2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Particle Computation: Designing Worlds to Control Robot Swarms with only Global Signals

Aaron Becker Erik D. Demaine Sándor P. Fekete James McLurkin

***Abstract*— Micro- and nanorobots are often controlled by global input signals, such as an electromagnetic or gravitational field. These fields move each robot maximally until it hits a stationary obstacle or another stationary robot. This paper investigates 2D motion-planning complexity for large swarms of simple mobile robots (such as bacteria, sensors, or smart building material).**

In previous work we proved it is NP-hard to decide whether a given initial configuration can be transformed into a desired target configuration; in this paper we prove a stronger result: the problem of finding an optimal control sequence is PSPACE-complete. On the positive side, we show we can build useful systems by designing obstacles. We present a reconfigurable hardware platform and demonstrate how to form arbitrary permutations and build a compact absolute encoder. We then take the same platform and use *dual-rail logic* to build a universal logic gate that concurrently evaluates AND, NAND, NOR and OR operations. Using many of these gates and appropriate interconnects we can evaluate any logical expression.

I. INTRODUCTION

Milli-, micro-, and nanorobots are capable of entering environments too small for their larger cousins. Swarms of these tiny robots may be ideal for targeted drug delivery, on-site micro construction, and minimally invasive surgery. An untethered swarm could reach areas deep in the body that traditional, robots and tooling cannot. These swarms are often controlled by an external, global electromagnetic field [1]–[3]. Motion planning for large robotic populations actuated by the same field in a tortuous environment is difficult.

We investigate the following basic problem: *Given a map of an environment, such as the vascular network shown in Fig. 1, along with initial and goal positions for each robot, does there exist a sequence of inputs that will bring each robot to its goal position?* In previous work [4], it was shown that this problem is at least NP-hard, by reduction to a 3SAT problem. In this paper we improve the analysis and show the problem is PSPACE-complete. This complexity result has some benefits: we show that we can design artificial environments capable of computation, and describe configurations of obstacles that result in useful robotic systems: absolute encoders, Boolean logic as shown in Fig. 2, and planar displays.

Department of Computer Science, Rice University, Houston, TX 77005, aabecker@gmail.com, 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

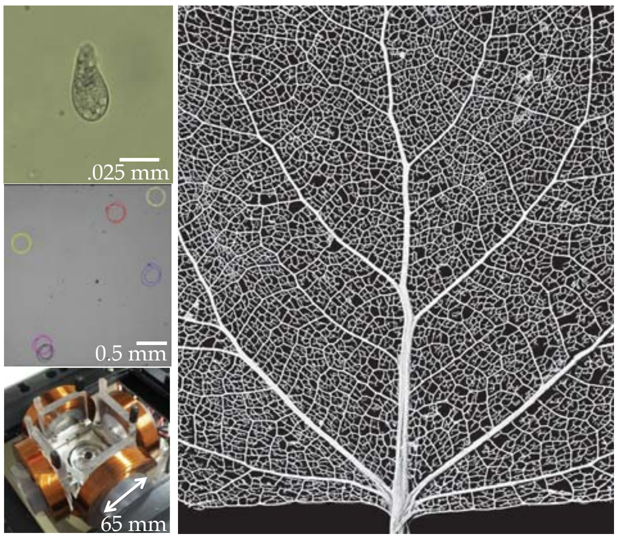


Fig. 1. (Left) State of the art in controlling small objects by force fields: after feeding iron particles to *T. pyriformis* cells and magnetizing the particles with a permanent magnet, the cells are mobile robots that can be turned by changing the orientation of an external magnetic field [5]. All cells are steered by the same global field. (Right) A complex vascular network, forming a typical environment for the parallel control of small robots. Given such a network along with initial and goal positions of N robots, is it possible to bring each robot to its goal position using a global control signal? (Right image credit: Royce Bair/Flikr/Getty Images)

We study this problem on a two-dimensional grid. We assume that robots cannot be individually controlled, but are all simultaneously given a message to travel in a given direction until they collide with an obstacle or another robot. 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 [6]–[9], except that all robots receive the same control inputs.

