Sudalaiyadum Perumal et al. *Microsystems & Nanoengineering* (2026) 12:287
https://doi.org/10.1038/s41378-026-01341-x

Microsystems & Nanoengineering
www.nature.com/micronano

ARTICLE Open Access

Amplification of computational power by the multiplication of bacteria exploring microfluidic networks encoding mathematical problems

Ayyappasamy Sudalaiyadum Perumal[1], Falco C. M. J. M. van Delft[2], Giulia Ippoliti[1], Ondřej Kašpar[1,3], Viola Tokárová[1,3], Monalisha Nayak[1], Matthew Cho[1], Jessica Li[1], Anja van Langen-Suurling[4], Charles de Boer[4], Frank Dirne[4], Dan V. Nicolau Jr[2,5,6]✉ and Dan V. Nicolau[1]✉

Abstract

Computational resources required for solving NP-complete problems grow exponentially with a polynomial increase in problem size. This exponentially increasing computational resource is runtime for sequential electronic computers and space for massively parallel DNA computing. Here, we report the proof of concept of a computer, whose operation consists in the exploration by motile bacteria of a microfluidic network encoding an algorithm for solving the Subset Sum Problem (SSP)—a classic NP-complete problem. Significantly, this computer performs operations combinatorially, via natural multiplication of bacteria operating as biological CPUs, translating into the continuous amplification of computational power, which grows seamlessly to match the problem size. A scaling analysis identifies the point where biocomputing with multiplying bacteria is expected to outperform electronic computers. The combinatorial, design-driven low error operation, low energy requirement for computing, and exponentially growing computational resources suggest that bacterial-driven biocomputation using microfluidic networks holds the potential to scale up successfully.

Introduction

Combinatorial mathematical problems, such as Non-deterministic Polynomial (NP)- complete problems, are crucial to myriad different and important applications, from information management[1], cryptography[2], and microelectronics[3], to molecular biology[4]. For these problems, the number of operations grows exponentially with a polynomial increase in the size of the problem.

Electronic computers remain the fastest and most precise hardware for computation[5], but they process operations largely sequentially. For combinatorial problems, this means that the required computing time grows exponentially with problem size, eventually making such problems intractable[6] for solid state-based computation. For DNA computing[7] and biocomputing with agents in networks[8] the exponentially increasing computational resource is DNA mass, and the volume of the microfluidic network, respectively.

Fundamentally, the computing approaches listed above address the exponential increase of the solution space with the size of combinatorial problems by using a physical quantity that also grows exponentially with the size of the problem, and by attempting to make this exponential increase technologically tolerable. In Moore's words, "No physical quantity can continue to change exponentially forever. Your job is delaying forever."[9] A more detailed discussion regarding computational difficulties when solving NP-complete problems is presented in SI-1.1.

Correspondence: Dan V. Nicolau Jr. (dan.nicolau@kcl.ac.uk) or Dan V. Nicolau (dan.nicolau@mcgill.ca)
[1]Department of Bioengineering, McGill University, Faculty of Engineering, Montreal, QC, Canada
[2]Molecular Sense Ltd., Liverpool, UK
Full list of author information is available at the end of the article
These authors contributed equally: Ayyappasamy Sudalaiyadum Perumal, Falco C. M. J. M. van Delft, Giulia Ippoliti

The Subset Sum Problem (SSP), an NP-complete problem, was proposed to be solved by network-based biocomputing[10,11]. The network-based computing, which capitalizes on the straightforward representation of NP-complete problems as graphs, proposes to solve combinatorial problems in three conceptual steps[12]: (i) encoding an algorithm for solving the problem of interest in a graph; (ii) translation of this abstract graph into a design of a physical network, followed by its fabrication; and (iii) actual computation, consisting in the exploration of this network, in a massively parallel manner, by a very large number of microscopic motile agents. Implementations of this approach consisted of using self-propelled cytoskeletal filaments, i.e., actin filaments or microtubules, as computational agents exploring a microfluidic network functionalized with complementary molecular motors, i.e., myosin or kinesin, respectively[8], or photons[13]. The major drawback of this approach is that the process is bottlenecked by the sequential 'booting' of the computer, thus still leading to an overall computing time scaling inadequately for instances of large problems.

