Jingyuan Zhu, Aseem Salhotra, Christoph Robert Meinecke et al.

Solving the 3-Satisfiability Problem Using Network-Based Biocomputation

Erstveröffentlichung in / First published in:

Advanced intelligent systems. 2022. 4(12). Wiley. ISSN: 2640-4567.

DOI: https://doi.org/10.1002/aisy.202200202

RESEARCH ARTICLE

ADVANCED INTELLIGENT SYSTEMS Open Access

www.advintellsyst.com

Solving the 3-Satisfiability Problem Using Network-Based Biocomputation

Jingyuan Zhu, Aseem Salhotra, Christoph Robert Meinecke, Pradheebha Surendiran, Roman Lyttleton, Danny Reuter, Hillel Kugler, Stefan Diez, Alf Månsson, Heiner Linke,* and Till Korten**

The 3-satisfiability Problem (3-SAT) is a demanding combinatorial problem that is of central importance among the nondeterministic polynomial (NP) complete problems, with applications in circuit design, artificial intelligence, and logistics. Even with optimized algorithms, the solution space that needs to be explored grows exponentially with the increasing size of 3-SAT instances. Thus, large 3-SAT instances require excessive amounts of energy to solve with serial electronic computers. Network-based biocomputation (NBC) is a parallel computation approach with drastically reduced energy consumption. NBC uses biomolecular motors to propel cytoskeletal filaments through nanofabricated networks that encode mathematical problems. By stochastically exploring possible paths through the networks, the cytoskeletal filaments find possible solutions. However, to date, no NBC algorithm for 3-SAT has been available. Herein, an algorithm that converts 3-SAT into an NBC-compatible network format is reported and four small 3-SAT instances (with up to three variables and five clauses) using the actin–myosin biomolecular motor system are experimentally solved. Because practical polynomial conversions to 3-SAT exist for many important NP complete problems, the result opens the door to enable NBC to solve small instances of a wide range of problems.

1. Introduction

The satisfiability problem (SAT) is a combinatorial decision problem that evaluates Boolean (binary) logic (see **Textbox 1**). Solving SAT is essential for technologies such as symbolic model checking,[1] planning in artificial intelligence,[2] automated theorem proving,[3] and hardware verification.[4] A specific representation of SAT, namely, the 3-SAT problem, is also increasingly recognized to be of scientific importance in its own right. For example, a recent study showed all Ising models, of fundamental importance in magnetism and beyond,[5,6] to be equivalent to instances of 3-SAT.[7] From a mathematical perspective, 3-SAT is considered to be the benchmark nondeterministic polynomial-time complete (NP complete) problem.[8] In addition, this problem is of particular interest because polynomial conversions to SAT exist for a wide range of other important NP complete problems.[9] Therefore, the

J. Zhu, A. Salhotra, P. Surendiran, R. Lyttleton, A. Månsson, H. Linke
NanoLund
Lund University
Box 118, 22100 Lund, Sweden
E-mail: alf.mansson@lnu.se; heiner.linke@lth.lu.se

J. Zhu, P. Surendiran, R. Lyttleton, H. Linke
Division of Solid State Physics
Lund University
Box 118, 22100 Lund, Sweden

A. Salhotra, A. Månsson
Department of Chemistry and Biomedical Sciences
Linnaeus University
39182 Kalmar, Sweden

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202200202.

DOI: 10.1002/aisy.202200202

C. R. Meinecke, D. Reuter
Center for Microtechnologies
Technische Universität Chemnitz
09126 Chemnitz, Germany

D. Reuter
Department Nano Device Technologies
Fraunhofer Institute for Electronic Nano Systems ENAS
09126 Chemnitz, Germany

H. Kugler
Faculty of Engineering
Bar-Ilan University
Ramat Gan 5290002, Israel

S. Diez, T. Korten
B CUBE - Center for Molecular Bioengineering and Cluster of Excellence Physics of Life
Technische Universität Dresden
01307 Dresden, Germany
E-mail: till.korten@tu-dresden.de

S. Diez
Max Planck Institute of Molecular Cell Biology and Genetics
01307 Dresden, Germany

Textbox 1. The 3-Satisfiability Problem (3-SAT).

A propositional logic formula, also called Boolean expression, is built from variables, operators AND (conjunction, also denoted by ∧), OR (disjunction, ∨), NOT (negation, ¬), and parentheses. For example:

$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \quad (1)$$

A literal is either a variable, called positive literal (e.g., x_1), or the negation of a variable, called negative literal (e.g., $\neg x_1$).

A clause is a collection of literals enclosed in parentheses (e.g., $(x_1 \vee x_2 \vee x_3)$).

A formula is in conjunctive normal form (CNF) or clausal normal form if it is a conjunction of one or more clauses. In CNF, all clauses are disjunctions (OR). Any propositional logic formula can be represented in CNF form.

The Boolean satisfiability problem (SAT) is the problem of determining whether there exists an assignment of values to its variables that makes the formula Φ TRUE. The formula is said to be satisfiable then and unsatisfiable otherwise. The example given above has the solution $x_1 = \text{TRUE}$ and $x_2 = \text{TRUE}$.

In 3-SAT, formulas are in 3-CNF, in which each clause is limited to at most three literals. Any CNF formula can be converted to 3-CNF.

ability to solve SAT enables solving a large number of different NP complete problems.

However, 3-SAT is conceptually simple; it is computationally demanding to solve in practice. Even though extensive efforts have been expended to solve 3-SAT on serial transistor-based computers,[10,11] only algorithms with exponential worst-case complexity have been developed.[11] Therefore, the processing time and the energy cost of solving a 3-SAT instance grow exponentially with problem size. The demand for more computational power, and the need for reducing the energy used in computation, therefore necessitate the search for energy-efficient alternatives to serial, electronic computers.[12,13]

Network-based biocomputation (NBC) is an emerging approach that is expected to require orders of magnitude less energy than electronic computers.[14] So far, this approach has been used to solve a small instance of the NP-complete Subset Sum Problem (SSP),[14,15] and an optimized algorithm has been developed for solving Exact Cover with NBC.[16] In NBC, a given combinatorial problem is encoded into a graphical, modular network that, after implementation by nanofabrication, is explored in a parallel manner by biological agents.[15] This novel computation approach is an inherently stochastic method based on nondeterministic operation due to the random choice that agents make in the network. Similar to quantum computing, the output of NBC is a sample of the population of all possible solutions.[17] The result of NBC is therefore reported as the solution of the problem instance together with a statistical confidence interval.[14,15]

Here we take a key step in demonstrating the general applicability of NBC to small instances of NP complete problems. We first present a new network algorithm for encoding 3-SAT instances in a graphic, planar layout suitable for NBC. We then implement this algorithm to experimentally solve four small instances of 3-SAT by allowing myosin-driven actin filaments to explore the nanofabricated network that encodes the problem. Finally, using a newly developed statistical approach, we confirm that our solutions are significant on a 99% confidence level.