A. Problem Definition

More precisely, we consider the following scenario, which we call GLOBALCONTROL-MANYROBOTS:

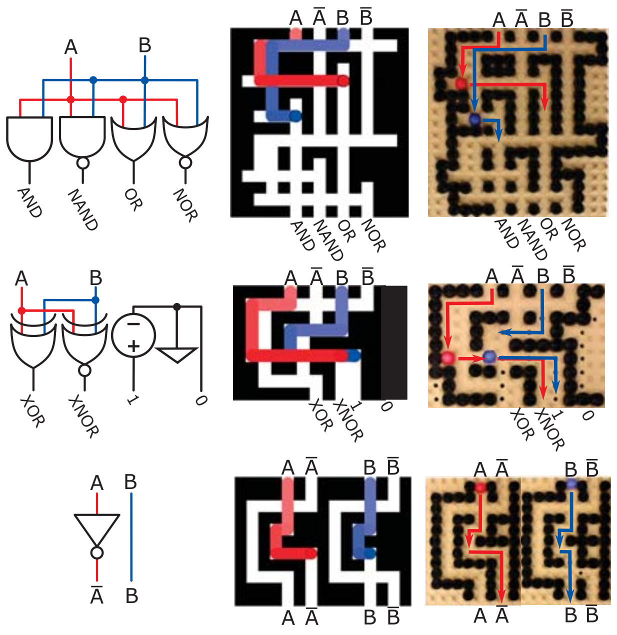


Fig. 2. Schematic, diagram, and physical implementation of dual-rail logic gates. Each gate employs the same clock sequence $\langle d, l, d, r\rangle$, the four inputs correspond to $A, \bar{A}, B, \bar{B}$, and the inputs are [1,1]. The top row is a universal logic gate whose four outputs are AND, NAND, OR, NOR. With input [1,1] the AND and OR outputs are set high. The middle row gate outputs the XOR, XNOR of the inputs and constants 1 and 0. The bottom row is a NOT gate and a connector. See the attached video at `http://youtu.be/mJWl-Pgfos0` for a hardware demonstration.

1) Initially, the planar square grid is filled with some unit-square robots (each occupying one cell of the grid) and some fixed unit-square blocks.
2) All robots are commanded in unison: the valid commands are "Go Up" (u), "Go Right" (r), "Go Down" (d), or "Go Left" (l). The robots all move in the commanded direction until they hit an obstacle or another robot. A representative command sequence is $\langle u, r, d, l, d, r, u, \ldots\rangle$. We call these global commands *force-field moves*. We assume we know the maximum dimension of the workspace and issue each command long enough for the robots to reach their maximum extent.
3) The goal is to get each robot to its specified position.

The algorithmic decision problem GLOBALCONTROL-MANYROBOTS is to decide whether a given configuration is solvable. This problem is computationally difficult: we prove PSPACE-completeness in Section III. 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.

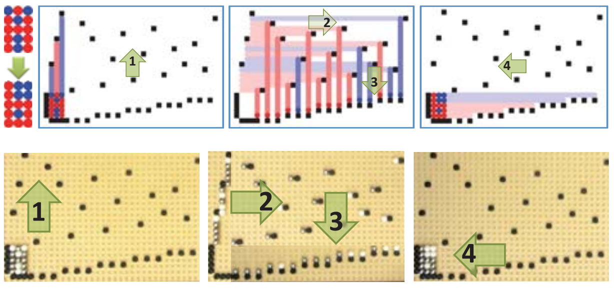


Fig. 3. (Top) Matrix permutation for N=15. Black cells are obstacles, white cells are free, and colored discs are individual robots. The world has been designed to permute the robots between 'A' into 'b' every four steps: $\langle u, r, d, l\rangle$. (1) The staggered obstacles on the left spread the 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 (4). (bottom) Hardware demonstration of a reconfigurable, gravity-fed manipulator that can rearrange (permute) arrays of colored spheres. The demonstration converts 'A' to 'b', but can be reprogrammed by switching the black stoppers to enable any array permutation. See video at `http://youtu.be/mJWl-Pgfos0`.

This makes developing algorithmic tools that enable global control by uniform commands important. In Sections II and IV, 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 robot swarm. See [10] for extended version.

II. MATRIX PERMUTATIONS

This section investigates a construction problem. Given the GLOBALCONTROL-MANYROBOTS constraints in I-A, 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.

A. Designing Workspace for a Single Permutation

A *matrix* is a 2D array of robots (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 *matrix permutation* assigns each element in A a unique position in B. Figs. 3 and 4 show constructions that execute matrix permutations of size $N = 15$ and 100, respectively. For simplicity of exposition, we assume henceforth that all matrices are $n \times n$ squares.

