Reconfiguring Massive Particle Swarms with Limited, Global Control

Aaron Becker[*] Erik D. Demaine[†] Sándor P. Fekete[‡]
Golnaz Habibi[*] James McLurkin[*]

Abstract

In this paper we investigate control of a large swarm of mobile particles (such as robots, sensors, or building material) that move in a 2D workspace using a global input signal, e.g., provided by gravity or a magnetic field. Upon activation, each robot moves in the same direction, maximally until it hits a stationary obstacle or another stationary robot. In a workspace with only simple exterior boundaries, this system model is of limited use because it has only two controllable degrees of freedom—all robots receive the same inputs and move uniformly. We show that adding a maze of obstacles to the environment can make the system drastically more complex and more useful.

We prove that it is NP-hard to decide whether a given initial configuration can be transformed into a desired target configuration, if we are given a fixed set of stationary obstacles. On the positive side, we provide constructive algorithms to design workspaces that efficiently implement arbitrary permutations between different configurations.

Keywords: Robot swarm, nano-particles, uniform inputs, parallel motion planning, complexity, array permutations.

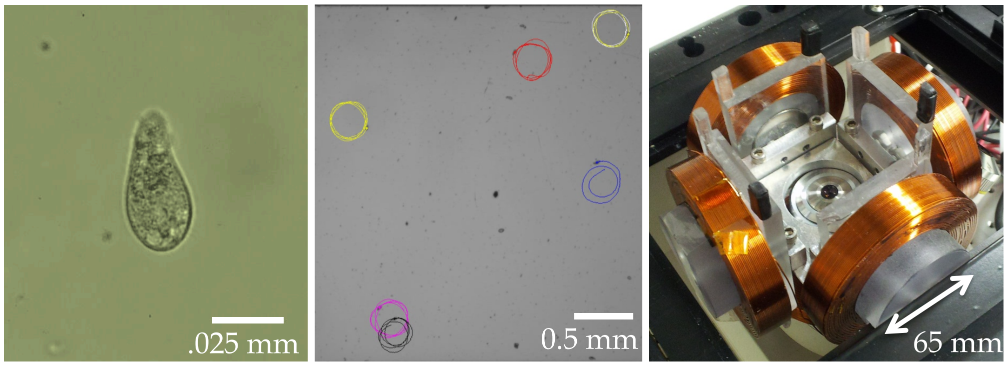
1 Introduction

One of the fundamental algorithmic paradigm shifts of recent years has been the move from individual, powerful computers and robots to large swarms of distributed processors, sensor nodes, and robots, each with limited computing power, only local control, and incomplete information, but with the objective of achieving complex global objectives. The inception and development of ALGOSENSORS has been to pursue these new challenges, and has spread to a wide range of topics from sensor networks, distributed networks and robotics, and experiments.

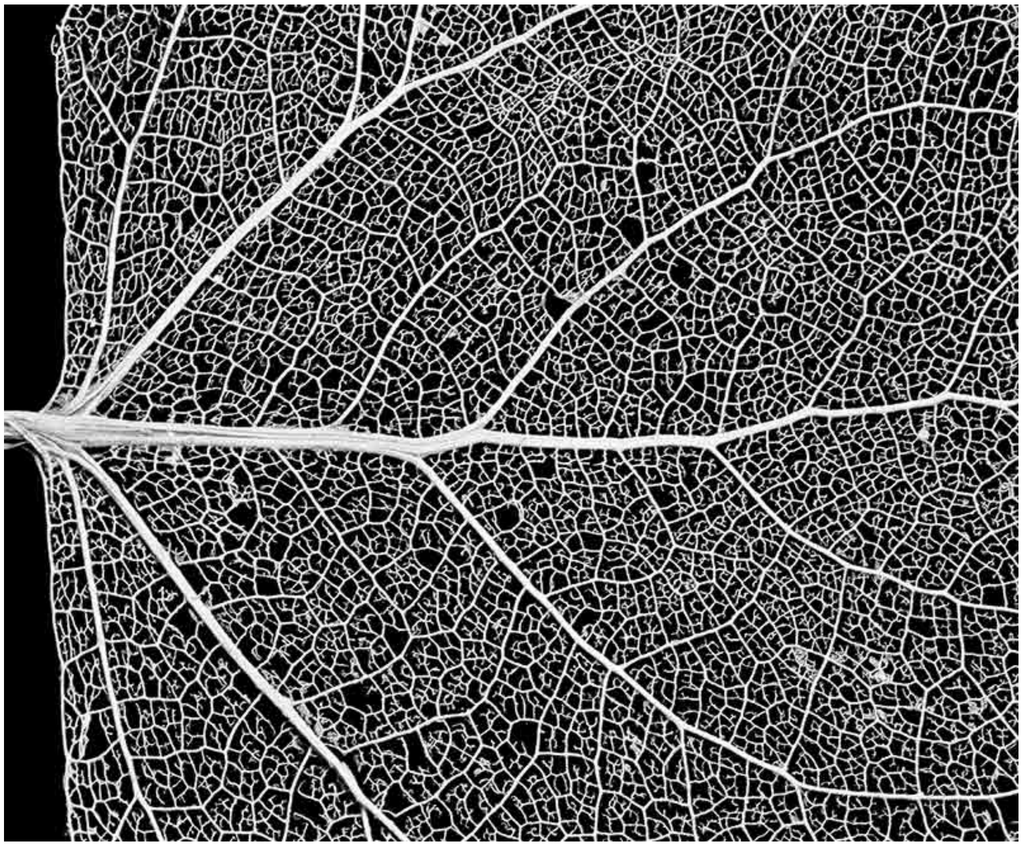
Since the first visions of massive sensor swarms, more than ten years of work on sensor networks have yielded considerable progress with respect to hardware miniaturization; however, we are still far away from the visions of "Smart Paint" [1] or "Smart Dust" [29], which triggered a considerable amount of theoretical research in the early years of ALGOSENSORS, e.g., our own work in [18, 19, 20, 31]. In the meantime, it has become clear that some of the impediments of shrinking hardware are not algorithmic (or pertaining to Computer Science in general), but rather of a physical nature.