2. Results

2.1. 3-SAT Network Algorithm

3-SAT (Textbox 1) is a variant of SAT that is restricted to a Boolean formula in conjunctive normal form (CNF), where each clause is limited to at most three literals. In 3-SAT, one asks whether the formula is satisfiable by assigning appropriate logical values to its variables. In the present study, we map 3-SAT to a spatially encoded NBC network that solves the mathematical problem. The network encoding is a critical step in implementing NBC for a given NP complete problem. We encode 3-SAT into network format in five steps. Briefly (see below for a detailed explanation including examples): Step 1: Literal assignments are encoded as binary numbers reflecting specific clauses that are satisfied by the respective literals.[3,18] Step 2: The binary numbers are represented by unary encoding in a planar network. Step 3: The bitwise 'OR' operation is realized as an 'OR-block' in the unary network via several types of junctions. Step 4: The entire 3-SAT instance is represented in a network consisting of one 'OR-block' per variable. Step 5: The network is further optimized to reduce the physical size (Figure 2). Satisfiability of the instance is tested at the target exit representing TRUE assignment for all clauses via the readout of the number of exiting computing agents. Notably, the algorithm for network design (Steps 1–5) scales polynomially with network size and requires no prior information about the solution.

As a minimal example, we use the 3-SAT instance Φ_1 with two variables (x_1 and x_2) and four clauses: $(x_1 \vee x_2)$, $(\neg x_1 \vee x_2)$, $(x_1 \vee \neg x_2)$ and $(\neg x_1 \vee x_2)$ (Table 1)

$$\varphi_1 = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \quad (2)$$

Φ_1 was chosen such that exactly one assignment ($x_1 = \text{TRUE}, x_2 = \text{TRUE}$) satisfies the logical statement. In addition, we also solve three other instances, namely, Φ_2, (two variables, four clauses, no solution), Φ_3 (three variables, five clauses, one solution), and Φ_4 (three variables, five clauses, no solution).

$$\varphi_2 = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2) \quad (3)$$

$$\varphi_3 = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) \quad (4)$$

Table 1. Binary representations of instance Φ_1. Binary numbers formed by the logical values from all four clauses. Each variable (x_1 and x_2) will generate one pair of binary numbers when assigned TRUE or FALSE.

	Assignment	$(x_1 \vee x_2)$	$(\neg x_1 \vee x_2)$	$(x_1 \vee \neg x_2)$	$(\neg x_1 \vee x_2)$	Binary
x_1	TRUE	fulfilled	Not fulfilled	fulfilled	Not fulfilled	1010_2
	FALSE	Not fulfilled	Fulfilled	Not fulfilled	fulfilled	0101_2
x_2	TRUE	fulfilled	Fulfilled	Not fulfilled	fulfilled	1101_2
	FALSE	Not fulfilled	Not fulfilled	fulfilled	Not fulfilled	0010_2

$$\varphi_4=(x_1\vee x_2)\wedge(\neg x_1\vee x_2)\wedge(x_1\vee\neg x_2)\wedge(\neg x_1\vee\neg x_2)\wedge(x_1\vee x_2\vee x_3) \quad (5)$$

Technically, Φ_1 and Φ_2 are instances not only of 3-SAT but also of 2-SAT, which is not NP-complete. However, we deliberately chose these very simple instances in order to facilitate understanding how the algorithm works. We use Φ_1 as an example to explicitly walk through the five steps of the network encoding algorithm:

2.1.1. Step 1: Binary Representation of 3-SAT Instance Φ_1

In the CNF representation of 3-SAT, any variable assignment will either fulfil a clause (the result of the clause is TRUE) or not. Thus, an instance with c clauses and n variables can be represented by n pairs of c-digit binary numbers: each digit of the binary numbers represents a clause and each pair of binary numbers represents the two possible assignments for each variable.[3] The digit is binary 1_2 if the variable assignment fulfils the clause and 0_2 otherwise (the 2 in subscript indicates the binary base). For our minimal example Φ_1, the TRUE assignment of x_1 (first row in Table 1) fulfils the first clause $(x_1 \vee x_2)$as well as the third clause $(x_1 \vee \neg x_2)$, while the second $(\neg x_1 \vee x_2)$ and fourth $(\neg x_1 \vee x_2)$ clauses are not fulfilled. Therefore, the respective binary representation is 1010_2. We give the complete binary representation for our minimal example Φ_1 in Table 1.

The whole 3-SAT instance is satisfiable if a combination with exactly one binary number from each pair can be found, such that the bitwise 'OR' operation for the combination returns all ones, indicating that each clause has been fulfilled. Table 2 lists all possible combinations of binary representations for Φ_1. The combination $(1010_2 \wedge 1101_2 = 1111_2)$ corresponding to $x_1 = \text{TRUE}, x_2 = \text{TRUE}$ satisfies Φ_1, indicating that this instance is satisfiable.

Table 2. Truth table for the instance formula Φ_1. The binary representation of both variables (x_1 and x_2) and bitwise 'OR' operation result of all the possible combinations of variable assignments. The assignments $x_1 = \text{TRUE}$ and $x_2 = \text{TRUE}$ represent the solution for Φ_1.

x_1	x_2	Binary x_1	Binary x_2	Bitwise 'OR' of Binary x_1 and Binary x_2	Output
TRUE	FALSE	1010_2	0010_2	1110_2	FALSE
FALSE	TRUE	0101_2	1101_2	1101_2	FALSE
TRUE	TRUE	1010_2	1101_2	1111_2	TRUE
FALSE	FALSE	0101_2	0010_2	0111_2	FALSE

2.1.2. Step 2: Encoding of Numbers in the Network

In our spatial network encoding method, information is processed in a unary numeral system where a symbol (in our case the network column) representing 1 is repeated N-times to represent number N. In the network, we define the leftmost column as column '0,' so any agent in that column will represent the number '0'. Moving to the right, the number for each column increases in increments of 1. For example, in Figure 1a, from left to right, each column represents a number from $0_{10} = 0000_2$ to $7_{10} = 0111_2$. An agent travelling in a given column thus represents the corresponding number. An example agent travelling along column $2_{10} = 0010_2$ is shown as an orange dashed line in Figure 1a.

2.1.3. Step 3: Bitwise OR Operation in Unary Encoding

Operations within the network are performed by connecting an input (entrance) row (at the top in Figure 1a,b) to the respective output (exit) row (near the bottom in Figure 1a,b). The network consists of a grid of individual junctions, where different types of junctions allow agents to travel either straight down—not changing the input—or diagonally down—changing the input by one for each row they travel (see Supporting Information for details). Thus, the number of network rows (and columns) required for each operation is the largest possible difference between input and output. By repeating the junctions at each column, the network block always performs the same operation with the same operand for a certain range of input columns.