Theorem 1: Any matrix permutation can be executed by a set of obstacles that transforms matrix A into matrix B in just four moves. For N robots, the arrangement requires $(3N+1)^2$ space, $4N+1$ obstacles, and $10N/v$ time, where v is robot speed in units/s.

Proof: Refer to Figure 3 for an example. MATLAB code implementing this is available at [11]. The move sequence is $\langle u, r, d, l \rangle$. We identify the bottom left workspace square as (0,0), place the bottom-left robot at (1,1), and label the starting configuration A from 1 to N bottom-to-top, left-to-right. We also assign these indices to the corresponding entries in B.

(Move 1) for $i = 1$ to n, place an obstacle at $(i, 1 + n \cdot (i + 1))$: We place n obstacles, one for each column, spaced vertically n units apart, such that moving u spreads the robot array into a staggered vertical line. Each robot now has its own row, and are arranged index 1 to N from bottom to top.

(Move 2) for $i = 1$ to N, let $[b_r, b_c]$ be the row and column for robot i in B. Place an obstacle at $(2(n \cdot b_r + b_c) - (n+1), n + 2i)$: We place N obstacles to stop each robot during the move r. Each robot 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) for $i = 1$ to N, place an obstacle at $(n + 2i - 1, \lfloor \frac{i\;1}{n} \rfloor)$:. Moving d arranges the robots into their desired rows. These rows are spread in a staggered horizontal line.

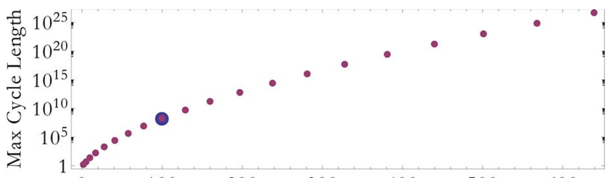
(Move 4) for $i = 1$ to n, place an obstacle at $(0, i)$: Moving l stacks the staggered rows into the desired permutation, and returns the array to the initial position. ∎

By reapplying the same permutation enough times, we can return to the original configuration. The permutation shown in Fig. 3 returns to the original image in 2 cycles. For a two-color image, we can always construct a permutation that resets in 2 cycles. We construct an *involution*, a function that is its own inverse, using cycles of length two that transpose two robots. This technique does not extend to images with more than two colors.

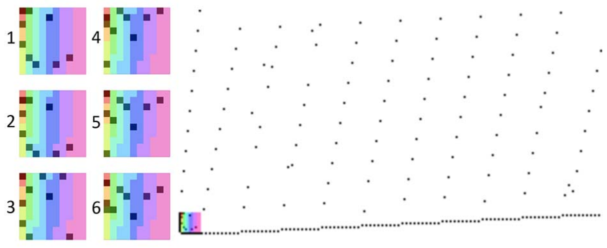
B. Physical Absolute Encoders and Animations

As shown in Fig. 3, 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. In an *absolute encoder* the current arrangement of robots serves as a unique representation of how many rotations have taken place. These applications exploit 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 robots through one permutation.

The cycle length is the least common multiple of the permutation cycles in the transformation $A \mapsto B$. Given N robots, 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 $\mathrm{LCM}(n_1, n_2, \ldots, n_k)$.



(a) Absolute encoder cycles



(b) Example encoder

Fig. 4. (a) Using a permutation gadget as an *absolute encoder*. Cycle length increases rapidly as the number of robots increases, and the current arrangement of robots uniquely represents how many rotations have taken place. (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. The first 6 permutations are shown at left with each cycle a different color.

This cycle length grows rapidly. For instance, using $N = 100$ robots, we can partition the robots into cycles of length {2, 3, 5, 7, 11, 13, 17, 19, 23}, see Fig. 4b. The LCM is 223,092,870. See [12] for a more in-depth look at the growth of the maximum cycle length as a function of N.

1) 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-robot arrangements □□■■■, ■□■□■, ■■□■□. Though permutations between any two exist, there is no single permutation that can generate all three. In fact, no single permutation can generate all possible permutations of the given robots. For the example in Fig. 4b, with 100 robots, 9 painted black and the rest white, the maximum cycle length we can generate is of length $\approx 2 \times 10^8$, 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 example above, there are $100!/(9!91!) \approx 2\times10^{12}$ permutations possible.

