Mapping, Foraging, and Coverage with a Particle Swarm Controlled by Uniform Inputs

Arun Mahadev, Dominik Krupke, Sándor Fekete, and Aaron T. Becker

***Abstract*— Consider a swarm of particles in a 1D or 2D grid that can be tracked and controlled by an external agent, but control inputs are applied uniformly so that each particle experiences the same applied forces. This paper presents algorithms for (1) mapping: building a representation of the free and obstacle regions of the workspace; (2) foraging: ensuring at least one particle reaches each of a set of desired locations; and (3) coverage: ensuring every free region on the map is visited by at least one particle.**

These algorithms extend to any number of particles, and show that additional particles reduce the time required for each task. In the limit, as the particle count increases, the time reduces to four moves for mapping and zero moves for foraging and coverage.

These methods may have particular relevance for fast MRI scans with magnetically controlled contrast media.

I. Introduction

In MR imaging, some tissues have poor *contrast*, which means that the boundaries between tissue types cannot be determined. To overcome this, particulate solutions of contrast agent are used to illuminate regions of interest [1]. 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 [2]. This paper explores using steerable magnetic micro particles to map a region. These particles can be steered by the global magnetic gradient of an MRI and visualised by the MRI [3], even when the tissues they move through have poor contrast.

As a current example, micro- and nano-particles can be manufactured in large numbers, see [4]–[10].

The particles considered in this paper move under the influence of a global command. All particles move in the same direction when commanded. In our previous work [11] 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. We assume particle position can be measured, but the workspace cannot be imaged. Particles respond identically to the global input unless they hit an obstacle or another particle. This paper focuses on discrete 2D workspace. Fig. 1 represents the complete mapping of a workspace with large number of particles. 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 cell in this direction as either obstacle or free. If the particle was able to move, that frontier cell is labelled 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 labelled as obstacle. The goal is to explore all frontier cells, thereby discovering all connected free cells and the obstacles that surround them.

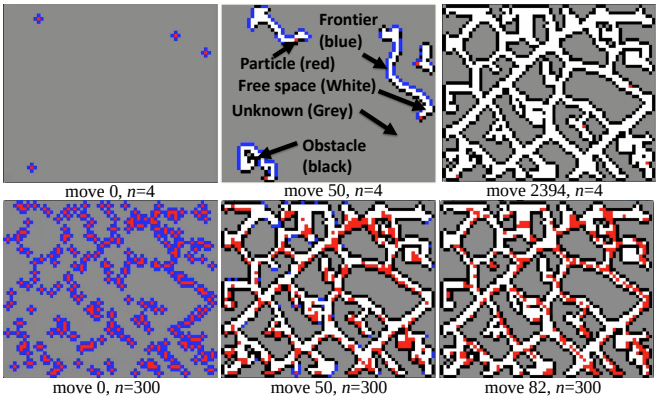

Fig. 1. Exploring a 2D environment with 500 blank spaces with $n = 4$ (top) and $n = 300$ particles, all controlled by a external global force. After 82 moves, the 300 particles have explored the entire space, while the 4 particles require a total of 2394 moves to fully cover the area.

The paper is arranged as follows. After a review of recent related work in Sec. II, we introduce the algorithms to perform the mapping and coverage in Sec. III. Sec. IV describes implementations of the algorithms in simulation and theoretical validation of efficiency. We discuss the performance of the algorithms on parameters which determine efficiency and also provide directions for further research in Sec. V.

II. Related Work

Coverage using one robot is a canonical robotics problem [12]. It has been studied in-depth for applications include lawn mowing, harvesting, floor cleaning, 3D printing, robotic painting and others.

Coverage means the robot has passed within $d/2$ of every location in the workspace where $d/2$ is the radius of the robot's footprint. Coverage with a swarm of robots is a key ability for a range of applications because swarms have higher fault tolerance and reduce completion time.

*This work was supported by the National Science Foundation under Grant No. [IIS-1553063] and [IIS-1619278].

A. Mahadev and A. Becker are with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204-4005 USA aviswanathanmahadev@uh.edu, atbecker@uh.edu S. Fekete and D. Krupke are with the Dept. of Computer Science, TU Braunschweig, Mühlenpfordtstr. 23, 38106 Braunschweig, Germany, s.fekete@tu-bs.de, d.krupke@tu-bs.de

Correspondingly, it has been studied from a control-theoretic perspective in both centralized and decentralized approaches. For examples of each, see CITE, and [13] where the coverage is performed in such a way that robots are not aware of each other's existence but always cover a cell that is a non-critical point which does not disconnect the graph.