Figure 1b shows an illustrative example network that performs an OR operation with operand 0101_2 for all inputs from 0000_2 to 1000_2. The number of rows in the network equals the value of the operand. Each input is connected to the respective output by first moving the respective number of rows straight down and then moving the respective number of rows diagonally down. Within the network, each input consists either of a *reset-FALSE junction* (see Table S2, Supporting Information for the junction layout) that ensures that agents move straight down (in case the input already contains the bit of the operand) or a *reset-TRUE junction* (see Table S2, Supporting Information for the junction layout) that forces agents to take the diagonal path and move to the right. All other junctions are *pass junctions* designed to ensure that agents stay on their respective paths (either straight down or diagonal). Placement of the *reset-TRUE junctions* is determined by checking if the binary number corresponding to the respective column and the operand have common '1' bits. In that case, the *reset-TRUE junction* is placed further down to move the agents fewer steps to the side and arrive at the correct result. For example, column 0000_2 has no common bits with the operand 0101_2 and thus the *reset-TRUE junction* is placed immediately at the entrance of the OR network block, guiding the agent to the correct exit at column 0101_2 ($0000_2 \vee 0101_2 = 0101_2$; purple path 1 in Figure 1b). In contrast, column 0110_2 has the second bit in common with the operand 0101_2. Therefore, the *reset-TRUE junction* is placed

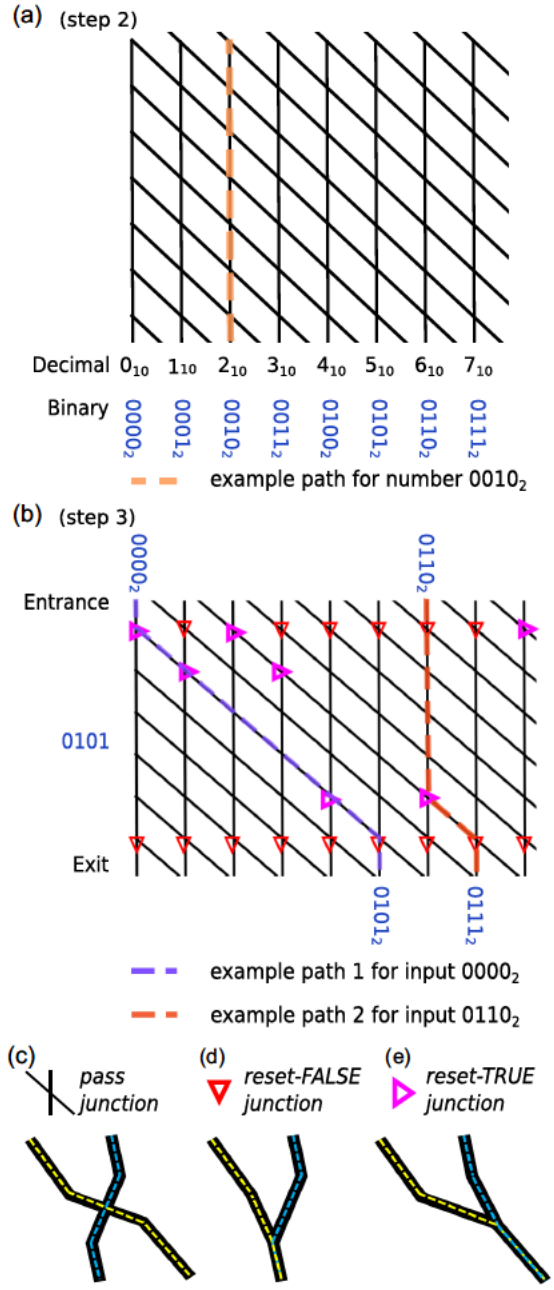

Figure 1. Network encoding examples. a) Example network shows numbers represented by columns ranging from 0000_2 to 0111_2. b) Example network encoding bitwise 'OR' with the operand 0101_2 for all inputs between 0000_2 and 1000_2. Path 1 shows the agent entering the network at column 0000_2 where a *reset-TRUE junction* is introduced; no shift is required. Path 2 shows the agent entering the network at column 0110_2, where there is a common '1' bit with operand 0101_2 at the second-most significant digit. The *reset-TRUE junction* is shifted by four (2^2) rows down. c–e) Schematics of *pass junction*, *reset-FALSE junction*, and *reset-TRUE junction*, in which the yellow and blue dashed lines represent the paths of filaments starting from left and right entrances, respectively.

0100_2 rows further down, guiding the agent to the correct exit at column 0111_2 ($0110_2 \vee 0101_2 = 0111_2$ red path 2 in Figure 1b).

2.1.4. Step 4: Network Representation of 3-SAT

In the binary representation of 3-SAT, each variable is represented by two binary numbers (described in Step 1). In order to enable the agents to randomly explore all possible variable assignments, each agent chooses randomly between either of these two numbers and performs the bitwise OR operation for its current value with that number as an operand. This is achieved by encoding both numbers as operands into the same OR operation block in the network and then replacing the first *reset-TRUE junction* at each input with a junction that is designed such that input agents have a 50% chance to choose each variable assignment by moving either diagonally or straight down. If the first *reset-TRUE junction* is at the first row of the OR operation block, it is replaced by a *split junction*, that is designed to distribute agents arriving from the top or the left equally between both exits of the junction (see Table S2, Supporting Information for the junction layout). If the first *reset-TRUE junction* is at a lower row of the OR operation block, it is replaced by a *split-top junction*, that is designed to distribute agents arriving from the top equally between both junction exits but keep agents arriving from the left on the diagonal path (see Table S2, Supporting Information for the junction layout).

As an example, we show the network encoding of instance Φ_1 in Figure 2a. To illustrate the solving process, we have marked all the possible legal paths in the network for this instance (Figure 2a, green dashed lines). In this network, the variable x_2 is represented by the first OR block with the only input 0000_2 and the operands 1101_2 and 0010_2 (corresponding to the assignments $x_2 = \text{TRUE}$ and $x_2 = \text{FALSE}$, respectively). In this first OR block, agents have a 50% chance to choose either operand and arrive at output 1101_2 or output 0010_2 (green dashed lines in Figure 2a). That way, agents entering the network stochastically pick either the assignment $x_2 = \text{TRUE}$ (corresponding to the binary number 1101_2, encoding fulfilment of clauses $(x_1 \vee x_2)$, $(\neg x_1 \vee x_2)$, and $(\neg x_1 \vee x_2)$) or the assignment $x_2 = \text{FALSE}$ (corresponding to the binary number 0010_2, encoding fulfilment of clause $(x_1 \vee \neg x_2)$). The output of the first OR operation block in row 13 of the network acts as the input for the next 'OR' operation block that represents variable x_1. There, agents randomly pick the binary number operands 10 10_2 ($x_1 = \text{TRUE}$) or 0101_2 ($x_1 = \text{FALSE}$). Thus, agents that arrived at column 0010_2 will exit the second OR operation block at columns 0111_2 or 1010_2, while agents that arrived at column 1101_2 will exit at columns 1101_2 or 1111_2.