2) Reversible Permutations: The permutations generators shown in Fig. 4b are one-way devices. Attempting

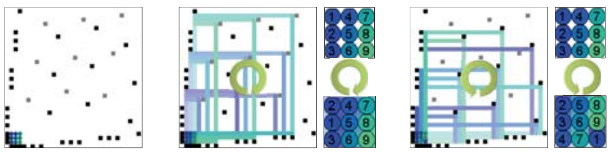


Fig. 5. The obstacles above generate the base permutation $p = (1, 2)$ in the CW direction $\langle u, r, d, l\rangle$ and $q = (1, 2, \cdots N)$ in the CCW direction $\langle r, u, l, d\rangle$. These can be applied repeatedly to BUBBLE SORT the matrix and generate any desired permutation.

to drive them in reverse $\langle l, d, r, u\rangle$ allows some robots 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. 5. Here, we encode the base permutation $p = (1, 2)$ in the CW direction $\langle u, r, d, l\rangle$ and $q = (1, 2, \cdots N)$ in the CCW direction $\langle r, u, l, d\rangle$. Repeated application of these two base permutations can generate any permutation, when used in a manner similar to BUBBLE SORT.

C. Designing a Workspace for Arbitrary Permutations

There are various ways in which we can exploit Theorem 1 in order to generate larger sets of (or even all) possible permutations. There is a tradeoff between the number of introduced obstacles and the number of moves required for realizing a permutation. We quote these theorems from [4], as they will be used in our PSPACE-proof. We start with obstacle sets that require only a few moves.

Lemma 2: Any permutation of N objects can be generated by the two base permutations $p = (1, 2)$ and $q = (1, 2, \cdots N)$. Moreover, any permutation can be generated by a sequence of length at most N^2 that consists of p and q.

Proof: See Fig. 5. 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 3: 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.

III. COMPLEXITY

In previous work [4], we showed that the problem GLOBALCONTROL-MANYROBOTS is computationally intractable in a particular sense: given an initial configuration of movable robots and fixed obstacles, it is NP-hard to decide whether any robot can be moved to a specified location. It was left as an important open problem whether an even stronger hardness result applies. In the following, we resolve this problem by proving PSPACE-completeness.

Theorem 4: GLOBALCONTROL-MANYROBOTS is PSPACE-complete: given an initial configuration of (labeled) movable robots and fixed obstacles, it is PSPACE-complete to compute a shortest sequence of force-field moves to achieve another (labeled) configuration.

Proof: The proof is largely based on a complexity result by Jerrum [13], who considered the following problem: Given a permutation group, specified by a set of generators, and a single target permutation π which is a member of the group, what is the shortest expression for the target permutation in terms of the generator? This problem was shown in [13] to be PSPACE-complete, even when the generator set consists of only two permutations, say, π_1 and π_2.

As shown in the previous Section III, we can realize any matrix permutation π_i of a square arrangement of robots by a set of obstacles, such that this permutation π_i is carried out by a quadruple of force-field moves. We can combine the sets of obstacles for the two different permutations π_1 and π_2, such that π_1 is realized by going through a clockwise sequence $\langle u, r, d, l\rangle$, while π_2 is realized by a counterclockwise sequence $\langle r, u, l, d\rangle$. We now argue that a target permutation π of the matrix can be realized by a minimum-length sequence of m force-field moves, if and only if π can be decomposed into a sequence of a total of n applications of permutations π_1 and π_2, where $m = 4n$.

The "if" part is easy: simply carry out the sequence of n permutations, each realized by a (clockwise or counterclockwise) quadruple of force-field moves. For the "only if" part, suppose we have a shortest sequence of m force-field moves to achieve permutation π, and consider an arbitrary subsequence that starts from the base position in which the robots form a square arrangement in the lower left-hand corner. It is easy to see that a minimum-length sequence cannot contain two consecutive moves that are both horizontal or both vertical: these moves would have to be be in opposite directions, and we could shorten the sequence by omitting the first move. Furthermore, by construction of the obstacle set, the first move must be u or r. Now it is easy to check that the choice of the first move determines the next three ones: u must be followed by $\langle r, d, l\rangle$; similarly, r must be followed by $\langle u, l, d\rangle$. Any other choice for moves 2–4 would produce a longer overall sequence, or destroy the matrix by leading to an arrangement from which no recovery to a square matrix is possible. Therefore, the overall sequence can be decomposed into

$m = 4n$ clockwise or counterclockwise quadruples. As described, each of these quadruples represents either π_1 or π_2, so π can be decomposed into n applications of permutations π_1 and π_2. This completes the proof. ■

Note that the result also implies the existence of solutions of exponential length, which can occur with polynomial space. Binary counters are particular examples of such long sequences that are useful for many purposes.