However, during the same time, many practical aspects of distributed systems have developed to the point where new algorithmic challenges are triggered, and new possibilities arise. One development is the ability to design, produce, and control particles at the nanoscale. Compared to classical visions of sensor networks with stationary nodes, a new twist is the capability to consider *mobile* particles, allowing a wide range of possible application, e.g., in the medical field. Because the physics of motion at the nanoscale require overcoming a considerable amount of resistance, and the capacity of storing energy for computation,

[*]Department of Computer Science, Rice University, Houston, TX 77005, aabecker@gmail.com,{gh4,jm23}@rice.edu.
[†]Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA 02139, USA, edemaine@mit.edu.
[‡]Dept. of Computer Science, TU Braunschweig, Mühlenpfordtstr. 23, 38106 Braunschweig, Germany, s.fekete@tu-bs.de



(a) (Left, center) after feeding iron particles to ciliate eukaryon (*Tetrahymena pyriformis*) and magnetizing the particles with a permanent magnet, the cell can be turned by changing the orientation of an external magnetic field. (Right) using two orthogonal Helmholz electromagnets (left), Becker et al. demonstrated steering many living magnetized *T. pyriformis* cells [8]. All cells are steered by the same global field.



(b) Biological vascular network (cottonwood leaf) Royce Bair/Flikr/Getty Images. Given such a network along with initial and goal positions of N particles, is it possible to bring each particle to its goal position using a global control signal? Note that this arrangement is *not* a tree, but is a graph structure with loops. MATLAB code for driving n robots through this network available at http://www.mathworks.com/matlabcentral/fileexchange/42892.

Figure 1: (Top) State of the art in controlling small objects by force fields. (Bottom) A complex vascular network, forming a typical environment for the parallel navigation of small objects.

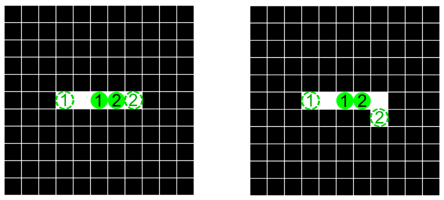


Figure 2: In this image, black cells are fixed, white cells are free, and solid green discs are individual particles, and goal positions are dashed green circles. For the simple world at left, it is impossible to maneuver both particles to end at their goals. The world at right has a finite solution: $\langle r, d, l \rangle$.

communication and motion control shrinks with the third power of object size, it is clear that classical approaches based on individual motion control cannot be applied.

The work in this paper is motivated by such new and current challenges arising in micro- and nano-robotics, where a global field is used to control many small agents. An example is using the global magnetic field from a MRI to guide magneto-tactic bacteria through a vascular network to deliver payloads at specific locations [9], and recent work using electromagnets to steer a magneto-tactic bacterium through a micro-fabricated maze [30].

Thus, we study the following basic problem: *Given a map of a vascular network, such as the one shown in Fig. 1b, along with initial and goal positions for each particle, does there exist a sequence of inputs that will bring each particle to its goal position?*

Here we study this problem on a two-dimensional grid, as a simple model of directional control. We assume that particles cannot be individually controlled, but rather, are all simultaneously given a message to travel maximally in a given direction. This assumption corresponds to situations with limited state feedback, or for robots that move at unpredictable speeds. Problems of this type are similar to sliding-block puzzles with fixed obstacles [28, 26, 12, 27], except that all particles receive the same control inputs.

More precisely, we consider the following scenario, which we call GlobalControl-ManyParticles:

1. Initially, the planar square grid is filled with some unit-square particles (each occupying a cell of the grid) and some fixed unit-square blocks.

2. All particles are commanded in unison: a valid command is "Go Up" (u),"Go Right" (r),"Go Down" (d), or "Go Left" (l). All particles move in the commanded direction until they hit an obstacle or another particle. A representative command sequence is $\langle u, r, d, l, d, r, u, \ldots \rangle$.

3. The goal is to get any particle to a specified position.

The algorithmic decision problem GlobalControl-ManyParticles is to decide whether a given puzzle is solvable. As it turns out, this problem is computationally difficult: we prove NP-hardness in Section 3. While this result shows the richness of our model (despite the limited control over the individual parts), it also constitutes a major impediment for constructive algorithmic work.

Thus we turn our attention to developing algorithmic tools for enabling global control by uniform commands. In the second part of the paper (Section 4), we develop several positive results. The underlying idea is to construct artificial obstacles (such as walls) that allow arbitrary rearrangements of a given two-dimensional particle swarm. For clearer notation, we will formulate the relevant statements in the language of matrix operations, which is easily translated into plain geometric language.

Our paper is organized as follows. After a discussion of related work in Section 2, we provide our main result on the complexity of the problem in Section 3. We then present constructive algorithmic results in Section 4, and end with concluding remarks in Section 5.

2 Related Work

Large Robot Populations. Due to the efforts of roboticists, biologists, and chemists (e.g. [39], [37],[10]), it is now possible to make and field very large (10^3–10^{14}) populations of simple robots.

Potential applications for these robots include targeted therapy, sensing, and actuation. With large populations come two fundamental challenges: (1) how to perform state estimation for the robots, and (2) how to control these robots.

Traditional approaches often assume independent control signals for each robot, but each additional independent signal requires engineering and design work. This becomes more challenging as the robot size decreases. At the molecular scale, there is a bounded number of modifications that can be made. Especially at the micro- and nano-scales it is not practical to encode autonomy in the robots. Instead, the robots are controlled and interacted with using global control signals.

More recently, robots have been constructed with physical heterogeneity so that they respond differently to a global, broadcast control signal. Examples include *scratch-drive microrobots*, actuated and controlled by a DC voltage signal from a substrate [14]; magnetic structures with different cross-sections that could be independently steered [21]; *MagMite* microrobots with different resonant frequencies and a global magnetic field [22]; and magnetically controlled nanoscale helical screws constructed to stop movement at different cutoff frequencies of a global magnetic field [38]. In our previous work with robots that can be modeled as nonholonomic unicycles, we showed that an inhomogeneity in turning speed is enough to make even an infinite number of robots controllable with regard to position. All these approaches show promise, but they also require both excellent state estimation and heterogeneous robots. In addition, the controllers required at best a summation over all the robot states [7] and at worst a matrix inversion [5].

In this paper we take a very different approach. We assume a population of approximately identical planar particles (which could be small robots) and one global control signal consisting of a vector all particle should move along. This system is not controllable because the particles move uniformly. Implementing any control signal translates the entire group identically. However, an obstacle-filled workspace allows us to break symmetry [6]. In that work [6], we showed that if we can command the particles to move one unit distance at a time, some configurations have easy solutions. Given a large free space, we have an algorithm showing that a single obstacle is sufficient for position control of N particles (video of position control: `http://www.youtube.com/watch?v=5p_XIad5-Cw`). In this work we do not allow incremental position control, and particles move to their full extent at each actuation.