Here, we report a proof of concept of a computing device based on the bacterial exploration of a network encoding the Subset Sum Problem. Aside from being robust and offering multiple optimization paths of the device and its operation, bacteria have the innate ability to divide. Bacterial division, while exploring the computational network, is equivalent to the multiplication of CPUs, thus amplifying the computational parallelism during data processing. We also present advances in the design and operation of the computer, perspectives, and possible paths for future developments.

Results and discussion

Network computing of the Subset Sum Problem using bacteria

Design and operation principles of a network encoding the Subset Sum Problem

The SUBSET SUM Problem (SSP) is one of the first reported NP-complete problems[14], with enduring importance[15]. Importantly, SSP translates other NP-complete problems[16,17], for instance, Number Partitioning[18], Job Sequencing[19], Steiner Tree[20], Exact Cover[21], and Satisfiability Problem (k-SAT)[22]. SI-1 presents a 'genealogy' of NP complete problems (SI-1.1, Fig. S5)[17], and Table S3, SI-1.2 to SI-1.5 provides an explicit example of translating an SSP instance into another NP-complete problem (Fig. S1.1a, b). From the operational point of view, we used SSP as a test NP-complete problem and bacterial-driven biocomputation devices because SSP offers a straightforward scalability comparison with prior[8] and recent work[13], as well as with DNA computing[23]. The importance of SSP for an important application, i.e., public-key cryptography[24], is detailed in SI-1.6.

SSP asks whether, given a set $S = (s_1, s_2, \ldots, s_N$ of N integers, where N represents the cardinality of the problem, there are subsets of S whose elements add up to a target integer, T. The number of possible subset combinations expressed as 2^N rises exponentially[25]. Given its NP-completeness, no efficient algorithms for solving any instance of this problem are known[26]. Despite various faster algorithms for SSP[27–29], to date, none of them could solve it in polynomial time, and they also require an exponential increase in memory space with the size of the problem. Finally, while dynamic programming solves SSP in pseudo-polynomial time[30], it does so at the expense of an exponential increase in the required physical memory (SI-1.4 and Fig S1.1A).

Figure 1 illustrates a network encoding of SSP[10], which is an application of the naïve, inclusion-exclusion algorithm[10,11,31]. This implementation has the property that a path exists between an entrance node and any one of the several exit nodes if, and only if, there is a subset of 'S' whose elements sum up to the node assigned to that integer sum. This methodology can be also used to encode, in physical form, a realization of the dynamic programming method[32] for solving SSP (Figs. S1.3 and S1.1A).

The network presented in Fig. 1 has a triangular design (Fig. S6), implemented as a planar mapping of a directed graph, with a single starting point ("Entry" at bottom-left, Figs. 1b and S6A. The network comprises: (i) Split junctions (Figs. 1c and S6B, left), where computational agents can change direction with a (preferably) 50%-50% distribution, (ii) Pass junctions (Figs. 1c and S6B, middle), where agents ideally cross without change of direction; and (iii) Join junctions (Figs. 1c and S6B, right), where traffic lines merge. The addition of an integer 'n' to a subset is represented by turning left at a Split junction and proceeding to the next (n-1) Pass junctions diagonally upwards. Skipping this addition of n is represented by turning right at that Split junction and proceeding horizontally to the next (n-1) Pass junctions. Importantly, in this design of the network computer, the Exits represent all possible target solutions, not only for the target integer, T, as in the original mathematical formulation of SSP. In an error-free computation, agents will exit only at the valid solution values ("Exit" at right in Fig. 1b, colored squares). Finally, to remove erroneous 'ghost' traffic, i.e., agents that did not follow correctly the logic embedded in the junctions, 'dirt lanes' ending in specially designed traps were placed on the bottom and the left (diagonal) side (Figs. 1b and S6A(c)).