IV. Particle Logic

In our previous work [4] we showed that with only fixed obstacles and robots that move maximally in response to an input, we can construct a variety of logic elements. These include variable gadgets that enable setting multiple copies of up to n variables to be true or false, m-input OR, and AND gates. Unfortunately, we cannot build NOT gates because our system of robots and obstacles is conservative—we cannot create a new robot at the output when no robot is supplied to the input. A NOT gate is necessary to construct a logically complete set of gates. To do this, we rely on a form of *dual-rail logic*, where both the state and inverse (A and $\bar{A}$) of each signal are propagated throughout the computation. Dual-rail logic is often used in low-power electronics to increase the signal to noise ratio without increasing the voltage [14]. With dual-rail logic we can now construct the missing NOT gate, as shown in Fig. 2 (bottom). The command sequence $\langle d, l, d, r\rangle$ inverts the input. By adding one-way valves we can ignore any superfluous commands. Note that regardless of the command sequence, all robots arrive at their output ports at exactly the same time.

We now revisit the OR and AND gates of [4] using dual-rail logic and the four inputs $A, \bar{A}, B, \bar{B}$. Surprisingly, with the gate in Fig. 2 (top) we can simultaneously compute AND, NAND, OR and NOR. using the same command sequence $\langle d, l, d, r\rangle$ as the NOT gate. Outputs can be piped for further logic using the interconnections in Fig 2 (bottom). Unused outputs can be piped into a storage area and recycled for later use.

These gates are reminiscent of the Fredkin gate, a three-bit gate that swaps the last two bits if the first bit is 1 [15]. They are conservative, in that the number of input and output 1's and 0's are unchanged. They also form a universal set. Unlike the Fredkin gate, our gate is kinematic rather than dynamic, making it robust to noise and self-synchronizing – at the end of every move the robots are in a known state, and will not move until we apply another input. However, unlike the Fredkin gate, our AND/NAND/OR/NOR gate is not reversible.

Dual-rail devices open up new opportunities, including XOR and XNOR gates, which are not conservative using single-rail logic. This gate, shown in Fig. 2 also outputs a constant 1 and 0.

With an AND and XOR we can compactly construct a half-adder. We are hindered by an inability to construct a fan-out device that produces multiple copies of an input. Instead, we must take any logical expression and create multiple copies of each input. For example, a half-adder requires only one XOR and one AND gate, but our particle computation requires two A and two B inputs.

V. Hardware Demonstrations

Fig. 6 shows our scale prototype of a reconfigurable GlobalControl-ManyRobots environment, using 1.27 diam steel and nylon bearings as our robots and a naturally-occuring gravity field as the control field. The prototype is a 61×61 cm square sheet of 2 cm thick medium-density fiberboard (MDF), with a lattice grid of hemispherical-profile, 1.27 cm grooves milled at 1.27 cm spacing in the x and y directions. At the intersection of each set of orthogonal grooves is a 4 mm diameter hole. We can then insert plastic-headed thumb screws with 1.27 cm diam heads (McMaster #91185A444) to serve as obstacles. The prototype is centered and glued on top of a 20×20 cm square section of MDF. Pushing down on any top board edge tilts the entire prototype u, r, l, or d, and the bearings roll until they hit an obstacle or another bearing. The companion video illustrates this prototype configured to create a permutation that converts 'A' to 'b' under the command sequence $\langle u, r, d, l\rangle$, also shown in Fig. 3.

We have also configured the prototype to generate the dual-rail universal Boolean gate in Fig. 2, see the accompanying video. The long open paths in the permutation arrangement often lead to errors when bearings pop off their proper paths. The enclosed mazes of the logic gates are more reliable and we have not recorded any errors.

VI. Conclusions

We analyzed the problem of steering many robots with uniform inputs in a 2D environment containing obstacles. We introduced dual-rail particle logic computation, and designed environments that can efficiently perform matrix operations on groups of robots in parallel—our matrix permutation requires only four moves for any number of robots. These matrix operations enabled us to prove the general motion planning problem PSPACE-complete.