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 [14] 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. This method closely resembles our approach in the sense that robots try to move in the same direction as long as possible. In our problem of interest the particles will always move in the same direction. The frontier cells exploration mentioned in [15] is an algorithm that is highly explorative as target locations to expand are selected using information from each robot, and the robot's vicinity. 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.

However, 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 [4], [16], [17] cannot have onboard computation. So we will be referring to them as particles in this paper.

Instead, this paper focuses on centralized techniques that apply the same control input to each member of the swarm.

III. Theory

This section examines the problem of mapping with uniform inputs in 1 and 2 dimensions.

A. Mapping in 1D

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

1) 1D mapping with 1 particle: A particle initialized uniformly randomly in a linear free-space m units wide. To map this region the particle needs to choose one direction, move until it hits a boundary, and then switch direction and move until it reaches 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 mth 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.

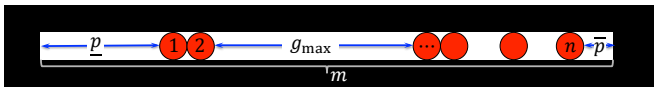

Fig. 2. Exploring a 1D environment of size m with n particles. Here $m = 20$ and $n = 6$. $\underline{p} = 4$, $\overline{p} = 1$ and $g_{max} = 7$

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

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

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. 2, 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 the $\overline{p}$ 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

$$\frac{1}{\binom{m}{n}}\sum_{\underline{p}=1}^{m-n+1}\sum_{\overline{p}=\underline{p}+n-1}^{m}\binom{\overline{p}-\underline{p}-1}{n-2}\left(2\underline{p}+m-\overline{p}+1\right) = \frac{3(1+m)}{1+n} \tag{2}$$

To fully *cover* 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 [18], but a tight bound when $m > n\log n$ is $\frac{m-n}{n+1} + \Theta\left(\sqrt{\frac{(m-n)\log(n+1)}{n+1}}\right)$ [19].

3) 1D mapping with scan and move costs: Often scanning (imaging) and moving the particles costs time and energy. When controlling particles with MRI as in [20], 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. This is a bicriteria problem, where the objective is to minimize the sum of the total costs for movement and scanning.

For simplicity we only consider moving left. This analysis can be applied in both directions. Assume the left boundary is located u units to the left of the leftmost particle. The optimal, yet unachievable, solution is to scan the workspace to detect the particle positions, next move $u+1$ units to the left, then scan to detect that the leftmost particle has only moved u units and thus has encountered the wall.

If we have weighted cost, a movement in uniform steps relative to the scan costs results in a constant competitive ratio. If we want to optimize the bicriteria problem, the task becomes more complicated. A constant competitive ratio for the scan costs is obviously impossible. We define *scan point* as a ??????. Setting the i-th scan point to k^i results in a $O(k)$ competitive ratio for the movement and a $O(\log n)$ competitive ratio for the number of scans where n is the position of the wall. It is possible to obtain a $O(\log n/W(\log n))$ competitive ratio for the bicriteria problem by making the i-th scan at position i^i. In this case, the competitive ratio for movement as well as scans is $\theta(\log n/W(\log n))$.

Let us first consider the number of scans. We found the boundary between steps j and $j+1$, i.e. $j^j \leq n < (j+1)^{j+1}$. This implies that $j \leq \log n/W(\log n)$ (the inverse of j^j), hence $\log n/W(\log n)+1$ scans have been made while the optimum are 2 scans. The moved distance is $(j+1)^{j+1}$ while the optimum is $n \geq j^j$. Hence, the ratio

$$\frac{(j+1)^{(j+1)}}{j^j} = (j+1)\frac{(j+1)^j}{j^j} \tag{3}$$

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

Since $j \leq \log n/W(\log n)$, we obtain

$$\frac{(j+1)^{(j+1)}}{j^j} \leq \mathrm{e}\frac{logn}{W(\log n)} + \mathrm{e} \tag{4}$$

for $j > 0$.

