the Closest Frontier are also much less than the other two as seen from Fig. 3.10 and Fig. 3.11.

Fig. 3.11 compares the performance of Algs. 3, 4, and 5 on the complex 2D map. For all algorithms the mean completion time and standard deviation of the completion time decreased with increasing numbers of particles. Random Moves performs worst, with the largest number of required moves and the largest standard deviation of required moves. Random Moves is slightly better than Elect Particle for large numbers of particles, but

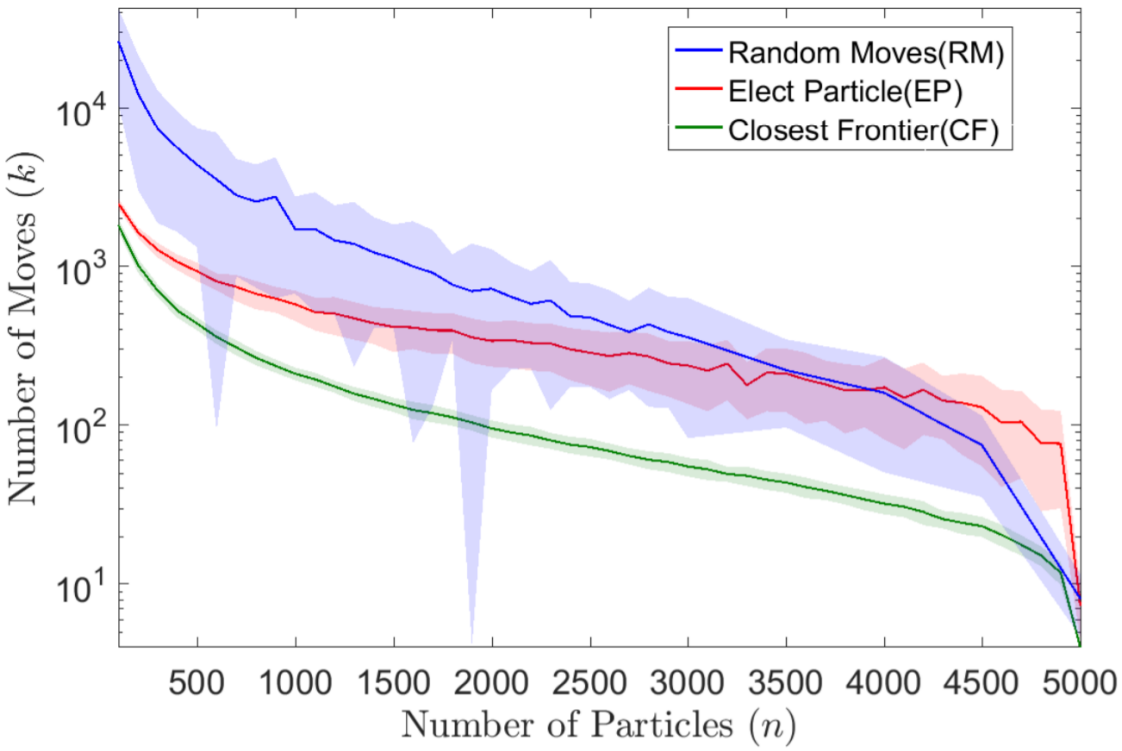


Figure 3.11: Comparison of three algorithms - Random Moves, Elect Particle and Closest Frontier for mapping the 2D Complex Map of 5000 free spaces.

both algorithms are beat by Closest Frontier, which has the minimum number of required moves and the smallest standard deviation. The maximum number of moves required using the Closest Frontier algorithm was for k=100 with an average of ≈1816 moves and standard deviation of 160 moves. This reduces to four moves with 0 standard deviation when n= 5000 (the total number of free spaces).

Fig. 3.12 compares mapping, coverage, and foraging on the complex 2D map. All trials used Alg. 5. Coverage is performed with a known map, but with all free cells initialized to be frontier cells. Similarly, foraging has a known map, but 10% of the empty cells are labelled as frontier cells. Foraging is easier than coverage and coverage is easier than mapping.

The final simulation test, shown in Fig. 3.13, compares the effect of different initial

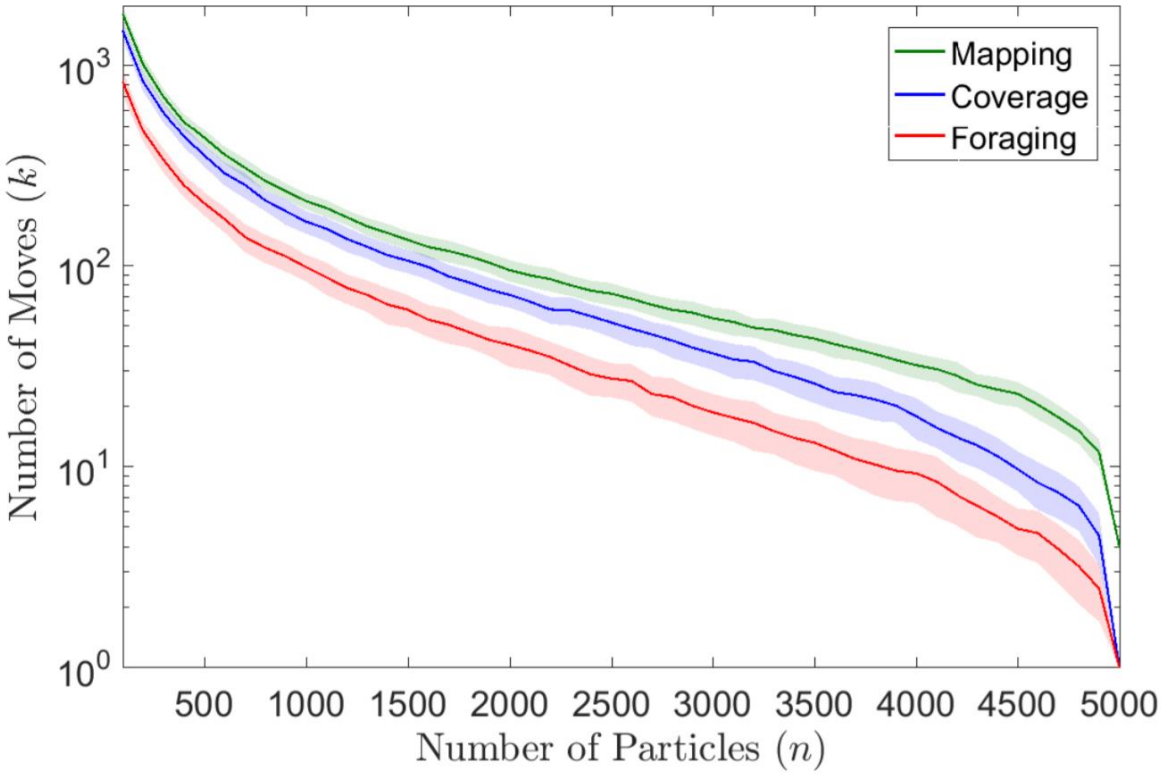


Figure 3.12: Comparison of three related problems: mapping, coverage, and foraging on the complex 2D map.