There remain many interesting problems to solve. We are motivated by practical challenges in steering micro-robots through vascular networks, which are common in

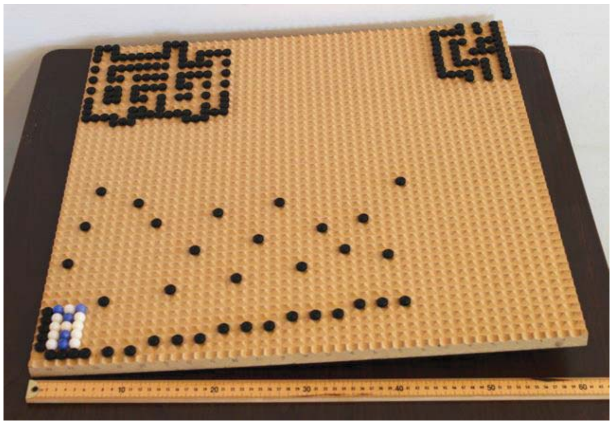


Fig. 6. Gravity-fed hardware implementation of GLOBALCONTROL-MANYROBOTS. Bottom left is a matrix permutation for changing 'A' to 'b', top left is a combination AND, NAND, OR, NOR gate, and top right is a NOT gate. See `http://youtu.be/mJWl-Pgfos0`.

biology. Though some are two-dimensional, including the leaf example in Fig. 1 and endothelial networks on the surface of organs, many of these networks are three dimensional. Magnetically actuated systems are capable of providing 3D control inputs, but control design poses additional challenges.

The paper investigated a subset of control in which all robots move maximally. Future work should investigate more general motion—what happens to our complexity proof if we can move all the robots a discrete distance, or along an arbitrary curve? We also abstracted important practical constraints e.g., ferromagnetic objects tend to clump in a magnetic field, and most magnetic fields are not perfectly uniform.

Using *dual-rail logic*, we are limited to conservative logic. We cannot create new robots, so logic such as a multi-bit adder require exponentially increasing numbers of inputs. Generating fan-out gates seems to require additional flexibility in our problem definition, because conservation rules are violated. Some way of encoding an order of precedence is needed so that a reversible operation on robot a can affect robot b. Possible approaches use non-unit size components–either 2×1 robots, or 0.5×1 obstacles.

Finally, our research has potential applications in micro-construction and nano-assembly. These applications require additional theoretical analysis to model heterogeneous objects and objects that bond when forced together, e.g., MEMS components and molecular chains.

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. Ricardo Marquez and Artie Shen assisted with photography and the hardware platform. This work was supported by the National Science Foundation under CPS-1035716.

REFERENCES

[1] 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*, vol. 59, no. 6, pp. 1287–1297, Jun. 2008.

[2] 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*, Karlsruhe, Germany, May 2013, pp. 5488–5493.

[3] D. de Lanauze, O. Felfoul, J.-P. Turcot, M. Mohammadi, and S. Martel, "Three-dimensional remote aggregation and steering of magnetotactic bacteria microrobots for drug delivery applications," *The International Journal of Robotics Research*, 11 2013. [Online]. Available: http://ijr.sagepub.com/content/early/2013/11/11/0278364913500543

[4] 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, pp. 51–66. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-45346-5_5

[5] 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.

[6] 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),*, Aug. 2000, pp. 211–219.

[7] M. Hoffmann, "Motion planning amidst movable square blocks: Push-* is NP-hard," in *Canadian Conference on Computational Geometry*, Jun. 2000, pp. 205–210.

[8] 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*, vol. cs.CC/0205005, 2002. [Online]. Available: http://arxiv.org/abs/cs/0205005

[9] M. Holzer and S. Schwoon, "Assembling molecules in ATOMIX is hard," *Theoretical Computer Science*, vol. 313, no. 3, pp. 447–462, 2 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397503005930

[10] A. Becker, E. D. Demaine, S. P. Fekete, and J. McLurkin, "Particle computation: Designing worlds to control robot swarms with only global signals," *ArXiv e-prints*, Feb. 2014.

[11] A. Becker. (2014, Feb.) "Particle Computation: Permute an array of particles with 4 global moves." MATLAB Central File Exchange. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/45538

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

[13] M. R. Jerrum, "The complexity of finding minimum-length generator sequences," *Theoretical Computer Science*, vol. 36, pp. 265–289, 1985.

[14] R. Zimmermann and W. Fichtner, "Low-power logic styles: CMOS versus pass-transistor logic," *Solid-State Circuits, IEEE Journal of*, vol. 32, no. 7, pp. 1079–1090, 1997.

[15] E. Fredkin and T. Toffoli, "Conservative logic," *International Journal of Theoretical Physics*, vol. 21, no. 3-4, pp. 219–253, 1982. [Online]. Available: http://dx.doi.org/10.1007/BF01857727