Amorphous computing [2] studies how *computational particles* distributed on a surface can be used to produce computational engines. In a similar manner, we show how to construct logic gates to perform computation in our system.

Dexterity Games. The problem we investigate is strongly related to dexterity puzzles—games that typically involve a maze and several balls that should be maneuvered to goal positions. *Pigs in Clover*, invented in 1880 by Charles Martin Crandall, involved steering four balls through 3 concentric incomplete circles [40].

These puzzles are dynamic and depend on the manual dexterity of the player. Our problem formulation also applies the same input to every agent, but imposes only kinematic restrictions on agents. This is most similar to the gravity-fed logic maze *Tilt*^{TM} `http://www.thinkfun.com/tilt`.

Computational Geometry: Robot Box-Pushing. Many variations of block-pushing puzzles have been explored from a computational complexity viewpoint, with a seminal paper proving NP-hardness by Gordon Wilfong in 1991 [43]. The general case of motion-planning when each command moves particles a single unit in a world composed of even a single robot and both *fixed* and *moveable* squares is in the complexity class PSPACE-COMPLETE [15] [27] [13]. Adding an additional robot does not decrease this complexity: given any single-robot problem, we can place a second robot in the boundary of the world and surround it with *fixed* squares without changing the original problem's complexity.

Ricochet Robots [16], Atomix [28], and PushPush [12] have the same constraint that robots when moved must move to their full extent. This constraint reflects physical realities where, due to uncertainties in sensing, control application, and robot models, precise quantified movements in a specified direction is not possible, but the input can be applied for a long period of time and be guaranteed that the robots will move to their fullest extent. In these games the robots move to their full extent with each input, but each robot can be actuated individually. The complexity of the problem with global inputs to all robots has remained an open problem.

Sensorless Manipulation. We have drawn particular inspiration from work on sensorless manipulation [17]. The basic idea in this work is to explicitly maintain the set of all possible robot configurations and to select a sequence of actions that reduces the size of this set and drives it toward some goal configuration. Carefully selected primitive operations can make this easier. For example, sensorless manipulation strategies often use a sequential composition of primitive operations, "squeezing" a part either virtually with a programmable force field or simply between two flat, parallel plates [25]. Sensorless manipulation strategies also may take advantage of limit cycle behavior, for example engineering fixed points and basins of attraction so that parts only exit a feeder when they reach the correct orientation [33, 35]. These two strategies have been applied to a much wider array of mechanisms such as vibratory bowls and tables [23, 42] or assembly lines [25, 3, 41], and have also been extended to situations with stochastic uncertainty [24, 34] and closed-loop feedback [4, 36].

Parallel Algorithms: SIMD. Another related area of research is Single Instruction Multiple Data (SIMD) parallel algorithms [32]. In this model, multiple processors are all fed the same instructions to execute, but they do so on different data. This model allows, for example, executing commands selectively only on certain processors, but still spending the time cost on all processors.

Our model is actually more extreme: the particles all respond in effectively the same way to the same instruction. The only difference is their location, and which obstacles or particles will thus block them. In some sense, our model is essentially Single Instruction, Single Data, Multiple Location.

3 Complexity

We prove that the general problem is computationally intractable:

Theorem 1. GlobalControl-ManyParticles *is NP-hard: given an initial configuration of movable particles and fixed obstacles, it is NP-hard to decide whether any particle can be moved to a specified location.*

Proof. We prove hardness by a reduction from 3SAT. Suppose we are given n Boolean variables $x_1, x_2, \ldots, x_n$, and m disjunctive clauses $C_j = U_j \vee V_j \vee W_j$, where each literal U_j, V_j, W_j is of the form x_i or $\neg x_i$. We construct an instance of GlobalControl-ManyParticles that has a solution if and only if all clauses can be satisfied by a truth assignment to the variables.

Variable gadgets. For each variable x_i that appears in k_i literals, we construct k_i instances of the protected variable gadget i shown in Figure 3b, with a particle initially at the top of the gadget. The gadget consists of a tower of n levels, designed for the overall construction to make n total variable choices. These choices are intended to be made by a move sequence of the form $\langle d,\ l/r,\ d,\ l/r,\ \ldots,\ d,\ l/r,\ d,\ l\rangle$, where the ith l/r choice corresponds to setting variable x_i to either true (l) or false (r). Thus protected variable gadget i ignores all but the ith choice by making all other levels lead to the same destination via both l and r. The ith level branches into two destinations, chosen by either l or r, which correspond to x_i being set true or false, respectively.

In fact, the command sequence may include multiple l and r commands in a row, in which case the last l/r before a vertical u/d command specifies the final decision made at that level, and the others can be ignored. The command sequence may also include a u command, which undoes a d command if done immediately after, or else does nothing; thus we can simply ignore the u command and the immediately preceding d if it exists. We can also ignore duplicate commands (e.g., d, d becomes d) and remove any initial

l/r command. After ignoring these superfluous commands, assuming a particle reaches one of the output channels, we obtain a sequence in the canonical form $\langle d,\ l/r,\ d,\ l/r,\ \ldots,\ d,\ l\rangle$ as desired, corresponding uniquely to a truth assignment to the n variables. (If no particle reaches the output port, it is as if the variable is neither true nor false, satisfying no clauses.) Note that all particles arrive at their output ports at exactly the same time.

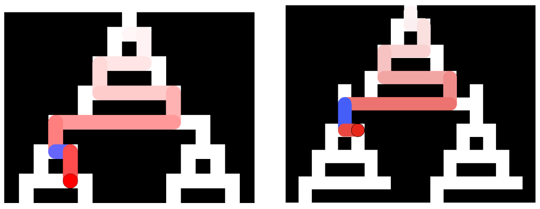

(a) variable (b) protected variable

Figure 3: Variable gadget that executes by a sequence of $\langle d, l/r\rangle$ moves. The ith l/r choice sets the variable to true or false by putting the ball in a separate column. The gadget is designed to respond to the ith choice but ignore all others. This lets us make several copies of the same variable by making multiple gadgets with the same i. In the figure $i = 4$ and $n = 6$. In the unprotected variable gadget, a $d, l/r$ move latches the ith choice, in the protected gadget an u/d is sufficient to latch the choice. The latching move is shown in blue.