particle distributions in the complex 2D map. Region fill places all n particles at a minimum Manhattan distance from a randomly selected location on the map. Flood fill places one particle at a randomly selected location in the free space, and places the remaining particles according to a breadth-first expansion inside the free space. Uniform distribution places the particles uniformly randomly. Region fill and flood fill have similar performance, while uniform distribution requires many fewer moves. This is because dispersing particles using only global inputs is difficult, and a uniform distribution starts with the particles dispersed, which allows it to map much faster.

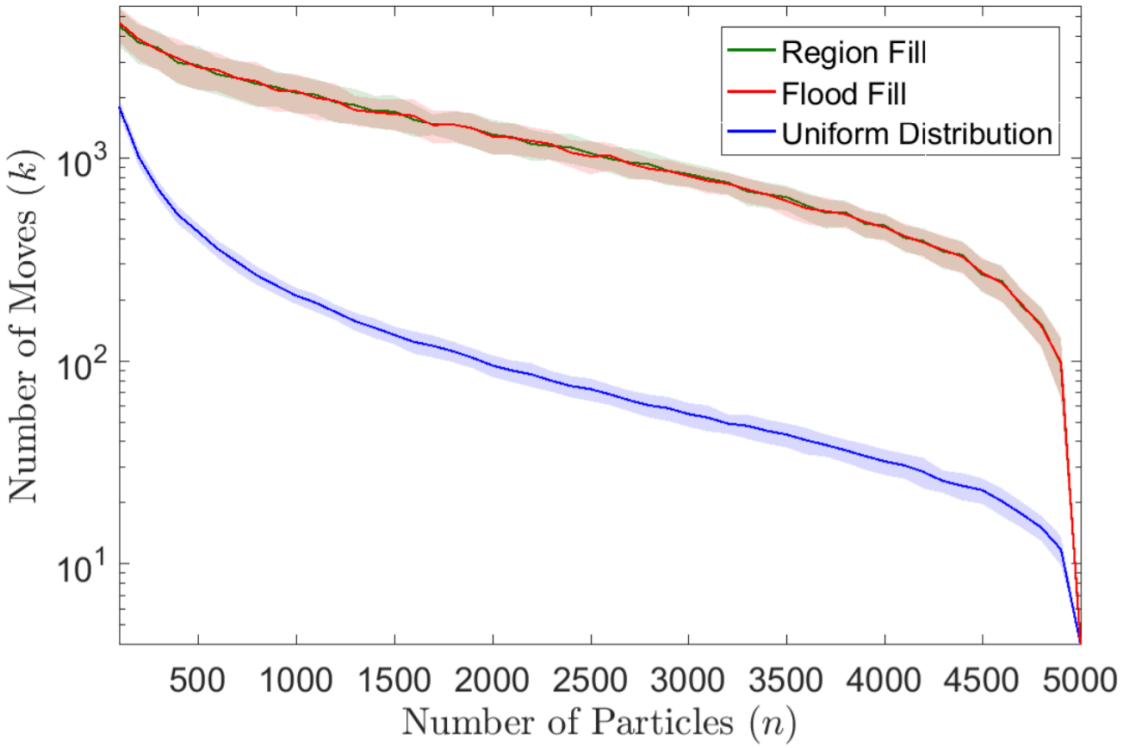


Figure 3.13: Comparison with different distributions: flood-fill, region fill, and uniform distribution for mapping on the complex 2D map. The results for flood-fill and region fill overlap.

Chapter 4

Shape Control

Boundary interactions are a common phenomenon in vivo and many applications, this gives rise to phenomenon such as viscosity and wall friction. In previous work done using obstacle interaction [48] it was demonstrated that with just one obstacle it was possible to shape a swarm of globally controlled particles. But in vivo we have bounded regions of freespace and boundaries that serve as static obstacles. The other consideration to keep in mind is that frictional forces will act on particles that come in contact with the boundaries. So in this work, for workspaces with such constrains, we develop a complete algorithm to perform accurate particle positioning as a solution for shape control of micro particle swarms as seen in Fig. 4.1. We consider ideal friction occurring when the particles are interacting with the boundary.

4.1 Painting a Picture with Global Control

In this section we will discuss the algorithms used to position particles in a bounded region. The algorithm goes through a process of selectively moving particles by placing certain particles on the walls while performing moves. The moves made by particles in the algorithm will look like the particles are creeping along or climbing up the internal wall. This is what we call as drift moves which will be discussed shortly. The algorithm is terminated when it is done executing n iterations where n is number of particles. In the k^{th} loop the k^{th} robot is moved from a *staging zone* to it's defined position in the *build zone* (Fig. 4.2). At the end of the k^{th} loop, robots 1 through k are in their desired final configuration in the build zone, and robots $k+1$ to n are in the staging zone. Although complete. This solution is not optimal. Also, the two crucial criterion for the