The computational process must find the correct T values, that is, to answer the first question (Q1): "Is a valid solution possible (at T)?". While providing an answer to Q1 is already solving an NP-complete problem[16], in many practical cases, it is desirable to also provide the

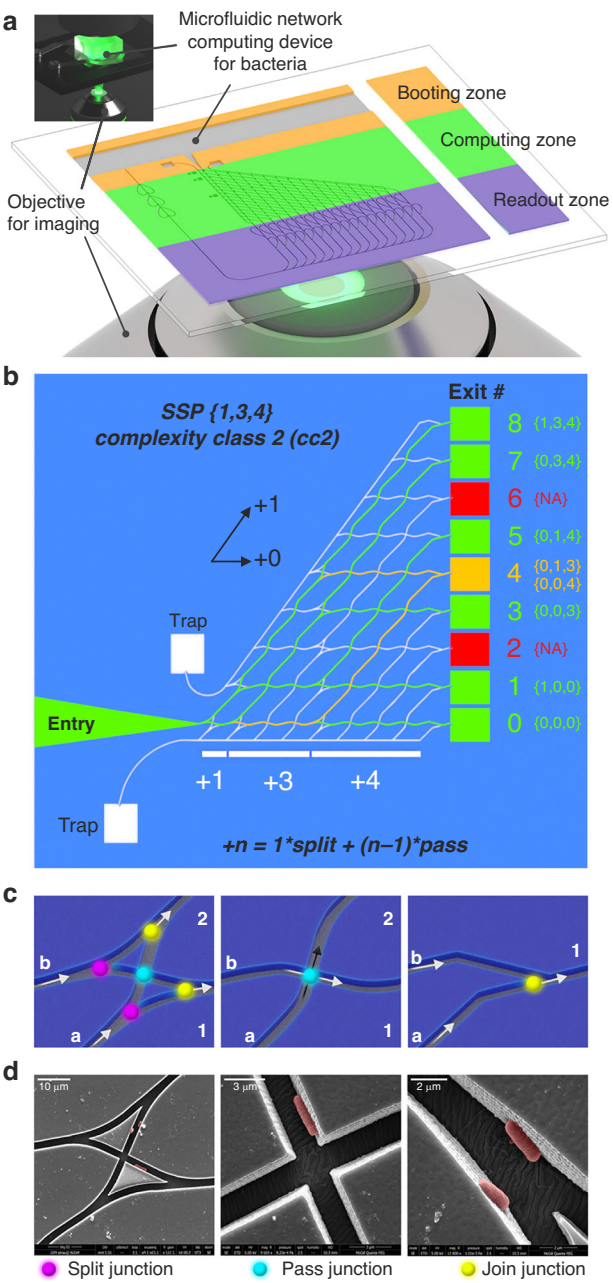


Fig. 1 Network design for solving the complexity class 2 SSP (1,3,4). An example of multiple subset routes for one valid exit is shown in yellow: the routes (+1, +3, +0) and (+0, +0, +4) both end at exit 4. **a** Overall design. **b** junction designs. **c** SEM images of junctions with bacterial agents in the channels (roofs removed)

composition of all combinations of elements in a subset which sum up to a specific T value, that is, to answer the question (Q2): "By which variable assignment(s) is this solution reached?". In the special case where every valid solution can only be reached by one route, i.e., by only one, unique subset of numbers (unique SSP), we denominate the associated SSP instance a *complexity class 1* (cc 1) problem[33]. In Fig. 1, an instance of the more complex case is shown, where one or more valid solutions can be reached by multiple routes, i.e., by more than one combination of numbers. Such an SSP instance is a *complexity class 2* (cc 2) problem. The construction and operation of the computing device are detailed in SI-2 and Movie S1.

The manufacturing process[33] (presented in Materials and Methods MM-1 to MM-3, Figs. S1 and S2) comprises (i) fabrication of a negative profile of the microfluidic computational network in silicon; (ii) preparation of the network via casting PDMS on this master; (iii) sealing the oxygen-permeable PDMS cast network on a transparent substrate, e.g., glass; and (iv) incubation of the computer with a solution containing bacteria. This approach has several significant benefits. First, once the agents enter the network, they are confined until the end of the computation or, if in error, are guided outside the network. Compared with previous work[8] this closed architecture leads to an inherent reduction in errors, e.g., agent escape, or worse, parasitic agents 'joining' the computation from surrounding fluids. Second, PDMS permeability to oxygen allows for the maintenance of the viability of aerobic bacteria. Third, the PDMS-based replication of the computational network can facilitate multiple experiments, including running them in parallel.