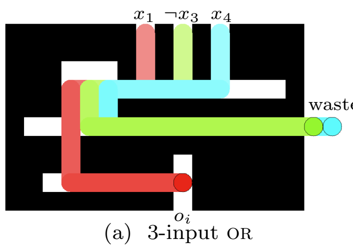
Clause gadgets. For each clause, we use the OR gadget shown in Figure 4a. The OR gadget has three inputs corresponding to the three literals. For each positive literal of the form x_i, we connect the corresponding input to the left output of an unused instance of protected variable gadget i. For each negative literal of the form $\neg x_i$, we connect the corresponding input to the right output of an unused instance of protected variable gadget i. (In this way, each variable gadget gets used exactly once.)

We connect the variable gadget to the OR gadget in a simple way: place the variable gadget above the clause so as to align the vertical output and input channels, and join them into a common channel. To make room for the three variable gadgets, we simply extend the black areas separating the three input channels in the OR gadget. The unused output channel of each variable gadget simply ends; by the properties of the variable gadget, any particle reaching that end cannot later reach the other output channel.

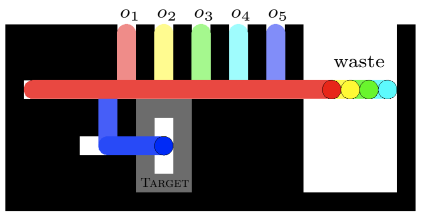
If any input channel of the OR gadget has a particle, then it can reach the output port by the move sequence $\langle d, l, d, r\rangle$. Furthermore, because variable gadgets place all particles on their output ports at the same time, if more than one particle reaches the OR gadget, they will move in unison as drawn in Figure 4a, and only one can make it to the output port; the others will be stuck in the "waste" row, even if extra $\langle l, r, u, d\rangle$ commands are interjected into the intended sequence. Hence, a single particle can reach the output of a clause if and only if that clause (i.e., at least one of its literals) is satisfied by the variable assignment.

Check gadget. At the end, we check that all clauses were simultaneously satisfied by the variable assignment, using the m-input AND gadget shown in Figure 4b. Specifically, we place the clause gadgets along a horizontal line, and connect their vertical output channels to the vertical input channels of the check gadget. Again we can align the channels by extending the black areas that separate the input channels of the AND gadget.

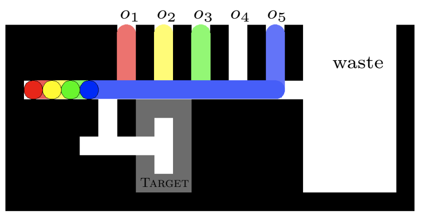
The intended solution sequence for the AND gadget is $\langle d, l, d, r\rangle$. The AND gadget is designed with the downward channel exactly m units to the right from the left wall, and $> 2m$ units from the right wall, so for any particle to reach the downward channel (and ultimately, the target location), at least m particles must be presented as input. Because each input channel will present at most one particle (as argued in a clause), a particle can reach the final destination if and only if all m clauses output a particle, which is possible exactly when all clauses are satisfied by the variable assignment.



(a) 3-input OR

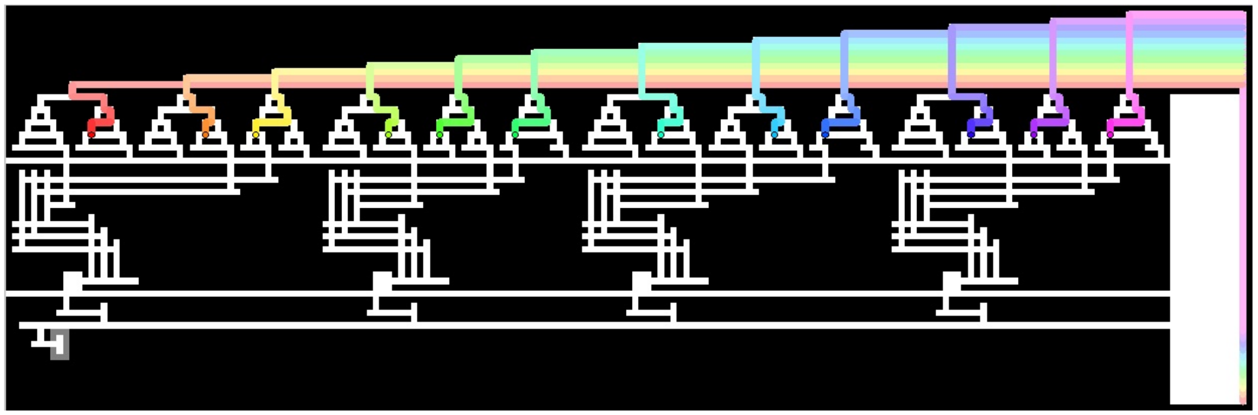


(b) m-input AND, *output*=TRUE

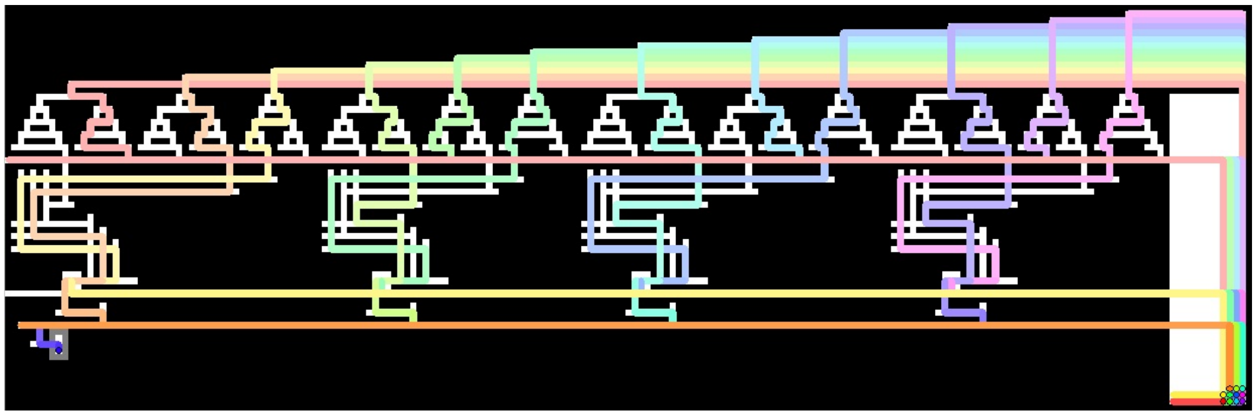


(c) m-input AND, *output*=FALSE

Figure 4: Gadgets that use the cycle $\langle d, l, d, r\rangle$. The 3-input OR gadget outputs one particle if at least one particle enters in an input line, and sends any extra particle to be recycled. The m-input AND gadget outputs one particle to the TARGET LOCATION, marked in gray, if at least m inputs are TRUE. Here $m = 5$. Excess particles are recycled.