B. Mapping in 2D

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 boundaries have been reached. As long as the freespace is connected, DFS is the optimal solution to mapping. Even if the environment is known in advance, the problem is NP-hard as can be shown by a trivial reduction to Hamiltonian Paths in Grid Graphs [21]. One can easily show that a simple depth-first-search 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, the depth-first-search hence 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 ($2*m+1$ 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 PTAS is known [22]. If the environment is unknown, as it is in our 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 [23]).

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. For each particle, at the beginning there can be at most four frontiers to be explored. As the particles begin to move, the number of frontier cells explored and the number of free cells identified increases. If we implement a random move algorithm as described in Alg. 1, 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 nodes 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. This method requires minimal computation and is probabilistically complete, so eventually the swarm maps the free space [24]. However, this method of mapping is inefficient, resulting in long mapping times, especially with small numbers of particles in large, torturous freespaces.

Algorithm 1 RANDOMMOVES($\mathbf{M}, \mathbf{r}$)

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

A better way to map the world would be to move particles towards unexplored nodes. We could choose one particle as a lead particle and base all motion on the location of that particle. In Alg. 2, one of the particles is selected as the leader. As long as the number of frontiers to be visited is atleast one, the algorithm proceeds by generating a $moveSeq$ from the current position of the elect particle to the nearest frontier node. The $moveSeq$ is generated by a DIJKSTRA shortest path algorithm to which we feed the map $\mathbf{M}$, source $lead$, and the nodes FRONTIER($\mathbf{M}$). A representative $moveSeq$ is $\langle u, r, d, d, r, u, \ldots\rangle$. The list of moves in $moveSeq$ are implemented by iterativly calling the MOVE&UPDATE function for the length of the list. This algorithm will surely explore the target frontier within the

length of the list $moveSeq$. Every time MOVE&UPDATE is implemented, the nearest frontier is updated and $moveSeq$ is also updated because as the particles start to move, the target frontier in each step might be explored by another particle which is not the elect particle. So, usually, with large swarm size it is possible that the whole $moveSeq$ need not be implemented. Only if the elect particle is the nearest particle to the target node, the complete $moveSeq$ is implemented. This is the case in Alg. 3 in which all the particles are considered and the $moveSeq$ is the list of moves required to move the swarm to it's closest frontier.

Algorithm 2 ELECTPARTICLE($\mathbf{M}, \mathbf{r}$)

```
lead ← RANDOM(r)
while |FRONTIER(M)|> 0 do
    moveSeq ← DIJKSTRA(M, lead, FRONTIER(M))
    for iter := 1 to |moveSeq| step 1 do
        {M, lead} ←MOVE&UPDATE(moveSeq, M, lead)
    end for
end while
```

In Alg. 3 in each loop, all the moves in $moveSeq$ need to be implemented to explore the target frontier since it is the shortest possible route to a frontier cell. When the mapping is initiated, there will be $4k$ equally valid destinations that can be visited since all four sides of each particle will be a frontier cell. The algorithm arbitrarily choose a direction to move to. Once frontiers are classified, there can be at most three more frontiers added per particle if it has landed on a free space or there will be a reduction of one frontier if the frontier cell contained an obstacle. Since each $moveSeq$ guarantees classification of one frontier node into obstacle or free space. If there are n free spaces, the maximum possible frontier cells count will be $3(n-2)+6$. If for moveSeq generation left, up, right, down is the order of preference for direction of move in case of equidistant nodes, then as an example, for k particles the maximum number of moves required to completely map the linear map environment will be 4(n-k+1) if all the particles are arranged in adjacent cells on the right corner of the linear map. The simulation results in Section IV show that both map complexity and distribution affect the number of moves taken to map the work space. In Alg. 1 we used no information from the data except for checking completion. In Alg. 2 we use the location and distance data from one particle to map the work space. Finally in Alg. 3 we can push the performance of the mapping by utilizing all the data such as location of all particles and distance measurements from all frontiers.

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

```
while |FRONTIER(M)|> 0 do
    moveSeq ← DIJKSTRA(M, r, FRONTIER(M))
    for iter := 1 to |moveSeq| step 1 do
        {M, r} ←MOVE&UPDATE(moveSeq, M, r)
    end for
end while
```

Q7.Prove the greedy algorithm always works

Not sure how to do this. We have experiments. There are so many options for arranging the freespace.