By observing column 1111_2, the satisfiability of the instance can be checked. If agents can find a way through the network and exit at column 1111_2, this indicates that there exists one combination for assignments to x_1 and x_2 that satisfies the formula. During the operation of a network device, the solution can be read out by following the legal path leading to column 1111_2. In the example given in Figure 2a, the solution is $x_1 = \text{TRUE}$ (1010_2) and $x_2 = \text{TRUE}$ (1101_2).

2.1.5. Step 5: Network Optimization

Finally, we optimized the network by reducing its physical size before nanofabrication. A smaller network is cheaper to fabricate

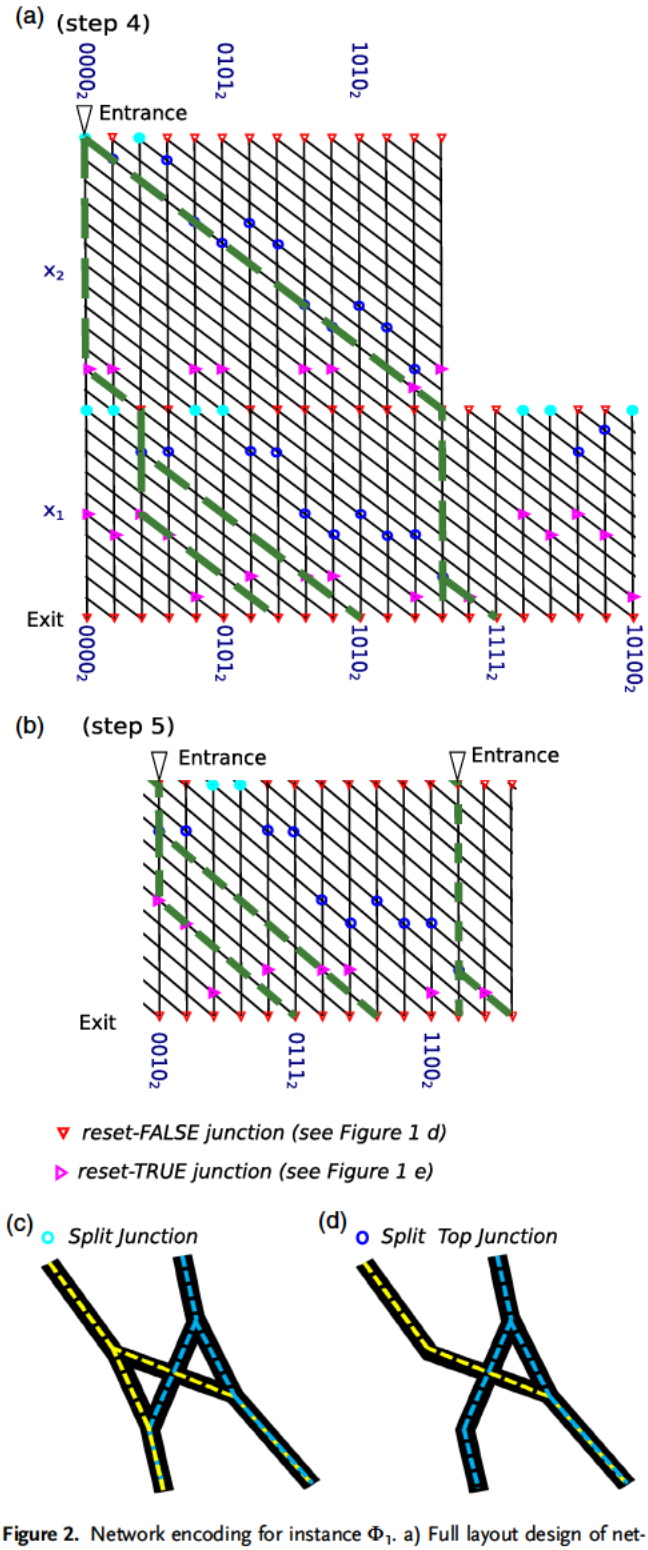

Figure 2. Network encoding for instance Φ_1. a) Full layout design of network encoding for instance Φ_1. The green paths are all the possible paths in the network, where agents can travel according to the network design. The corresponding detailed physical layout of the junctions is shown in Figure 3. b) The optimized spatially encoded network design for 3-SAT instance Φ_1 following step 5. c,d) Schematics of *split junction*, and *split top junction*, in which the yellow and blue dashed lines represent the paths of filaments starting from left and right entrances, respectively.

and can be explored by fewer agents in a shorter time. Specifically, we first replaced the single entrance at the top with two separate entrances, placed at the two columns that represent the two binary numbers for the first variable (see Figure 2a (entire network) compared with Figure 2b (optimized version of network in 2a). By doing this, we removed the variable space for the first variable, reducing the travelling distance for all agents. Moreover, with two entrances, more agents can enter the network at the same time, increasing operation speed. Second, we truncated the rows to the left of the first variable, because that part of the network is not explored by agents moving correctly according to the network design. Third, recognizing that we need to record the network output only at the one column that represents the all-one binary number (target exit, see step 4), we removed all columns to the right of that column. The optimized network encoding instance is significantly smaller than the original design (compare Figure 2a,b) but still explores all nontrivial, possible solutions, and its creation requires no prior knowledge of the solution. The network layout for all the instances can be found in Supporting Information S3–S6, Supporting Information.

3. Experimental Demonstration

In order to demonstrate that the network algorithm works in practice and to analyze the error rates of the 3-SAT networks, we fabricated four networks that encode small 3-SAT instances with up to three variables and five clauses. As biological agents, we used actin filaments that were propelled by myosin II motor fragments (heavy meromyosin) along the surface in gliding motility assays.[19–23] Actin filaments are highly flexible with a very low escape rate from nanochannels,[21] when propelled by fast myosin II molecular motors from skeletal muscle. With a speed of up to 15 $\mu m\,s^{-1}$, a high value among biomolecular motors,[24] this motor–filament combination enables fast computation.[20,21,24–30] Reliable guiding of agents through the network was enabled by two features: 1) physical guiding, provided by channels that are patterned into a layer of polymer to form a network, and 2i) chemical guiding, achieved by selectively functionalizing only the floor of the channels to enable actin–myosin motility. As a result of (1) and (2), the filaments can only attach to the channel floors where they are propelled by motors and move, guided by channel walls.[21] Recent improvements of the actomyosin function and longevity in nanofabricated networks[31] have been instrumental in facilitating the effective use of this motor–filament system for the present purpose.

The general layout of the networks implementing the algorithm explained earlier is shown in Figure 3a with details of the key junctional elements given in Figure 3b–f (see also Table S2, Supporting Information). The width of the network channels was 100 nm with a wall height of 400 nm. Two loading zones (rows of large heart shapes in Figure 3a) were placed to guide filaments to enter the network from columns representing numbers corresponding to one variable (here at column 2 and column 13, see Figure 3a,g). One row of rectifier structures was introduced at the bottom of the network to stop filaments from re-entering the network. One long outer channel was introduced between exits and loading zones to recirculate the filaments.