(a) Setting each variable



(b) Satisfying the 3SAT problem

Figure 5: Combining 12 variable gadgets, three 3-input OR gadgets, and an m-input AND gadget to realize the 3SAT expression $(A \vee \neg B \vee C) \wedge (B \vee \neg C \vee D) \wedge (\ A \vee B \vee D) \wedge (\neg A \vee \neg C \vee D)$.

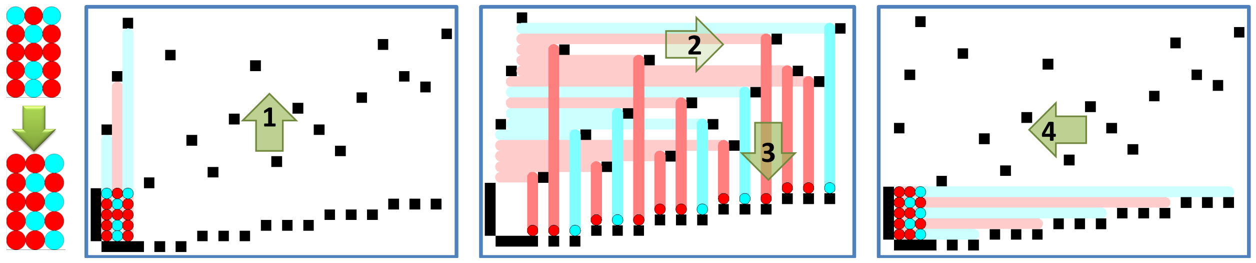


Figure 6: In this image for $N = 15$, black cells are obstacles, white cells are free, and colored discs are individual particles. The world has been designed to permute the particles between 'A' into 'B' every four steps: $\langle u, r, d, l\rangle$. See video at http://www.youtube.com/watch?v=3tJdRrNShXM

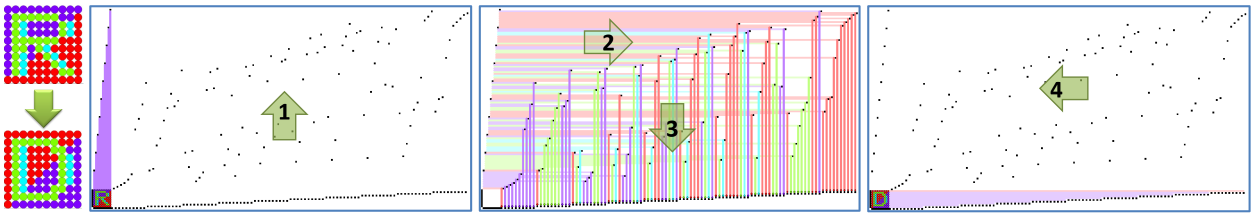


Figure 7: In this larger example with $N = 100$, the different control sections are easier to see than in Fig. 6. (1) The staggered obstacles on the left spread the matrix vertically, (2) the scattered obstacles on the right permute each element, and (3) the staggered obstacles along the bottom reform each row, which are collected by at (4). The cycle resets every 740 iterations. See `http://www.youtube.com/watch?v=eExZO0HrWRQ` for an animation of this gadget.

This completes the reduction and the NP-hardness proof. □

We conjecture that GlobalControl-ManyParticles is in fact PSPACE-complete. One approach would be to simulate nondeterministic constraint logic [26], perhaps using a unique move sequence of the form $\langle d,\ l/r,\ d,\ l/r,\ \dots\rangle$ to identify and "activate" a component. One challenge is that all gadgets must properly reset to their initial state, without permanently trapping any particles.

4 Matrix Permutations

This section investigates a complementary problem. Given the same particle and world constraints as before, what types of control are possible and economical if we are free to design the environment?

First we describe an arrangement of obstacles that implement an arbitrary matrix permutation in four commands. Then we provide efficient algorithms for sorting matrices, and finish with potential applications.

4.1 Designing Workspace for a Single Permutation

For our purposes, a *matrix* is a 2D array of particles (each possibly a different color). For an $a_r \times a_c$ matrix A and a $b_r \times b_c$ matrix B, of equal total size $N = a_r \cdot a_c = b_r \cdot b_c$, a *matrix permutation* assigns each element in A a unique position in B. Figs. 6 and 7 show example constructions that execute matrix permutations of total size $N = 25$ and 100, respectively.

For simplicity of exposition, we assume henceforth that all matrices are $n \times n$ squares.

Theorem 2. *Any matrix permutation can be executed by a set of obstacles that transforms matrix A into matrix B in just four moves. For N particles, the arrangement requires $(3N + 1)^2$ space, $4N + 1$ obstacles, and $12N/\text{speed}$ time.*

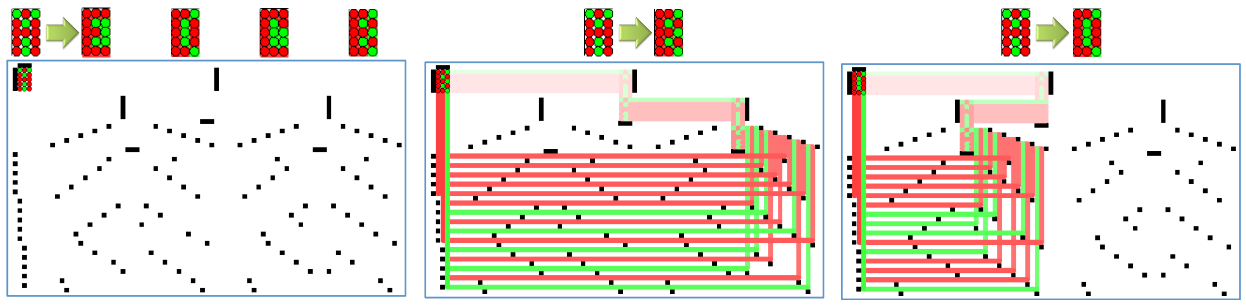


Figure 8: For any set of k fixed, but arbitrary permutations of $n \times n$ pixels, we can construct a set of $O(kN)$ obstacles, such that we can switch from a start arrangement into any of the k permutations using at most $O(\log k)$ force-field moves. Here $k = 4$ and transforms 'A' into 'B', C', 'D', or 'E' in eight moves: $\langle r, d, (r/l), d, (r/l), d, l, u \rangle$.