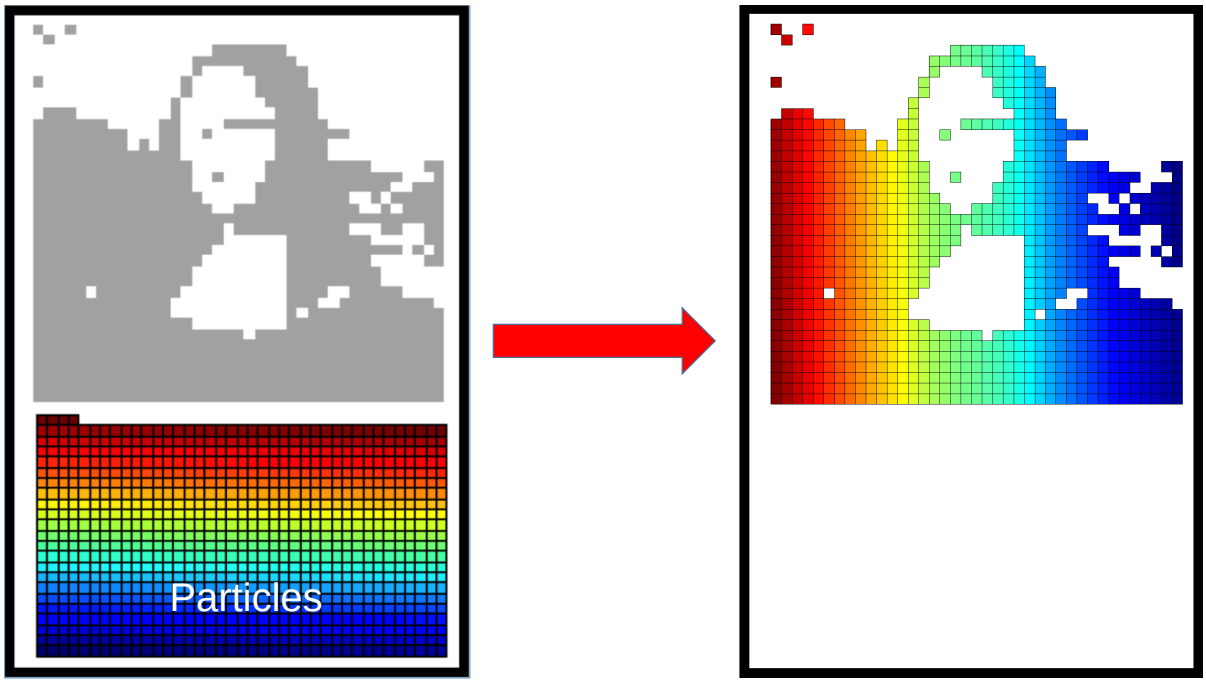


Figure 4.1: n-particle positioning algorithm can be used to generate complex patterns and designs. Here it is used to make a beautiful pixel painting of the Mona Lisa.

algorithm to work is that: 1) The starting configuration(*staging zone*) and the ending configuration(*build zone*) should be disjoint. 2) The staging zone particles must start as ordered set which is filled from the bottom row first. This can be easily accomplished by moving the particles to corners until the desired initial shape shape and distribution is achieved.

The workspace is a rectangular bounded space where the walls have infinite friction. The build zone has dimension (w_b, h_b) contains the position of the final configuration of n robot. The axis-aligned staging zone of dimension (w_s, h_s) containing the starting configuration of n robots. d is the diameter of the particles. Furthermore, there must be at least ϵ space above the build zone, ϵ below the staging zone, and $\epsilon + d$ to the left of the build and staging zone.

The n robots position control algorithm relies on a DRIFTMOVE$(\alpha, \beta, \epsilon, \theta)$ control input, described in Alg. 7 and shown in Fig.4.3. For $\theta = 0°$, a drift move consists

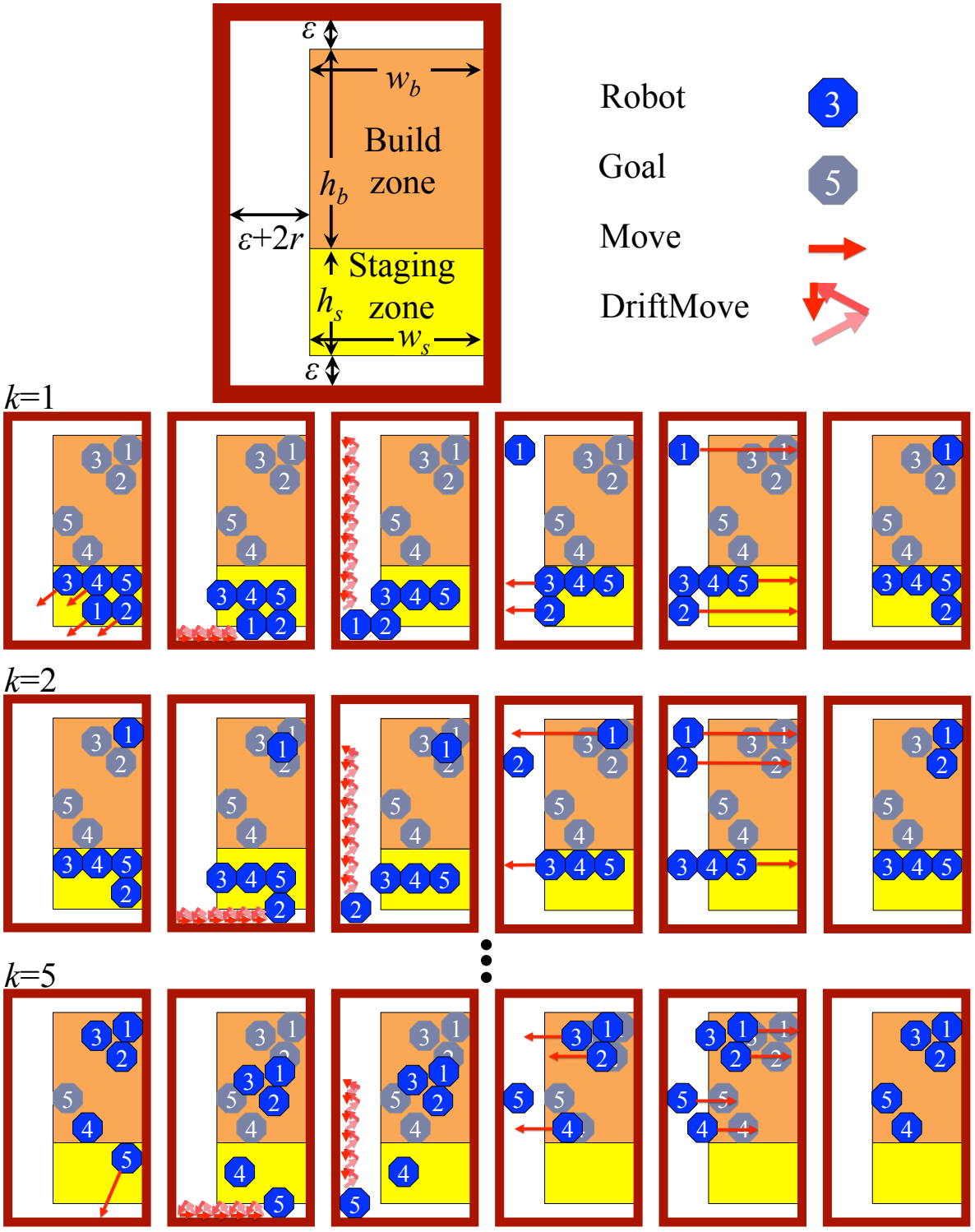


Figure 4.2: Illustration of Alg. 6, n robot position control using wall friction.