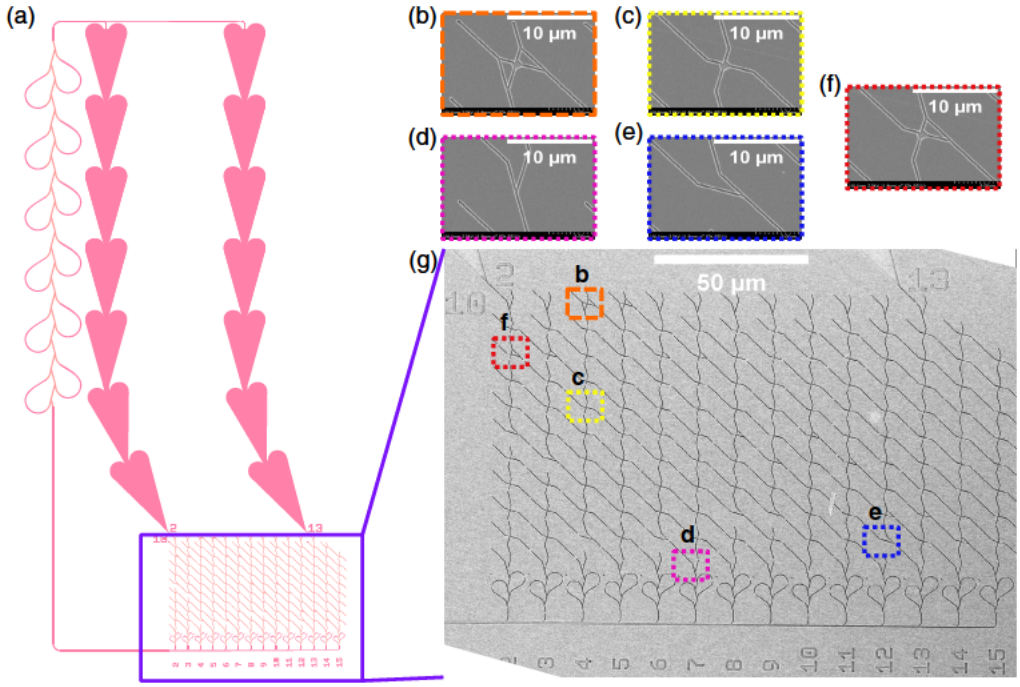

Figure 3. Physical network layout for solving instance Φ_1. a) Device layout. Schematic of the actual device used for experiments. Filaments enter the network via the loading zones at two entrances on the top. Filaments that exit the network at the bottom are collected via the recirculation path and guided to loading zones for multitime operations. b) Scanning electron microscopy (SEM) image of a *split junction*. c) SEM image of a *pass junction*. d) SEM image of a *reset-FALSE junction*. e) SEM image of a *reset-TRUE junction*. f) SEM image of a *split top junction*. g) SEM image of the overall network.

The devices were then operated using an in vitro motility assay,[19] where fluorescently labelled actin filaments from the solution were collected in loading zones from where they were guided by nonlabeled myosin-II motor fragments to explore the network. To read out the results, fluorescence timelapse movies of the process were recorded (see details in the Experimental Section).

The data in **Figure 4**a, based on such recordings, show the standard deviation projection of the resulting time series. In this projection, more frequently travelled paths appear brighter. Each straight part of the paths at the bottom represents one exit. Quantitatively, the distribution of the filaments leaving each exit was obtained by counting the filaments in image sequences where each filament was tracked (blue bars, Figure 4b). Similarly, the error rate at *pass junctions* was determined to be $2.20\% \pm 0.09\%$ by counting the fraction of agents that take a turn at *pass junctions* (567) divided by the total number of *pass junction* crossing events (25 748).

In our algorithm, the satisfiability of the formula is determined by whether there are filaments that leave the network at the target exit, that is, at column 15 corresponding to 1111_2 for problems Φ_1 (Figure 2a) and Φ_2, or column 31 corresponding to 11111_2 for problem Φ_3 and Φ_4, respectively. However, in a real device, filaments may inadvertently turn incorrectly at *pass junctions* and exit at incorrect columns (not following the rules of the networks), thus creating a background of errors in the filament distribution histogram. Therefore, we have to be able to distinguish correct results for a given exit, where filaments arrive while following the rules of the junctions in the network, and incorrect results, where filaments arrive at a given exit due to wrong turns.

3.1. Quantifying the Significance of Computation using Statistical Analysis

NBC is inherently stochastic and thus delivers answers with a confidence level lower than 1 (100%). Calculating the confidence level is thus an integral part of solving a problem instance. To quantitatively assess the computation result from NBC, we developed a statistical method to estimate the probability that the results measured for each network indicate that the corresponding problem instance is satisfiable. For the networks encoding 3-SAT instances, this is equivalent to estimating the probability that the target exit corresponds to a correct result.

The statistical analysis is described in detail in Supporting Information (section "Statistical evaluation of the experimental results"). Briefly, we begin by estimating the fraction of useful agents (those that do not make wrong turns at any *pass junction*) from the average number of *pass junctions* that agents will travel through and the fractional error per *pass junctions* (f, see the Supplementary Materials and Methods). We then calculated the expected number of agents that follow a legal path and an illegal path using the normal approximation of a binomial distribution and arrived at a z-score-based calculation of a significance level that also corresponds to a certain confidence. Comparing the experimental readout to the expected values (Figure 4b) resulting from our analysis, instances Φ_1 and Φ_3 are satisfied