Proof. Refer to Figures 6 and 7 for examples. The move sequence is $\langle u, r, d, l \rangle$. **Move 1:** We place n obstacles, one for each column, spaced n units apart, such that moving u spreads the particle array into a staggered vertical line. Each particle now has its own row. **Move 2:** We place N obstacles to stop each particle during the move r. Each particle has its own row and can be stopped at any column by its obstacle. We leave an empty column between each obstacle to prevent collisions during the next move. **Move 3:** Moving d arranges the particles into their desired rows. These rows are spread in a staggered horizontal line. **Move 4:** Moving l stacks the staggered rows into the desired permutation, and returns the array to the initial position. □

By reapplying the same permutation enough times, we can return to the original configuration. The permutations shown in Fig. 6 return to the original image in 2 cycles, while Fig. 7 requires 740 cycles. For a two-color image, we can always construct a permutation that resets in 2 cycles, but this does not extend to images with more than 2 colors.

4.2 Designing One Workspace for Arbitrary Permutations

There are various ways in which we can exploit Theorem 2 in order to generate larger sets of (or even all) possible permutations. As it turns out, there is a tradeoff between the number of introduced obstacles and the number of moves required for realizing a permutation.

We start with obstacle sets that require only few moves.

Theorem 3. *For any set of k fixed, but arbitrary, permutations of $n \times n$ pixels, we can construct a set of $O(kN)$ obstacles, such that we can switch from a start arrangement into any of the k permutations using at most $O(\log k)$ force-field moves.*

Proof. Build a binary tree of depth $\log k$ for choosing between the permutations by a sequence of $\langle r$, d, (r/l), d, (r/l), …, d, (r/l), d, l, $u\rangle$ with $\log k$ (r/l) decisions between the initial prefix $\langle r, d\rangle$ and final suffix $\langle d, l, u\rangle$. This gets the pixels to the set of obstacles for performing the appropriate permutation. □

Corollary 4. *For any $\varepsilon > 0$, we can construct a set of $(N!)^{\varepsilon}$ obstacles such that any permutation of $n \times n = N$ pixels can be achieved by at most $O(N \log N)$ force-field moves.*

Proof. Follows from Theorem 3 by $k = (N!)^{\varepsilon}/N$. □

Now we proceed to more economical sets of obstacles, with arbitrary permutations realized by clockwise and counterclockwise move sequences. We make use of the following lemma, which shows that two base permutations are enough to generate any desired rearrangement.

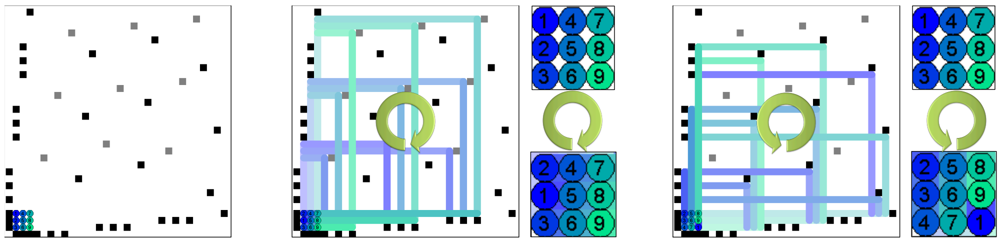


Figure 9: Repeated application of two base permutations can generate any permutation, when used in a manner similar to Bubble Sort. The obstacles above generate the base permutation $p = (12)$ in the CW direction $\langle u, r, d, l\rangle$ and $q = (12 \cdots N)$ in the CCW direction $\langle r, u, l, d\rangle$.

Lemma 5. *Any permutation of N objects can be generated by the two base permutations $p = (12)$ and $q = (12 \cdots N)$. Moreover, any permutation can be generated by a sequence of length at most N^2 that consists of p and q.*

Proof. See Fig. 9. Similar to Bubble Sort, we use two nested loops of N. Each move consists of performing q once, and p when appropriate. □

This allows us to establish the following result.

Theorem 6. *We can construct a set of $O(N)$ obstacles such that any $n \times n$ arrangement of N pixels can be rearranged into any other $n \times n$ arrangement π of the same pixels, using at most $O(N^2)$ force-field moves.*

Proof. Use Theorem 2 to build two sets of obstacles, one each for p and q, such that p is realized by the sequence $\langle u, r, d, l\rangle$ (clockwise) and q is realized by $\langle r, u, l, d\rangle$ (counterclockwise). Then we use the appropriate sequence for generating π in $O(N^2)$ moves. □

Using a larger set of generating base permutations allows us to reduce the number of necessary moves. Again, we make use of a simple base set for generating arbitrary permutations.

Lemma 7. *Any permutation of N objects can be generated by the N base permutations $p_1 = (12), p_2 = (13), \ldots, p_{N-1} = (1(N-1))$ and $q = (12 \cdots N)$. Moreover, any permutation can be generated by a sequence of length at most N that consists of the p_i and q.*

Proof. Straightforward, analogous to Theorem 6: in each step i, apply q once, and swap element $\pi(i)$ into position i. □

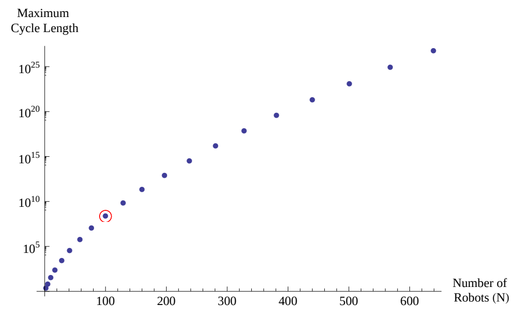
Theorem 8. *We can construct a set of $O(N^2)$ obstacles such that any $n \times n$ arrangement of N pixels can be rearranged into any other $n \times n$ arrangement π of the same pixels, using at most $O(N \log N)$ force-field moves.*

Proof. Use Theorem 2 to build N sets of obstacles, one each for $p_1, \ldots, p_{N-1}, q$. Furthermore, use Lemma 7 for generating all permutations with at most N different of these base permutation, and Theorem 3 for switching between these $k = N$ permutations. Then we can get πs with at most N cycles, each consisting of at most $O(\log N)$ force-field moves. □