The minimum time is a rectangle k units tall and n/k wide, with the n particles arranged along the left wall. 1 move left registers the left wall, repeating n/k moves up and right to find the top boundary and the right boundary, followed by $n/k-1$ moves down and left to find the bottom boundary, with one final move down to register the furthest left bottom boundary. This is $4(n/k)$ moves.

Q3.What is the worst case?

Q4.What is the complexity of the 2D problem?

Q5.Can we do better than the greedy algorithm for 2D coverage?

Q6.What is the running time for the greedy algorithm?

IV. SIMULATION

A. 1D mapping

Simulations were conducted in Mathematica, code at github [25]. Fig. 3 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.

B. 2D mapping

The mapping simulation was repeated 100 times for a given number of particles in a map with 5000 free spaces. In each run the particles were placed randomly throughout the workspace. For smaller ratios of particles/workspace the standard deviation of the completion time is large. As the number of particles increases from 100 to 5000 by increments of 100, the deviation decreases. The function of moves vs. number of particles is an exponentially decreasing curve. A comparison between the simulation results of the three algorithms is shown in Fig. 6. The log plot shows that all three algorithms have a almost logarithmic relationship between the number of free space m and number of particles k. The maximum number of moves required using the Closest Frontier algorithm was for k=100 with an average moves $\approx$1816 and standard deviation of 160 moves. This

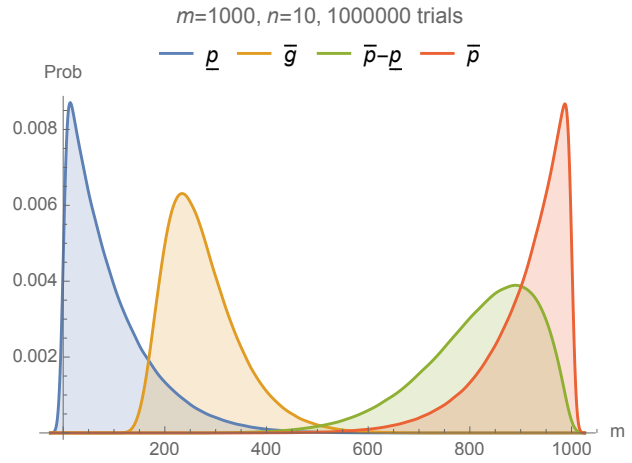

Fig. 3. With a connected freespace $m = 1000$ and $n = 10$ particles, the distributions for the gap before the first $\overline{p}$ and after the last $\underline{p}$ gaps are symmetric. The maximum gap $\overline{g} \approx 250$.

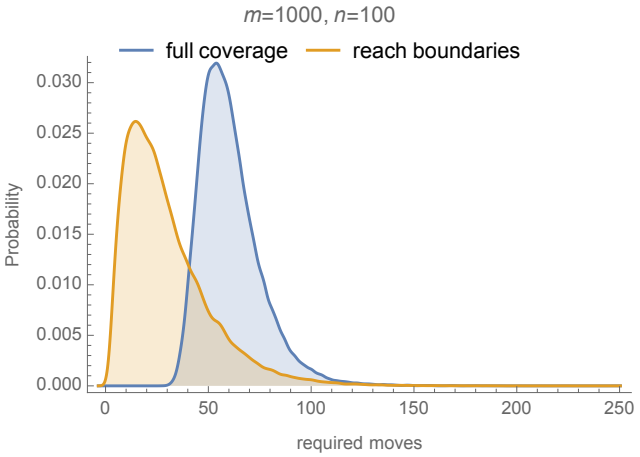

Fig. 4. Full coverage requires 60.7 moves on average, while reaching the boundaries requires only 29.8.

reduces to four moves with 0 standard deviation when n= 5000 (the total number of free spaces).

The comparison plot between the mapping of the three maps *complex*, *empty rectangle* and *linear* shows that the complex map requires the most moves. In the Fig. 5 there is an observable difference in moves between the linear and rectangular workspaces. One reason is because the number of useful moves is reduced in the linear case, where only left and right are useful moves. Up and down eliminate boundaries, but never discover free areas. Whereas inthe *empty rectangle* case most of the moves are moves which add to the number of mapped spaces. 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.