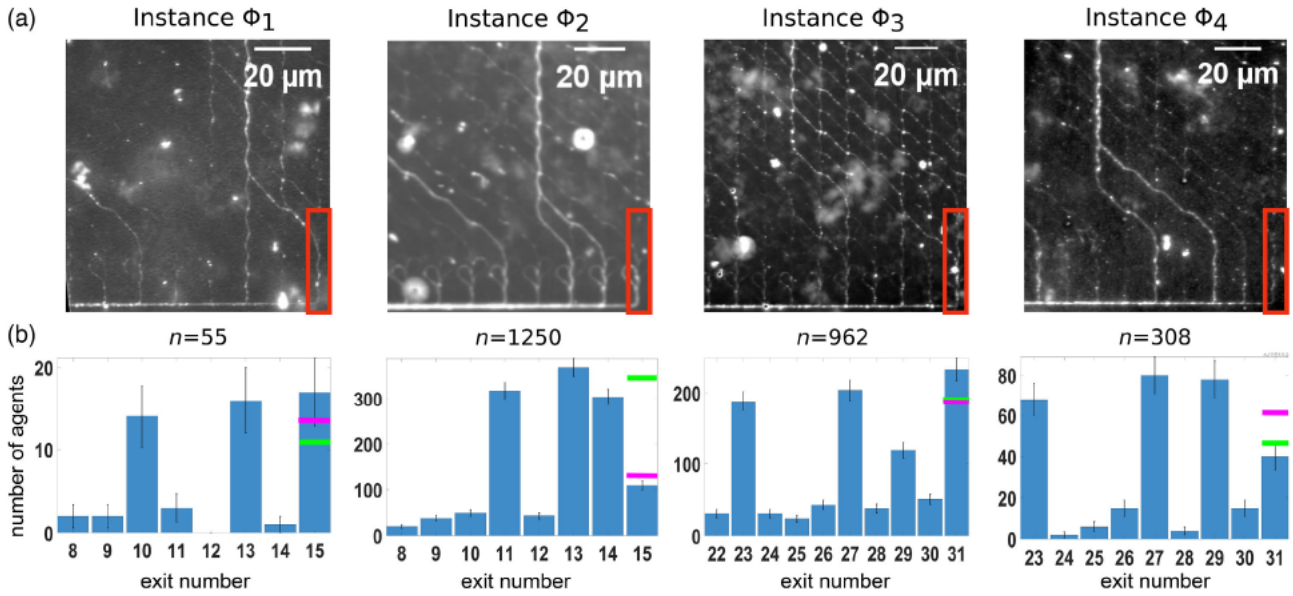

Figure 4. Experimental read-out for four instances of 3-SAT (Φ_1, Φ_2, Φ_3, Φ_4,). a) Standard deviation of projections of 1000 typical fluorescence micrographs (duration of 400 s) of actin filaments moving through the device. b) Experimental results obtained from manually counted actin filaments passing the exits at the bottom. Total counted events (n) are displayed. Error bars represent the counting error ($\sqrt{n}$). For the target column (15 for Φ_1, Φ_2 and 31 for Φ_3, Φ_4, shown in red boxes in (a)), values above the magenta bar are significantly correct exits ($p < 0.01$, corresponding to 99% confidence level, see the Supporting Information-1), values below the green bar are significantly incorrect exits ($p < 0.01$, see the Supporting Information-1). Both criteria are mutually exclusive, otherwise, the result is not conclusive. For example, Φ_2 would be inconclusive if column 15 had received more than 132 and less than 333 filaments.

with a 99% confidence level. In contrast, our results suggest that the instances Φ_2 and Φ_4 are unsatisfied with a 99% confidence level.

3.2. Scaling Considerations

To estimate requirements for scaling our 3-SAT NBC algorithm to solving larger 3-SAT instances, we consider a benchmark instance with eight clauses and ten variables, which results in a network similar in size to a current electronic CPU: The network will have ≈2300 rows corresponding to a chip of $\approx 24 \times 24\ mm^2$, which is certainly feasible to fabricate. It is also fully reasonable to explore a network of this size using the actomyosin system, in view of the recently achieved longevity of motile function for several hours to almost a day[31] and a gliding velocity of $\approx 10\ \mu m\ s^{-1}$. To avoid a majority of filaments crossing the network making errors, the error rate at *pass junctions* needs to be below 0.07%,[15] a factor of 4 better than the best error rate reported for an NBC calculation so far[15] and achievable with error-free 3D junctions.[32] To ensure that a sufficient number of agents explore this network, ≈50 000 actin filaments are required,[14,33] which corresponds to $\approx 5 \times 10^{-10}$ mg actin, an amount that is available at a very low cost. Third, agents need to be detected efficiently at exits, for example, using electrical/optical sensors.[34–36] It is important to note that energy consumption would not pose any limitation due to the energy efficiency of the biomolecular motors.[14] The above developments all build on available technology and would thus be, in principle, readily achievable. However, the reliability of components such as 3D junctions and sensors would need to be improved and fully integrated into the networks, making the developments outside the scope of the present work. Concluding from these considerations, the major limiting factor for further scale-up of the SAT network encoding presented here is the unary encoding. The unary encoding requires that the networks grow exponentially with the number of clauses in the SAT problem. For example, each OR operation block needs 8 rows (and 8 columns) for 3 clauses, 32 rows for 5 clauses, and 256 rows for 8 clauses. The number of OR operation blocks required is one less than the number of variables. Thus, the network grows exponentially with the number of clauses and linearly with the number of variables (i.e., $S \propto 2^c \vee n$, where S is the network size, c is the number of clauses and n is the number of variables). Notably, the energy required to fabricate the network, and the run time of the network design algorithm, both are proportional to the network size and thus have the same scaling behavior as the network itself. With the current network design, an instance with 8 clauses and 10 variables would require a network with 256×10 junction rows and columns. With a size of an individual junction of $\approx 10 \times 10\ \mu m$, this results in a network of $\approx 26 \times 26\ mm^2$, similar in size to a current CPU. Given that the fabrication technologies for CPU and biocomputation network are similar, the energy required for fabricating such a network would also be similar. This suggests that, in order to avoid exponential scaling of the network, it will be important to store information about the agent path in binary encoding—preferably on the agents themselves, rather than in the network. See Section 4 in ref.[37] for an example of how SAT could be solved with information stored on the agents. The corresponding network is independent in size of the number of clauses and

requires only one junction per variables, meaning that a SAT instance with 8 clauses and 10 variables would require a network with a size of $\approx 10 \times 100\,\mu m^2$. Such networks are expected to be limited, not by network size, but by the number of computing agents and the respective protein mass. For example, 2×10^{18}, 2 μm-long actin filaments weigh ≈100 g. If we assume an operation frequency of 0.1 Hz (similar to the frequency observed for the networks presented here), that number of actin filaments could perform as many computing operations per second as $\approx 5 \times 10^7$ CPU cores (where each operate at a frequency of ≈4 GHz). At the same time, the myosin motor proteins propelling this number of actin filaments would dissipate only ≈200 W of power (calculated as described in detail in the study by Nicolau et al.,[14] assuming 10 motors/filament, 100 steps s^{-1}, and 70 kJ mol^{-1} for the hydrolysis of ATP under physiological conditions[38]). Therefore, we believe that NBC has potential as an energy-efficient alternate computing technology. The challenges and strengths of this technology were recently detailed in a roadmap for network-based biocomputation.[37]

4. Conclusion

We have demonstrated the use of NBC to solve four small instances of the key NP complete problem, 3-SAT. To achieve this, a highly multidisciplinary approach was necessary. First, we developed an algorithm that encodes 3-SAT into the network format. We then used this algorithm to encode four small 3-SAT instances into four networks of channels. After nanofabrication and fine tuned selective functionalization of the network channels, myosin-driven actin filaments effectively searched the entire network. Finally, we devised a statistical approach to determine whether a significant number of filaments ($p < 0.01$) arrived at the target exit, indicating whether the respective 3-SAT instance was satisfiable or not. All networks were solved successfully with a 99% confidence level, demonstrating correct network operation. This is a key step toward the broad general applicability of NBC because it enables solving NP complete problem for which a polynomial conversion to SAT exists. Many of such conversions already exist, because NP completeness is usually proven mathematically by finding a polynomial conversion to SAT.[8] Thus, the network encoding of 3-SAT presented here enables NBC to solve small instances of any NP complete problem for which NP-completeness was proven via a polynomial conversion to SAT.