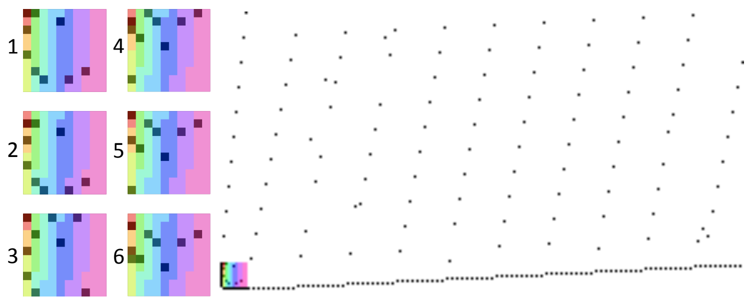
This is best possible with respect to the number of moves, in the following sense:

Theorem 9. *Suppose we have a set of obstacles such that any permutation of an $n \times n$ arrangement of pixels can be achieved by at most M force-field moves. Then M is at least $\Omega(N \log N)$.*

Proof. Each permutation must be achieved by a sequence of force-field moves. Because each decision for a force-field move $\langle u, d, l, r\rangle$ partitions the remaining set of possible permutations into at most four different subsets, we need at least $\Omega(\log(N!)) = \Omega(N \log N)$ such moves. □



(a) Absolute encoder cycles



(b) Example encoder

Figure 10: (a) Using a permutation gadget as an absolute encoder—the cycle length increases rapidly as the number of particles increases. (b) An obstacle arrangement to permute a 10×10 matrix in four moves $\langle u, r, d, l\rangle$, with a cycle length over 200 million. Shown at left are the first 6 permutations. The cycles are shown in different colors.

4.3 Physical Pseudo-Random Permutations and Animations

As shown in Figs. 6 and 7, a permutation gadget allows us to design a display that is hard-coded with a set of pictures.

A potential practical application uses these permutations as a physical absolute encoder or as a pseudo-random number generator. This application exploits the fact that these physical permutations are cyclic, and that we can design the cycle length. Applying the CW circular movements $\langle u, r, d, l\rangle$ in succession moves all the particles through one permutation.

The cycle length is the least common multiple of the permutation cycles in the transformation $A \mapsto B$. Given N particles, we want to partition the set of k permutation cycles in such a way that the sum $\sum_{i=1}^{k} n_i = N$ and maximizes $\mathrm{LCM}(n_1, n_2, \ldots, n_k)$.

This cycle length grows rapidly. For instance, using $N = 100$ particles, we can partition the particles into cycles of length {2, 3, 5, 7, 11, 13, 17, 19, 23}, see Fig. 10b. The LCM is 223,092,870. See [11] for a more in-depth look at the growth of the maximum cycle length as a function of N.

Animations. It would be useful if we could design permutations to generate sequences of images, e.g. "R", "o", "b", "o", "t". Surprisingly, there are sequences of just three images that cannot be constructed with a single permutation. Consider the three 5-particle arrangements □□■■■, ■□■□■, ■■□■□. Though permutations between any two exist, there is no single permutation that can generate all three. No single permutation can generate all possible permutations of the given particles. The example in Fig. 10b, with 100 particles, 9 painted black and the rest white, the maximum cycle length we can generate is of length $\approx 2\times10^8$, but for permutations of length N with repeated elements $N_1, N_2, \ldots$, the total number of permutations is

$$\frac{N!}{N_1!N_2!\ldots N_k!}$$

For the examples above, there are $100!/(9!91!) \approx 2 \times 10^{12}$ permutations possible.

Reversible Permutations. The permutations generators shown in Fig. 10b are one-way devices. Attempting to drive them in reverse $\{l, d, r, u\}$ allows some particles to escape the obstacle region. It is possible to insert additional obstacles to encode an arbitrary permutation when run in reverse, at a cost of $2N$ additional obstacles and requiring an area in worst case $3N \times 3N$ rather than $N \times 2N$. An example is shown in Fig. 9.

5 Conclusions

In this paper we analyzed the complexity of steering many particles with uniform inputs in a 2D environment with obstacles. We are motivated by practical challenges in steering magnetically-actuated particles through vascular networks. Many examples of natural, locally 2D vascular networks exist, e.g. the leaf example in Fig. 1b, and endothelial networks on the surface of organs.

Clearly, there are may exciting new challenges that lie ahead. The next step is to extend the complexity analysis to PSPACE-complete. We can construct AND and OR gates. Using dual-rail logic we can also implement NOT, NAND and NOR gates. Generating fan-out gates seems to require additional complexity in our BlockWorld construction. Some way of encoding an order of precedence so that a reversible operation on particle a will affect particle b is needed. Potential approaches use either 2×1 particles, or 0.5×1 obstacles so that the presence of a first particle can enable an action on a second particle, and yet be distinguished from the absence of the first particle and the presence of the second. With uniform 1×1 obstacles and particles, these cases are indistinguishable. Finally, platforms that can navigate in three dimensions pose a large number of additional challenges.

Acknowledgments. We acknowledge the helpful discussion and motivating experimental efforts with *T. pyriformis* cells by Yan Ou and Agung Julius at RPI and Paul Kim and MinJun Kim at Drexel University.

References

[1] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, J. Thomas F. Knight, R. Nagpal, E. Rauch, G. J. Sussman, and R. Weiss. Amorphous computing. *Communications of the ACM*, 43(5):74–82, 2000.

[2] H. Abelson, J. Beal, and G. J. Sussman. Amorphous computing. Technical Report MIT-CSAIL-TR-2007-030, MIT, MA 02139 USA, June 2007.

[3] S. Akella, W. H. Huang, K. M. Lynch, and M. T. Mason. Parts feeding on a conveyor with a one joint robot. *Algorithmica*, 26(3):313–344, 2000.

[4] S. Akella and M. T. Mason. Using partial sensor information to orient parts. *The International Journal of Robotics Research*, 18(10):963–997, 1999.

[5] A. Becker and T. Bretl. Approximate steering of a unicycle under bounded model perturbation using ensemble control. *IEEE Trans. Robot.*, 28(3):580–591, June 2012.

[6] 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 Int. Rob. and Sys.*, Oct. 2013 (accepted).

[7] A. Becker, C. Onyuksel, and T. Bretl. Feedback control of many differential-drive robots with uniform control inputs. In *IEEE Int. Rob. and Sys.*, Oct. 2012.

[8] A. Becker, Y. Ou, and A. Julius. Feedback control of many magnetized tetrahymena pyriformis cells by exploiting phase inhomogeneity. In *IEEE Int. Rob. and Sys.*, 2013 (accepted).

[9] 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. *Magn Reson Med*, 59(6):1287–1297, June 2008.

[10] P.-T. Chiang, J. Mielke, J. Godoy, J. M. Guerrero, L. B. Alemany, C. J. Villagómez, A. Saywell, L. Grill, and J. M. Tour. Toward a light-driven motorized nanocar: Synthesis and initial imaging of single molecules. *ACS Nano*, 6(1):592–597, Feb. 2011.