V. Conclusion and Future Work

This paper presented techniques for controlling the particle swarms in 1D and 2D grids. These particles can be tracked

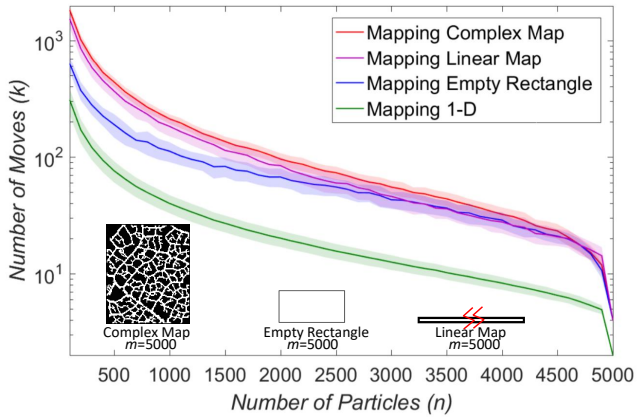

Fig. 5. Comparison of mapping 2D maps of three types using the Closest Frontier algorithm and the 1-D mapping of the linear map. The complex map, empty rectangle map, and linear map (similar to a 1D map, but with obstacles to detect on top and bottom) have 5000 free spaces.

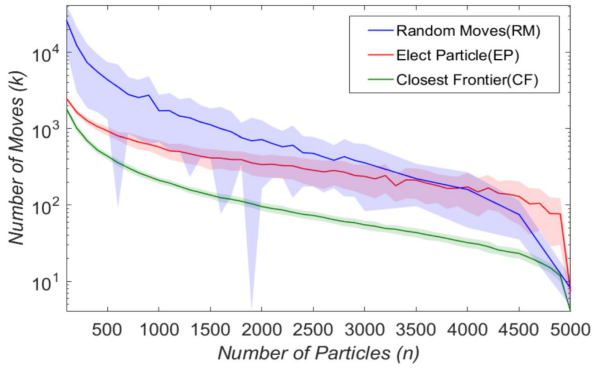

Fig. 6. Comparison of three Algorithms - Random Moves (Blue), Elect Particle (Red) and Closest Frontier (Green) for mapping the 2D Complex Map of 5000 free spaces.

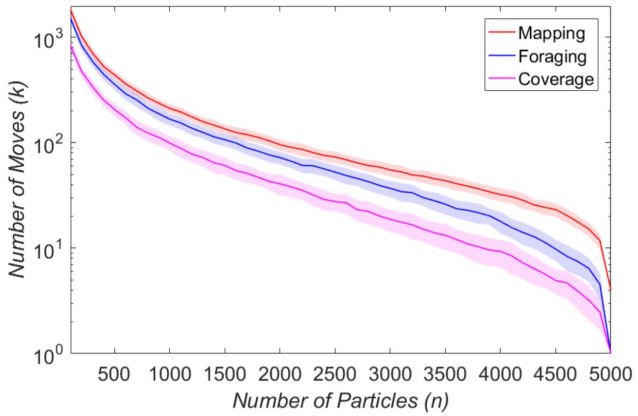

Fig. 7. Comparison of three related problems: coverage, mapping, and foraging on the complex 2D map.

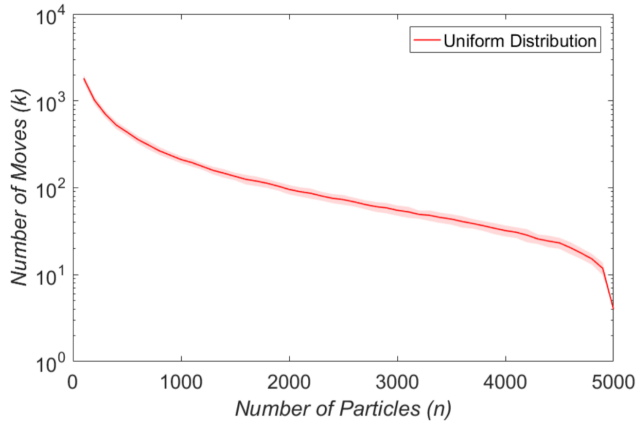

Fig. 8. Comparison with different distributions: flood-fill, region fill, and uniform distribution for mapping on the complex 2D map.