5. Experimental Section

Fabrication of Computational Networks: The fabrication of the computational networks was done on a single-side-polished 150 mm Si (100) wafer. A 70 nm-thick SiO_2 on the Si substrate was made by dry thermal oxidation. The layer growing was carried out under oxygen atmosphere with 3% HCl. After O_2 plasma cleaning and surface activation steps for 30 s, a resist layer (ALLRESIST AR-P 6200:10) was spin coated onto the SiO_2, to a thickness around 400 nm, and hard baked at 180 °C for 2 min. The network was patterned by electron beam lithography (Vistec SB254) with a dose of around 100 μC cm^{-2}. The AR-P 6200:10 was developed in 600 594 developer for 60 s, rinsed with IPA, and flushed in a conductance-controlled deionized (DI) water bath. Finally, the wafer was spun dry and separated into 1 cm^2 chips.

To make the surface suitable for protein anchoring and support of motility, silanization with trimethylchlorosilane (TMCS; Sigma-Aldrich Sweden AB, Stockholm, Sweden) was performed. In detail, the samples were O_2 plasma etched (Plasma Preen II-862, Plasmatic Systems, Inc., North Brunswick, NJ) at 5 mbar for 45 s to remove resist residues and remove the adsorbed water molecules. Etching was performed within a Faraday cage to achieve isotropic etching so the sidewalls in the nanostructures were etched as well.[31] The samples were then immersed in DI water for 5 min to generate as many as possible hydroxyl groups on the surface for later silanization. The silanization was then performed in a controlled chemical vapor deposition (CVD) system following the method in the study by Lindberg et al.[39]

Protein Preparations: Rabbits were sacrificed in accordance with the ethical guidelines and procedures approved by the Regional Ethical Committee for Animal Experiments in Linköping, Sweden (ref. no.: 73-14). Myosin-II was isolated from rabbit hind leg fast muscles,[40,41] followed by digestion with Tosyl-L-lysine-chloromethyl ketone hydrochloride (TLCK) treated α-chymotrypsin to obtain heavy meromyosin (HMM).[42] Actin was isolated from rabbit back muscles.[43] Fluorescence labeling of actin filaments was achieved using Rhodamine–phalloidin.[44]

Solutions and Chemicals: For all solutions used in the in vitro motility assay type of experiments, low-ionic strength solution (LISS) was first prepared as a basal component, containing 1 mM magnesium chloride ($MgCl_2$), 10 mM 3-(*N*-morpholino) propanesulfonic acid (MOPS), 0.1 mM K_2 ethylene glycol tetra-acetic acid (EGTA), with a pH of 7.4 and ionic strength of 15 mM. Wash solution was prepared by adding 1 mM Dithiothreitol (DTT) and 50 mM potassium chloride (KCl) to the LISS solution (final concentrations). Finally, assay solution was prepared by adding 0.2 mg mL^{-1} creatine phosphokinase (CPK), 2.5 mM creatine phosphate (CP), 1 mM magnesium adenosine triphosphate (MgATP), an oxygen scavenger mixture (GOC): 3 mg mL^{-1} glucose : 0.1 mg mL^{-1} glucose oxidase, 0.02 mg mL^{-1} catalase, 10 mM DTT, and 45 mM (KCl) to the LISS solution, giving a final ionic strength of 60 mM.[45,46]

In Vitro Motility Assays: HMM, fluorescently labeled actin and nonfluorescent-blocking actin,[47] were diluted in the wash solution. To perform the in vitro motility assay, flow cells were built using a glass coverslip and a nanostructured chip on top, separated with Parafilm spacers.[21,39] Flow cells were incubated using the following steps: 120 μg mL^{-1} HMM (5 min); 1 mg mL^{-1} bovine serum albumin (2 min); wash solution (washing, 1×); 0.5 μM blocking actin (2 min); 1 mM MgATP dissolved in wash buffer (2 min); wash solution (washing, 1×); 15 nM rhodamine–phalloidin-labeled actin filaments (2 min); wash (1×) and final incubation with assay solution (2 min). The in vitro motility assays for all nanostructured chips were performed within the temperature range 24–26 °C using a temperature-controlled objective coil. Usually, the flow cells were sealed with silicone vacuum grease during experimental observations.

Imaging Methods: In vitro motility assays experiments were imaged using an inverted fluorescence microscope (Zeiss Axio Observer.D1), mounted with either 63× or 40× objective (numerical aperture: 1.4 or 1.3) and Hamamatsu EMCCD camera (512 × 512 pixels). Image sequences were recorded at an exposure time of either 200 ms (4.96 frames s^{-1}) or 400 ms (2.5 frames s^{-1}). A mercury short-arc lamp (OSRAM) was used along with a filter (cyanine-3) for the excitation of rhodamine fluorescence. Images were then analyzed with Fiji software.[48]

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

J.Z. and A.S. contributed equally to this work. This work was funded by: European Union's Horizon 2020 research and innovation programme under grant agreement no 732482 (Bio4Comp) and NanoLund, Lund

University. Correction added on 17 August 2023, after first online publication: Projekt Deal funding statement has been added.

Open Access funding enabled and organized by Projekt DEAL.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

T.K., H.L., S.D., A.M., H.K. and D.R.: took care of conceptualization and supervision. J.Z., T.K., and H.K.: took care of methodology. J.Z., C.M., P.S., R.L, and A.S.: took care of investigation. J.Z. and T.K.: took care of formal analysis. J.Z., A.S., and T.K.: took care of writing the original draft.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Keywords

molecular motors, nanofabrication, network-based biocomputation, nondeterministic polynomials, parallel computation, satisfiability problems

Received: July 13, 2022
Revised: August 19, 2022
Published online: October 2, 2022