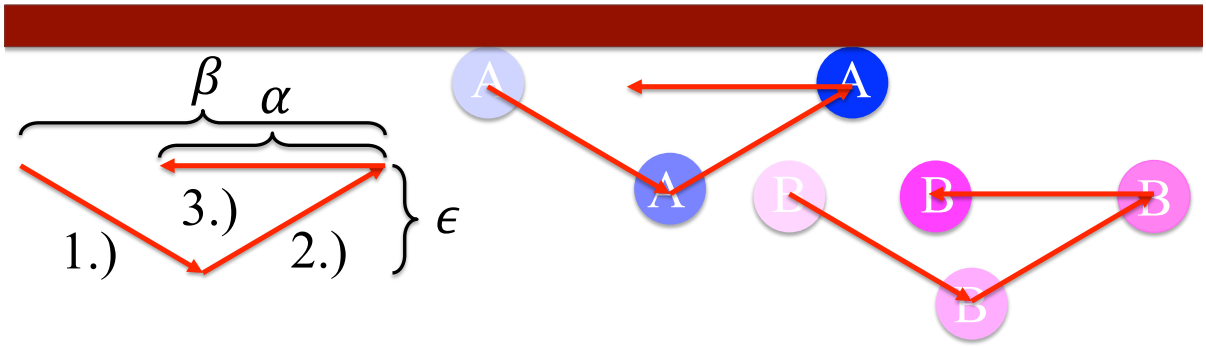


Figure 4.3: A DRIFTMOVE$(\alpha, \beta, \epsilon, 0°)$ repeats a triangular movement sequence $\{(\beta/2, -\epsilon), (\beta/2, \epsilon), (-\alpha, 0)\}$. At the sequence end, robot A has moved β units right, and robot B has moved $\beta - \alpha$ units right.

of repeating a triangular movement sequence $\{(\beta/2, -\epsilon), (\beta/2, \epsilon), (-\alpha, 0)\}$. In a given drift move any particle touching a top wall moves right β units, while every particle not touching the top moves right $\beta - \alpha$.

The lower left corner represents $(0, 0)$ and in this frame, p_k the x, y position of the k^{th} robot, and f_k the final x, y position of the k^{th} robot. Label the robots in the staging zone from left-to-right and bottom-to-top, and the f_k configurations top-to-bottom and right-to-left as shown in Fig. 4.7.

Algorithm 6 PositionControl$n$RobotsUsingWallFriction(k)

Move($-\epsilon, d/2 - p_{ky}$)
while $p_{kx} > d/2$ **do**
    DRIFTMOVE$(\epsilon, \min(p_{kx} - d/2, \epsilon), \epsilon, 180°)$
end while
$m \leftarrow \text{ceil}(\frac{f_{ky} - d/2}{\epsilon})$
$\beta \leftarrow \frac{f_{ky} - d/2}{m}$
$\alpha \leftarrow \beta - \frac{d/2 - p_{ky} - \epsilon}{m}$
for m iterations **do**
    DRIFTMOVE$(\alpha, \beta, \epsilon, 90°)$
end for
Move $(d/2 + \epsilon - f_{kx}, 0)$
Move $(f_{kx} - d/2, 0)$

Alg. 6 proceeds as follows: First, the robots are moved left away from the right wall, and down so all robots in k^{th} row touch the bottom wall. Second, a set of DriftMoves are executed that move all robots in k^{th} row left until k touches the left wall, with no net

Algorithm 7 DRIFTMOVE$(\alpha, \beta, \epsilon, \theta)$

particles touching the wall move β units, while particles not touching the wall move $\beta - \alpha$ units.

1: $R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$
2: MOVE$(R \cdot [\beta/2, -\epsilon]^\top)$
3: MOVE$(R \cdot [\beta/2, \epsilon]^\top)$
4: MOVE$(R \cdot [-\alpha, 0]^\top)$

movement of the other robots. Third, a set of DriftMoves are executed that move only robot k to its target height and return the other robots to their initial heights. Fourth, all robots except robot k are pushed left until robot k is in the correct relative x position compared to robots 1 to $k-1$. Finally, all robots are moved right until robot k is in the desired target position. Running time is $O(n(w+h))$.

4.2 Experiment, Results and Inferences

4.2.1 Position Control Algorithm for n Robots

Alg. 6 was simulated in MATLAB using square block robots with unity width. Code is available at [49]. Simulation results are shown in Fig. 4.4 for arrangements with an increasing number of robots, n= [8, 46, 130, 390, 862]. The distance moved grows quadratically with the number of robots n. A best-fit line $210n^2 + 1200n - 10,000$ is overlaid by the data.

In Fig. 4.4, the amount of clearance is $\epsilon = 1$. Control performance is sensitive to the desired clearance. As ϵ increases, the total distance decreases asymptotically, as shown in Fig. 4.5, because the robots have more room to maneuver and fewer DriftMoves are required.

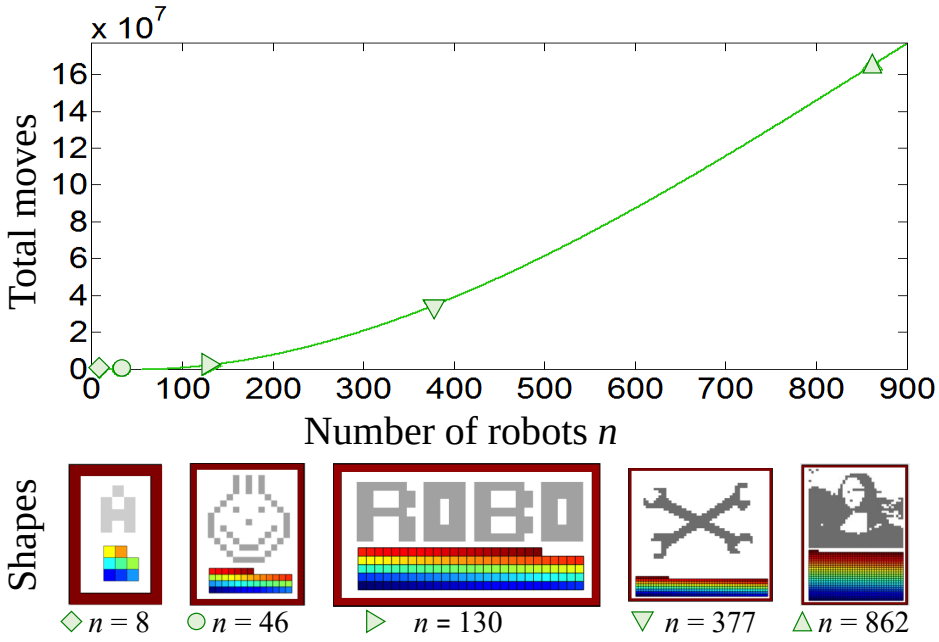


Figure 4.4: The required number of moves under Alg.6 using wall-friction to rearrange n square-shaped robots grows quadratically with n. See hardware implementation and simulation at [49].