[11] M. Deléglise and J.-L. Nicolas. Maximal product of primes whose sum is bounded. *ArXiv e-prints*, July 2012.

[12] E. D. Demaine, M. L. Demaine, and J. O'Rourke. Pushpush and push-1 are np-hard in 2d. In *Proceedings of the 12th Annual Canadian Conference on Computational Geometry (CCCG),*, pages 211–219, Aug. 2000.

[13] E. D. Demaine and R. A. Hearn. Playing games with algorithms: Algorithmic combinatorial game theory. In M. H. Albert and R. J. Nowakowski, editors, *Games of No Chance 3*, volume 56, pages 3–56. Mathematical Sciences Research Institute Publications, Cambridge University Press, 2009.

[14] 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*, 32(2):218–246, 2013.

[15] D. Dor and U. Zwick. Sokoban and other motion planning problems. *Computational Geometry*, 13(4):215–228, 1999.

[16] B. Engels and T. Kamphans. On the complexity of randolph's robot game. Technical report, Rheinische Friedrich-Wilhelms-Universität Bonn Institut für Informatik I, University of Cologne, Germany, 2005.

[17] M. Erdmann and M. Mason. An exploration of sensorless manipulation. *IEEE J. Robot. Autom.*, 4(4):369–379, Aug. 1988.

[18] S. P. Fekete and A. Kröller. Geometry-based reasoning for a large sensor network. In *Proc. 22nd Annu. ACM Sympos. Comput. Geom.*, pages 475–476, 2006. Video available at http://compgeom.poly.edu/acmvideos/socg06video/index.html.

[19] S. P. Fekete and A. Kröller. Topology and routing in sensor networks. In *Proc. 3rd Intern. Workshop Algorithmic Aspects Wireless Sensor Networks (ALGOSENSORS)*, pages 6–15, 2007.

[20] S. P. Fekete, A. Kröller, D. Pfisterer, S. Fischer, and C. Buschmann. Neighborhood-based topology recognition in sensor networks. In *Proc. 1st Intern. Workshop Algorithmic Aspects Wireless Sensor Networks (ALGOSENSORS)*, volume 3121 of *Lecture Notes in Computer Science*, pages 123–136. Springer, 2004.

[21] S. Floyd, E. Diller, C. Pawashe, and M. Sitti. Control methodologies for a heterogeneous group of untethered magnetic micro-robots. *Int. J. Robot. Res.*, 30(13):1553–1565, Nov. 2011.

[22] D. Frutiger, B. Kratochvil, K. Vollmers, and B. J. Nelson. Magmites - wireless resonant magnetic microrobots. In *IEEE Int. Conf. Rob. Aut.*, Pasadena, CA, May 2008.

[23] O. C. Goemans, K. Goldberg, and A. F. van der Stappen. Blades: a new class of geometric primitives for feeding 3d parts on vibratory tracks. In *Int. Conf. Rob. Aut.*, pages 1730–1736, May 2006.

[24] K. Goldberg, B. V. Mirtich, Y. Zhuang, J. Craig, B. R. Carlisle, and J. Canny. Part pose statistics: estimators and experiments. *IEEE Trans. Robot. Autom.*, 15(5):849–857, Oct. 1999.

[25] K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10(2):201–225, 1993.

[26] R. A. Hearn and E. D. Demaine. Pspace-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *arXiv:cs/0205005*, cs.CC/0205005, 2002.

[27] M. Hoffmann. Motion planning amidst movable square blocks: Push-* is NP-hard. In *Canadian Conference on Computational Geometry*, pages 205–210, June 2000.

[28] M. Holzer and S. Schwoon. Assembling molecules in atomix is hard. *Theoretical Computer Science*, 313(3):447–462, 2 2004.

[29] J. Kahn, R. Katz, and K. Pister. Emerging challenges: Mobile networking for smart dust. *J. Comm. Networks*, pages 188–196, 2000.

[30] I. S. M. Khalil, M. P. Pichel, B. A. Reefman, O. S. Sukas, L. Abelmann, and S. Misra. Control of magnetotactic bacterium in a micro-fabricated maze. In *IEEE International Conference on Robotics and Automation*, pages 5488–5493, Karlsruhe, Germany, May 2013.

[31] A. Kröller, S. P. Fekete, D. Pfisterer, and S. Fischer. Deterministic boundary recognition and topology extraction for large sensor networks. In *Proc. 17th ACM-SIAM Sympos. Discrete Algorithms*, pages 1000–1009, 2006.

[32] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann, 1991.

[33] K. M. Lynch, M. Northrop, and P. Pan. Stable limit sets in a dynamic parts feeder. *IEEE Trans. Robot. Autom.*, 18(4):608–615, Aug. 2002.

[34] M. Moll and M. Erdmann. Manipulation of pose distributions. *The International Journal of Robotics Research*, 21(3):277–292, 2002.

[35] T. D. Murphey, J. Bernheisel, D. Choi, and K. M. Lynch. An example of parts handling and self-assembly using stable limit sets. In *Int. Conf. Int. Rob. Sys.*, pages 1624–1629, Aug. 2005.

[36] T. D. Murphey and J. W. Burdick. Feedback control methods for distributed manipulation systems that involve mechanical contacts. *The International Journal of Robotics Research*, 23(7-8):763–781, 2004.

[37] 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.*, 32(1):129–139, Jan. 2013.

[38] K. E. Peyer, L. Zhang, and B. J. Nelson. Bio-inspired magnetic swimming microrobots for biomedical applications. *Nanoscale*, 2013.

[39] M. Rubenstein, C. Ahler, and R. Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *IEEE Int. Conf. Rob. Aut.*, pages 3293–3298, May 2012.

[40] P. van Delft and J. Botermans. *Creative Puzzles of the World*, page 141. Cassell, 1978.

[41] A. van der Stappen, R.-P. Berretty, K. Goldberg, and M. Overmars. Geometry and part feeding. *Sensor Based Intelligent Robots*, pages 259–281, 2002.

[42] T. H. Vose, P. Umbanhowar, and K. M. Lynch. Friction-induced velocity fields for point parts sliding on a rigid oscillated plate. In *Robotics: Science and Systems*, Zurich, Switzerland, June 2008.

[43] G. Wilfong. Motion planning in the presence of movable obstacles. *Annals of Mathematics and Artificial Intelligence*, 3(1):131–150, 1991.