[1] K. L. McMillan, presented at *Int. Conf. on Computer Aided Verification*, Boulder, CO, July 2003.
[2] H. Kautz, B. Selman, presented at *IJCAI*, Stockholm, Sweden, July-August, **1999**.
[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA **2009**.
[4] E. Clarke, M. Talupur, H. Veith, D. Wang, presented at *Int. Conf. on Theory and Applications of Satisfiability Testing*, Santa Margherita Ligure, Italy **2003**.
[5] Y. Roudi, J. Tyrcha, J. Hertz, *Phys. Rev. E: Stat. Nonlinear Soft Matter Phys.* **2009**, *79*, 051915.
[6] J. J. Rice, G. Stolovitzky, Y. Tu, P. P. de Tombe, *Biophys. J.* **2003**, *84*, 897.
[7] G. De las Cuevas, T. S. Cubitt, *Science* **2016**, *351*, 1180.
[8] M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Vol. *174*, Freeman, San Francisco, **1979**.
[9] A. Biere, M. Heule, H. van Maaren, *Handbook of Satisfiability*, Vol. *185*, IOS Press, Amsterdam **2009**.
[10] J. P. Marques-Silva, K. A. Sakallah, *IEEE Trans. Comput.* **1999**, *48*, 506.
[11] T. D. Hansen, H. Kaplan, O. Zamir, U. Zwick, in *Proc. of the 51st Annual ACM SIGACT Symp. on Theory of Computing*, Association for Computing Machinery, Phoenix, AZ **2019**.
[12] Q. Liu, L. Wang, A. G. Frutos, A. E. Condon, R. M. Corn, L. M. Smith, *Nature* **2000**, *403*, 175.
[13] R. S. Braich, N. Chelyapov, C. Johnson, P. W. Rothemund, L. Adleman, *Science* **2002**, *296*, 499.
[14] D. V. Nicolau, Jr., M. Lard, T. Korten, F. C. van Delft, M. Persson, E. Bengtsson, A. Mansson, S. Diez, H. Linke, D. V. Nicolau, *Proc. Natl. Acad. Sci.* **2016**, *113*, 2591.
[15] D. V. Nicolau, D. V. Nicolau, G. Solana, K. L. Hanson, L. Filipponi, L. S. Wang, A. P. Lee, *Microelectron. Eng.* **2006**, *83*, 1582.
[16] T. Korten, S. Diez, H. Linke, D. V. Nicolau, H. Kugler, *N. J. Phys.* **2021**, *23*, 085004.
[17] L. Hardy, arXiv preprint quant-ph/0101012 **2001**.
[18] W. Guo, J. Wang, M. He, X. Ren, Q. Wang, W. Tian, presented at *2018 IEEE 4th Int. Conf. on Computer and Communications (ICCC)*, IEEE, Piscataway, NJ, December 2018.
[19] S. J. Kron, J. A. Spudich, *Proc. Natl. Acad. Sci.* **1986**, *83*, 6272.
[20] R. Bunk, M. Sundberg, A. Mansson, I. A. Nicholls, P. Omling, S. Tagerud, L. Montelius, *Nanotechnology* **2005**, *16*, 710.
[21] M. Sundberg, R. Bunk, N. Albet-Torres, A. Kvennefors, F. Persson, L. Montelius, I. A. Nicholls, S. Ghatnekar-Nilsson, P. Omling, S. Tågerud, A. Månsson, *Langmuir* **2006**, *22*, 7286.
[22] H. Hess, G. Saper, *Acc. Chem. Res.* **2018**, *51*, 3015.
[23] G. Saper, H. Hess, *Chem. Rev.* **2020**, *120*, 288.
[24] C. Reuther, R. Catalano, A. Salhotra, V. Vemula, T. Korten, S. Diez, A. Mansson, *N. J. Phys.* **2021**, *23*, 125002.
[25] A. Mansson, R. Bunk, M. Sundberg, L. Montelius, *J Biomed Biotechnol* **2012**, *2012*, 647265.
[26] M. Lard, L. ten Siethoff, J. Generosi, M. Persson, H. Linke, A. Mansson, *IEEE Trans. Nanobiosci.* **2015**, *14*, 289.
[27] A. S. Perumal, M. Nayak, V. Tokárová, O. Kašpar, D. V. Nicolau, presented at *Int. Conf. on Bio-inspired Information and Communication*, Cham, Swizerland **2019**.
[28] T. Q. Uyeda, S. J. Kron, J. A. Spudich, *J. Mol. Biol.* **1990**, *214*, 699.
[29] M. Persson, E. Bengtsson, L. ten Siethoff, A. Mansson, *Biophys.1 J.* **2013**, *105*, 1871.
[30] J. Y. Zhu, T. Korten, H. Kugler, F. van Delft, A. Mansson, D. Reuter, S. Diez, H. Linke, *N. J. Phys.* **2021**, *23*, 105004.
[31] A. Salhotra, J. Y. Zhu, P. Surendiran, C. R. Meinecke, R. Lyttleton, M. Usaj, F. W. Lindberg, M. Norrby, H. Linke, A. Mansson, *N. J. Phys.* **2021**, *23*, 085005.
[32] C. Reuther, S. Steenhusen, C. R. Meinecke, P. Surendiran, A. Salhotra, F. W. Lindberg, A. Mansson, H. Linke, S. Diez, *N. J. Phys.* **2021**, *23*, 125002.
[33] M. Konopik, T. Korten, H. Linke, E. Lutz, *N. J. Phys.* **2021**, *23*, 095007.
[34] M. Curreli, R. Zhang, F. N. Ishikawa, H. K. Chang, R. J. Cote, C. Zhou, M. E. Thompson, *IEEE Trans. Nanotechnol.* **2008**, *7*, 651.
[35] K. I. Chen, B. R. Li, Y. T. Chen, *Nano Today* **2011**, *6*, 131.
[36] X. Duan, Y. Li, N. K. Rajan, D. A. Routenberg, Y. Modis, M. A. Reed, *Nat. Nanotechnol.* **2012**, *7*, 401.
[37] F. C. M. J. M. van Delft, A. Månsson, H. Kugler, T. Korten, C. Reuther, J. Zhu, R. Lyttleton, T. Blaudeck, C. R. Meinecke, D. Reuter, S. Diez, H. Linke, *Nano Futures* **2022**, *6*, 032002.
[38] H. Wackerhage, U. Hoffmann, D. Essfeld, D. Leyk, K. Mueller, J. Zange, *J. Appl. Physiol.* **1998**, *85*, 2140.
[39] F. W. Lindberg, M. Norrby, M. A. Rahman, A. Salhotra, H. Takatsuki, S. Jeppesen, H. Linke, A. Mansson, *Langmuir* **2018**, *34*, 8777.
[40] S. J. Kron, Y. Y. Toyoshima, T. Q. Uyeda, J. A. Spudich, *Methods Enzymol.* **1991**, *196*, 399.
[41] M. Sata, S. Sugiura, H. Yamashita, S. Momomura, T. Serizawa, *Circ. Res.* **1993**, *73*, 696.
[42] Y. Okamoto, T. Sekine, *J. Biochem.* **1985**, *98*, 1143.
[43] J. D. Pardee, J. B. T. M. I. E. Aspudich, *Methods Cell Biol.* **1982**, *85*, 164.
[44] M. Balaz, A. Mansson, *Anal. Biochem.* **2005**, *338*, 224.
[45] E. Homsher, F. Wang, J. R. Sellers, *Am. J. Physiol.* **1992**, *262*, C714.

[46] M. Persson, M. Gullberg, C. Tolf, A. M. Lindberg, A. Mansson, A. Kocer, *PLoS One* **2013**, *8*, 55931.
[47] M. A. Rahman, A. Salhotra, A. Mansson, *J. Muscle Res. Cell Motil.* **2018**, *39*, 175.
[48] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, J. Y. Tinevez, D. J. White, V. Hartenstein, K. Eliceiri, P. Tomancak, A. Cardona, *Nat. Methods* **2012**, *9*, 676.