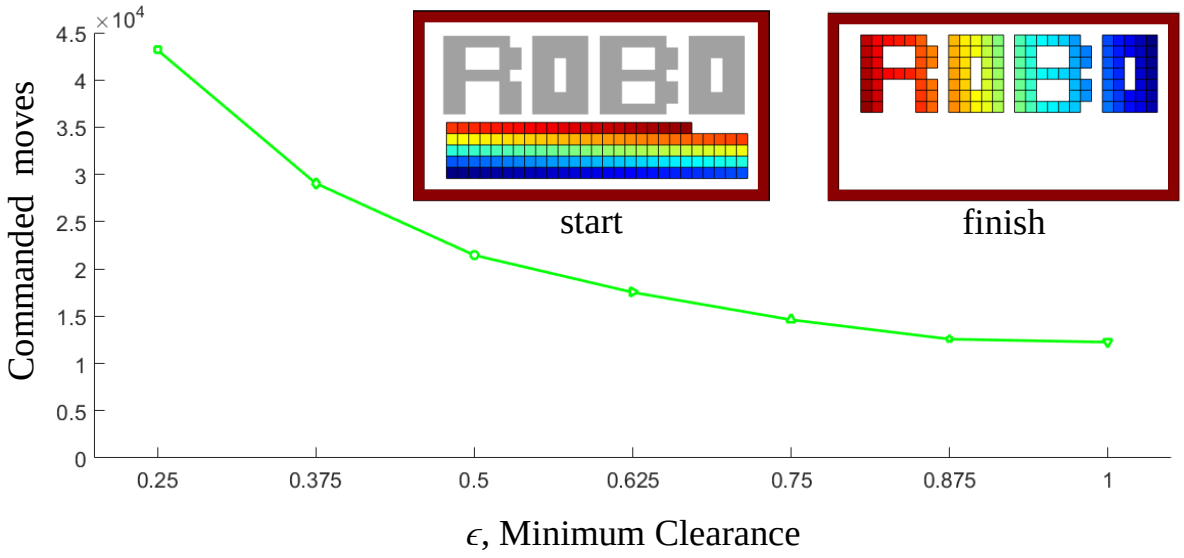


Figure 4.5: Control performance is sensitive to the desired clearance ϵ. As ϵ increases, the total distance decreases asymptotically.

4.2.2 Hardware Experiment: Position Control of n Robots

The hardware platform depicted in Fig. 4.6 is an assembled practical setup that assumes that $\epsilon = 1$ cm. The workspace is a 7×7 grid space of dimension 14×14 cm. Each slot in Fig. 4.6(a) is 1.5 cm wide and each spacer tooth is 5 mm in width. All particles are 3D-printed plastic (Fig. 4.6(c)) whose top is a 2 cm diameter cylinder with a narrower base of 1.4 cm diameter that encapsulates(Fig. 4.6(d)) a steel bearing ball of 1.2 cm diameter (Fig. 4.6(f)) . Wall friction is emulated by a toothed wall design to keep particles from moving out of place while implementing the drift move. The workspace boundary is mounted on top of a white sheet of acrylic. Underneath the acrylic, a grid of 9 mm diameter magnets are glued to a thin wooden board with 2 cm spacing between the centers of any two adjacent magnets as seen in Fig. 4.6(b). This generates the global control input. A video attachment [50] shows the algorithm at work. This discretized setup requires several modifications to Alg. 6. In this demonstration, all moves are 2 cm in length. All drift moves are an counterclockwise *square* move of size 2 cm$\times$2 cm. Once the k^{th} roller gets to its designated location in each loop, a correction step is implemented. This correction step increases by two the total number of moves required per particle. Fig. 4.2 shows there are only 6 stages per particle involved in Alg. 6. The fixed step algorithm requires 8 stages per particle as shown in Fig. 4.7.

A significant difference between Alg. 6 and the fixed move implementation of it is that Alg. 6 enables placing particles at arbitrary, non-overlapping locations, while the fixed move implementation requires goal locations at the center of grid cells.

The goal configuration highlighted in the top right corner represents a 'U' made of seven sliders. The dark red configuration is the current position of the sliders. Due to the discretized movements allowed by the boundary, drift moves follow a 1 cm square. Free robots return to their start positions but robots on the boundary to move laterally, generating a net sliding motion.