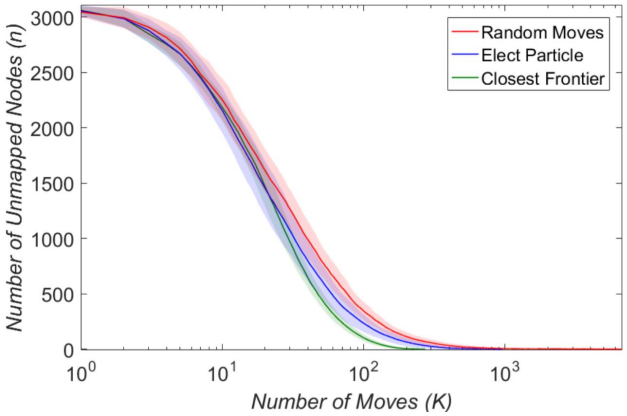

Fig. 9. Performing mapping on the complex 2D map with $n = 500$ particles. Random moves requires an average of ?? moves. Elect particle requires ?? moves and Closest Frontier requires ?? moves

and controlled by an external agent, but control inputs are applied uniformly so that each particle experiences the same applied forces. This paper presented benchmark algorithms for (1) mapping: building a representation of the free and obstacle regions of the workspace; (2) foraging: ensuring at least one particle reaches each of a set of desired locations; and (3) coverage: ensuring every free region on the map is visited by at least one particle. These problems are inspired by challenges when using contrast particles for MR imaging.

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.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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

[5] S. Martel, S. Taherkhani, M. Tabrizian, M. Mohammadi, D. de Lanauze, and O. Felfoul, "Computer 3D controlled bacterial transports and aggregations of microbial adhered nano-components," *Journal of Micro-Bio Robotics*, vol. 9, no. 1-2, pp. 23–28, 2014.

[6] P. S. S. Kim, A. Becker, Y. Ou, A. A. Julius, and M. J. Kim, "Imparting magnetic dipole heterogeneity to internalized iron oxide nanoparticles for microorganism swarm control," *Journal of Nanoparticle Research*, vol. 17, no. 3, pp. 1–15, 2015.

[7] B. R. Donald, C. G. Levey, I. Paprotny, and D. Rus, "Planning and control for microassembly of structures composed of stress-engineered MEMS microrobots," *The International Journal of Robotics Research*, vol. 32, no. 2, pp. 218–246, 2013. [Online]. Available: http://ijr.sagepub.com/content/32/2/218.abstract

[8] A. Ghosh and P. Fischer, "Controlled propulsion of artificial magnetic nanostructured propellers," *Nano Letters*, vol. 9, no. 6, pp. 2243–2245, 2011/10/23 2009. [Online]. Available: http://dx.doi.org/10.1021/nl900186w

[9] Y. Ou, D. H. Kim, P. Kim, M. J. Kim, and A. A. Julius, "Motion control of magnetized tetrahymena pyriformis cells by magnetic field with model predictive control," *Int. J. Rob. Res.*, vol. 32, no. 1, pp. 129–139, Jan. 2013.

[10] F. Qiu and B. J. Nelson, "Magnetic helical micro-and nanorobots: Toward their biomedical applications," *Engineering*, vol. 1, no. 1, pp. 21–26, 2015.

[11] 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.

[12] H. Choset, "Coverage for robotics–a survey of recent results," *Annals of mathematics and artificial intelligence*, vol. 31, no. 1, pp. 113–126, 2001.

[13] 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.

[14] D. Latimer, S. Srinivasa, V. Lee-Shue, S. Sonne, H. Choset, and A. Hurst, "Towards sensor based coverage with robot teams," vol. 1, pp. 961–967, 2002.

[15] B. Yamauchi, "Frontier-based exploration using multiple robots," pp. 47–53, 1998.

[16] S. Martel, "Magnetotactic bacteria for the manipulation and transport of micro-and nanometer-sized objects," *Micro-and Nanomanipulation Tools*, 2015.

[17] 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.

[18] 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.

[19] 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

[20] 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.

[21] 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.

[22] 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.

[23] C. Icking, T. Kamphans, R. Klein, and E. Langetepe, "Exploring simple grid polygons," in *International Computing and Combinatorics Conference*. Springer, 2005, pp. 524–533.

[24] 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.

[25] A. Mahadev and A. T. Becker, “”Mapping with Particles under Uniform Inputs”, https://github.com/aabecker/ magneticcontroller/tree/master/ massiveuniformcontrol/iros2017mapping/code,” Mar. 2017.