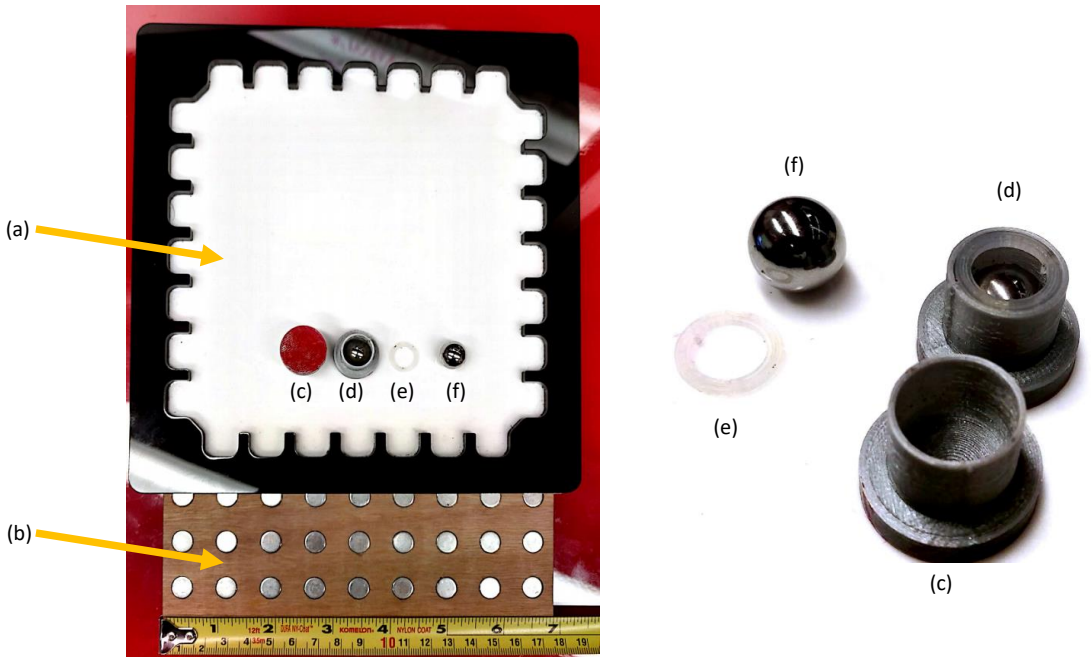


Figure 4.6: The setup consists of (a)magnetic board, (b) work space and (c) particles. (d), (e), (f) are the parts of the particles. On the right the close up of the parts of the particles are shown clearly.

Fig. 4.7 follows the motion of the sliders through iterations k=1, 2, and 7. All robots receive the same control inputs, but boundary interactions break the control symmetry. Robots reach their goal positions in a first-in, first-out arrangement beginning with the bottom-left robot from the staging zone occupying the top-right position of the build zone.

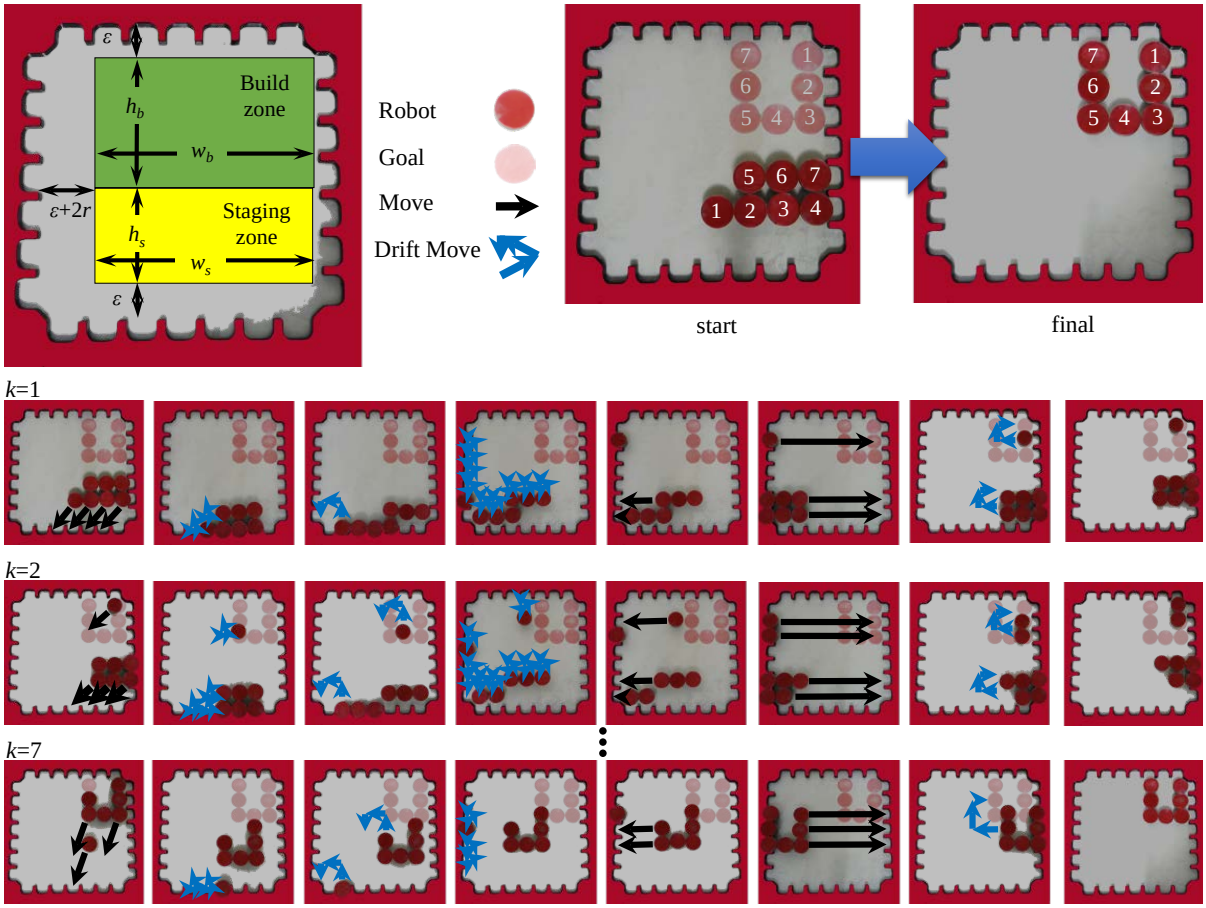


Figure 4.7: Illustration of Alg. 6, discretized n robot position control using wall friction.

Chapter 5

Conclusion

In this work, we have discussed about manipulation of large swarms of particles using global inputs and introduced benchmarking algorithms for tasks such as aggregation, mapping and shape control. This work establishes the necessary theoretical groundwork and baseline for developing algorithms for controlling large microbot swarms. We presented optimal and greedy algorithms to aggregate *small particles* using global control with guarantees that these algorithms will always aggregate particles for any bounded world which can be represented as a connected polyomino. Algorithm *connect to first* has both low computational time and, in simulations, requires fewer moves than five other algorithms. It requires 50,607 moves to converge all the particles in the complete leaf world shown in Fig. 1.1 (see video [51]). We also introduced challenges inherent with large particle aggregation, which poses new problems and complexities. The insight and proofs produced here provided a basis for further research on overcoming the problems involved with *large* particle manipulation.

Based on the learning from the succesful aggregation of particles we developed techniques for performing mapping with such particles.

We provided a spectrum of theoretical and practical new insights with potential relevance for fast MRI scans with magnetically controlled contrast media. In particular, we developed a fundamentally new approach for searching for an object at an unknown distance D, where the search is subject to two different and independent cost parameters for *moving* and for *measuring*. We showed that regardless of the relative cost of these two operations, there is a simple $O(\log D/\log\log D)$-competitive strategy. Second, presented benchmark algorithms for 2D and 3D mapping, foraging, and coverage problems. These

results form a baseline for future work, which should focus on improving performance. Extensions to 3D are especially relevant to the motivating problem of MR-scanning in living tissue.

Finally we provided algorithms for precise position control with wall friction interaction. Here we assumed friction was sufficient to completely stop particles in contact with the boundary. The algorithms require retooling to handle small friction coefficients. The algorithms assumed a rectangular workspace. This is a reasonable assumption for artificial environments, but in vivo environments are curved.

Future efforts should be directed toward improving the technology and tailoring it to specific robot applications. Our work assumed the workspace was bounded. That assumption is violated in biological vascular systems, which connect to larger vasculature. One avenue for future research is to add constraints to serve as virtual walls and actively prevent particles from escaping through a set of exits. Additional complexities such as medium viscosity and wall friction must be studied before the algorithms are applied in vivo/in vitro. Future work should focus not only on aggregating, but also on avoiding accumulation in sensitive regions.

References

[1] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A Low Cost Scalable Robot System for Collective Behaviors," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3293–3298.

[2] Y. Ou, D. H. Kim, P. Kim, M. J. Kim, and A. A. Julius, "Motion Control of Magnetized Tetrahymena Pyriformis Cells by a Magnetic Field with Model Predictive Control," *The International Journal of Robotics Research*, vol. 32, no. 1, pp. 129–140, 2013.

[3] P.-T. Chiang, J. Mielke, J. Godoy, J. M. Guerrero, L. B. Alemany, C. J. Villagomez, A. Saywell, L. Grill, and J. M. Tour, "Toward a Light-Driven Motorized Nanocar: Synthesis and Initial Imaging of Single Molecules," *ACS nano*, vol. 6, no. 1, pp. 592–597, 2011.

[4] P. Pouponneau, J.-C. Leroux, and S. Martel, "Magnetic Nanoparticles Encapsulated into Biodegradable Microparticles Steered with an Upgraded Magnetic Resonance Imaging System for Tumor Chemoembolization," *Biomaterials*, vol. 30, no. 31, pp. 6327–6332, 2009.

[5] J. Litvinov, A. Nasrullah, T. Sherlock, Y.-J. Wang, P. Ruchhoeft, and R. C. Willson, "High-Throughput Top-Down Fabrication of Uniform Magnetic Particles," *PloS one*, vol. 7, no. 5, p. e37440, 2012.

[6] J.-B. Mathieu and S. Martel, "Magnetic Microparticle Steering Within the Constraints of an MRI System: Proof of Concept of a Novel Targeting Approach," *Biomedical microdevices*, vol. 9, no. 6, pp. 801–808, 2007.

[7] L. Mellal, D. Folio, K. Belharet, and A. Ferreira, "Magnetic Microbot Design Framework for Antiangiogenic Tumor Therapy," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on.* IEEE, 2015, pp. 1397–1402.

[8] H. B. Na, I. C. Song, and T. Hyeon, "Inorganic Nanoparticles for MRI Contrast Agents," *Advanced materials*, vol. 21, no. 21, pp. 2133–2148, 2009.

[9] P. Caravan, J. J. Ellison, T. J. McMurry, and R. B. Lauffer, "Gadolinium (iii) Chelates as MRI Contrast Agents: Structure, Dynamics, and Applications," *Chemical reviews*, vol. 99, no. 9, pp. 2293–2352, 1999.

[10] P. Vartholomeos, M. Akhavan-Sharif, and P. E. Dupont, "Motion Planning for Multiple Millimeter-Scale Magnetic Capsules in a Fluid Environment," in *IEEE Int. Conf. Rob. Aut.*, May 2012, pp. 1927–1932.

[11] S. Chowdhury, W. Jing, and D. J. Cappelleri, "Controlling Multiple Microrobots: Recent Progress and Future Challenges," *Journal of Micro-Bio Robotics*, vol. 10, no. 1-4, pp. 1–11, 2015. [Online]. Available: http://dx.doi.org/10.1007/s12213-015-0083-6

[12] A. Becker, G. Habibi, J. Werfel, M. Rubenstein, and J. McLurkin, "Massive Uniform Manipulation: Controlling Large Populations of Simple Robots with a Common Input Signal," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, Nov. 2013, pp. 520–527.

[13] A. Becker, C. Onyuksel, T. Bretl, and J. McLurkin, "Controlling Many Differential-Drive Robots with Uniform Control Inputs," *Int. J. Robot. Res.*, vol. 33, no. 13, pp. 1626–1644, 2014.

[14] S. Shahrokhi and A. T. Becker, "Stochastic Swarm Control with Global Inputs,," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015.

[15] A. V. Mahadev, D. Krupke, J.-M. Reinhardt, S. P. Fekete, and A. T. Becker, "Collecting a Swarm in a Grid Environment using Shared, Global Inputs," in *Automation Science and Engineering (CASE), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1231–1236.

[16] A. Mahadev and A. T. Becker, "Mapping with Particles under Uniform Inputs," Mar. 2017. [Online]. Available: https://github.com/RoboticSwarmControl/IROS2017mappingParticles

[17] S. Shahrokhi, A. Mahadev, and A. T. Becker, "Algorithms For Shaping a Particle Swarm With a Shared Control Input Using Boundary Interaction," Mar. 2017. [Online]. Available: https://github.com/shivashahrokhi/SwarmPublications/tree/master/IROS17

[18] A. Becker, E. Demaine, S. Fekete, and J. McLurkin, "Particle Computation: Designing Worlds to Control Robot Swarms with only Global Signals," in *IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong: IEEE, May 2014, pp. 6751–6756.

[19] A. Becker, E. D. Demaine, S. P. Fekete, G. Habibi, and J. McLurkin, "Reconfiguring Massive Particle Swarms with Limited, Global Control," in *Algorithms for Sensor Systems*, ser. Lecture Notes in Computer Science, P. Flocchini, J. Gao, E. Kranakis, and F. Meyer auf der Heide, Eds. Springer Berlin Heidelberg, 2014, vol. 8243, pp. 51–66. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-45346-5_5

[20] J. O'Kane, "Almost-Sensorless Localization," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2005.

[21] J. O'Kane and S. M. LaValle, "Almost-Sensorless Localization," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 3764–3769.

[22] J. S. Lewis and J. M. O'Kane, "Planning for Provably Reliable Navigation Using an Unreliable, Nearly Sensorless Robot," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1342–1357, 2013. [Online]. Available: http://ijr.sagepub.com/content/32/11/1342.abstract

[23] S. Akella and M. T. Mason, "Orienting Toleranced Polygonal Parts," *The International Journal of Robotics Research*, vol. 19, no. 12, pp. 1147–1170, Dec. 2000.

[24] S. Alpern and S. Gal, *The Theory of Search Games and Rendezvous*, ser. International Series in Operations Research and Managment Science. Boston, Dordrecht, London: Kluwer Academic Publishers, 2003. [Online]. Available: http://opac.inria.fr/record=b1117985

[25] E. J. Anderson and S. P. Fekete, "Two Dimensional Rendezvous Search," *Operations Research*, vol. 49, no. 1, pp. 107–118, 2001.

[26] M. Meghjani and G. Dudek, "Multi-Robot Exploration and Rendezvous on Graphs," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 5270–5276.

[27] P. Zebrowski, Y. Litus, and R. T. Vaughan, "Energy Efficient Robot Rendezvous," in *Computer and Robot Vision, 2007. CRV'07. Fourth Canadian Conference on*. IEEE, 2007, pp. 139–148.

[28] G. Aloupis, J. Cardinal, S. Collette, F. Hurtado, S. Langerman, and J. ORourke, "Draining a Polygon—or—Rolling a Ball Out of a Polygon," *Computational geometry*, vol. 47, no. 2, pp. 316–328, 2014.

[29] A. Becker, E. Demaine, S. Fekete, G. Habibi, and J. McLurkin, "Reconfiguring Massive Particle Swarms with Limited, Global Control," in *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS)*, Sophia Antipolis, France, Sep. 2013, pp. 51–66.

[30] H. Choset, "Coverage for Robotics–a Survey of Recent Results," *Annals of mathematics and artificial intelligence*, vol. 31, no. 1, pp. 113–126, 2001.

[31] X. Zheng, S. Jain, S. Koenig, and D. Kempe, "Multi-Robot Forest Coverage," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on.* IEEE, 2005, pp. 3852–3857.

[32] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein, "Distributed Covering by Ant-Robots using Evaporating Traces," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 918–933, 1999.

[33] D. Latimer, S. Srinivasa, V. Lee-Shue, S. Sonne, H. Choset, and A. Hurst, "Towards Sensor Based Coverage with Robot Teams," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 1. IEEE, 2002, pp. 961–967.

[34] B. Yamauchi, "Frontier-Based Exploration using Multiple Robots," in *Proceedings of the second international conference on Autonomous agents.* ACM, 1998, pp. 47–53.

[35] S. P. Feketea, R. Kleina, and A. Nüchter, "Searching With an Autonomous Robot. (Video + Abstract)," in *Proceedings of the 20th ACM Symposium on Computational Geometry*, 2004, pp. 449–450.

[36] S. P. Fekete, R. Klein, and A. Nüchter, "Online Searching with an Autonomous Robot," *Computational Geometry: Theory & Applications*, vol. 34, pp. 102–115, 2006.

[37] S. P. Fekete, J. S. B. Mitchell, and C. Schmidt, "Minimum Covering with Travel Cost," *Journal of Combinatorial Optimization*, vol. 20, pp. 32–51, 2010.

[38] S. K. Lee, S. P. Fekete, and J. McLurkin, "Structured Triangulation in Multi-Robot Systems: Coverage, Patrolling, Voronoi Partitions, and Geodesic Centers," *International Journal of Robotics Research*, vol. 9, no. 35, pp. 1234–1260, 2016.

[39] S. Martel, "Magnetotactic Bacteria for the Manipulation and Transport of Micro- and Nanometer-Sized Objects," *Micro-and Nanomanipulation Tools*, 2015.

[40] X. Yan, Q. Zhou, J. Yu, T. Xu, Y. Deng, T. Tang, Q. Feng, L. Bian, Y. Zhang, A. Ferreira, and L. Zhang, "Magnetite Nanostructured Porous Hollow Helical Microswimmers for Targeted Delivery," *Advanced Functional Materials*, vol. 25, no. 33, pp. 5333–5342, 2015.

[41] P. Reviriego, L. Holst, and J. A. Maestro, "On the Expected Longest Length Probe Sequence for Hashing with Separate Chaining," *Journal of Discrete Algorithms*, vol. 9, no. 3, pp. 307–312, 2011.

[42] M. Raab and A. Steger, *"Balls into Bins" — A Simple and Tight Analysis.* Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 159–170. [Online]. Available: http://dx.doi.org/10.1007/3-540-49543-6_13

[43] A. Chanu, O. Felfoul, G. Beaudoin, and S. Martel, "Adapting the Clinical MRI Software Environment for Real-Time Navigation of an Endovascular Untethered Ferromagnetic Bead for Future Endovascular Interventions," *Magnetic Resonance in medicine*, vol. 59, no. 6, pp. 1287–1297, Jun. 2008.

[44] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter, "Hamilton Paths in Grid Graphs," *SIAM Journal on Computing*, vol. 11, no. 4, pp. 676–686, 1982.

[45] P. N. Klein, "A Linear-Time Approximation Scheme for TSP in Undirected Planar Graphs with Edge-Weights," *SIAM Journal on Computing*, vol. 37, no. 6, pp. 1926–1952, 2008.

[46] C. Icking, T. Kamphans, R. Klein, and E. Langetepe, "Exploring Simple Grid Polygons," in *International Computing and Combinatorics Conference*. Springer, 2005, pp. 524–533.

[47] J. D. Kahn, N. Linial, N. Nisan, and M. E. Saks, "On The Cover Time of Random Walks on Graphs," *Journal of Theoretical Probability*, vol. 2, no. 1, pp. 121–128, 1989.

[48] A. T. Becker and G. Habibi, ""Massive Uniform Manipulation: Control Large Populations of Simple Robots with a Common Input Signal." http://www.mathworks.com/matlabcentral/fileexchange/42889," MATLAB Central File Exchange, Jul. 2013. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/42889

[49] A. V. Mahadev and A. T. Becker, "Arranging a Robot Swarm with Global Inputs and Wall Friction [discrete]. matlab central file exchange," Feb. 2017. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/54526

[50] S. Shahrokhi, A. Mahadev, and A. T. Becker, "Rearranging Particles using Boundaries," Youtube, Mar 2017. [Online]. Available: https://youtu.be/KcAS2e6nd5s

[51] A. V. Mahadev, D. Krupke, J.-M. Reinhardt, S. P. Fekete, and A. T. Becker, "Collecting a Swarm in a Grid Environment Using Shared, Global Inputs," Youtube, Mar 2016. [Online]. Available: https://www.youtube.com/watch?v=rY7Br4l4SHY