The lateral dimensions of the computational conduits in the actual device were dictated by the size of the bacterial computational agents SI-2.1 and SI-2.2 (Fig. S6), that is, the required widths and depths of the channels must be larger than 1 µm, i.e., bacterial cell width, to minimize crowding, but in the same time maintain movement directionality[34]. Devices with these dimensions can be fabricated, on large areas, by optical photolithography. Still, because error-free junction operation is essential to the accuracy of the computation, we employed e-beam lithography for precise manufacturing at the nm scale (presented in MM-1 Fig. S2).

Bacteria as computational agents

Bacteria offer certain advantages as computation agents in network-based biocomputing[33]: (i) there is a wide variety of candidates, with detailed information on velocity, flagellar arrangements, sizes, morphology, and division rates[35]; (ii) bacteria require low-cost and straightforward, well-known experimental protocols; (iii) a wide range of fluorescent tagging and tools for gene editing for programmable traits are readily available; (iv) each bacterium is an independent, autonomously motile, computing unit with an inherent capacity to process; and importantly, (iv) bacteria divide, thus providing the self-multiplication of the computing power.

Using these performance criteria, we employed *Escherichia coli* HCB437 and *Vibrio natriegens* as motile computation agents described in M2.2 (Table S1). *E. coli*

HCB437 (ΔcheA to ΔcheZ), a strain studied extensively, with well-known motility characteristics, including in microfluidic devices[36,37], presents a smooth swimming phenotype due to a mutation-induced impairment of the chemo-sensor[38]. *V. natriegens* is a more recently studied species with a high division rate (doubling every 9–12 min at room temperature)[39]. The first stage of revealing the advantages of bacterial division on computation performance consisted of a baseline study focused on slow-dividing *E. coli* with negligible chances of multiplication in the timeframe of solving a chosen problem, followed by experiments using *V. natriegens* with higher multiplication rates during computation.

An SSP network (Figs. 1; S6A) consists of an array of junctions. The Pass junction is designed to maximize the proportion of visiting agents maintaining their current direction of motion, requiring 90° angle intersections (Figs. S6A, S2.1), making the diversion of bacterial trajectories physically difficult. In contrast, while the optimal geometry of the Split junctions would be an equal Split of the directions of the agents after visiting the junctions, their optimal design was decided by fabrication considerations, that is, by the uniformity of structures repeating throughout the network, rather than the motility characteristics of individual bacterial species.

Turning preferences of bacteria, which are species-specific[35], can also be influenced by the wall materials[36] and by the network geometry (SI-2.2; Fig. S7). Figure 2a, b present the bacterial error rates at different junctions. Further examples of motility patterns, e.g., U-turns, modulated by channel width and height, are presented in Fig. S6C and Movie S2. First, the Split junctions (Figs. 2a, SI-2.2 and S7) induce a near-equal distribution of directionality, i.e., 50.1% left vs. 49.8% right for *E. coli*, whereas *V. natriegens* presents a sizable directional bias, i.e., 61.1% left vs. 38.3% right, due to the asymmetry of the monoflagellated architecture. Second, both species present low error rates in Pass junctions (Fig. 2b), i.e., 1.3% for *E. coli* and 0.3% for *V. natriegens*, because of additional confinement of the PDMS roof on top of the junctions. The resilient operation of the split and pass junction is demonstrated by exploring bacteria (in this case, *E. coli* HCB 437) of different sizes and motility profiles through various case scenarios of junction trajectories. The ability of bacteria to accommodate adjacent motile bacteria, explore, and operate without blocking is presented in Movie S2. All experimental studies used triplicate experiments.

Readout of computation results

Quantitative and qualitative solutions to SSP using exit counts and density maps

The readout of the solutions of SSP instances can be obtained using two methods: *exit counting*, indicating the number of agents arriving at each exit, which responds to the first SSP question (Q1): "Is a valid solution possible (at T)?"; and *density maps*, obtained by adding the pixels from the motility patterns, which are qualitative representations proving answers to both Q1 and a second SSP question (Q2): A detailed discussion is presented in MM 3, SI-2.3 and Movie S3, SI-2.4 and Movie S4; Fig. S3.

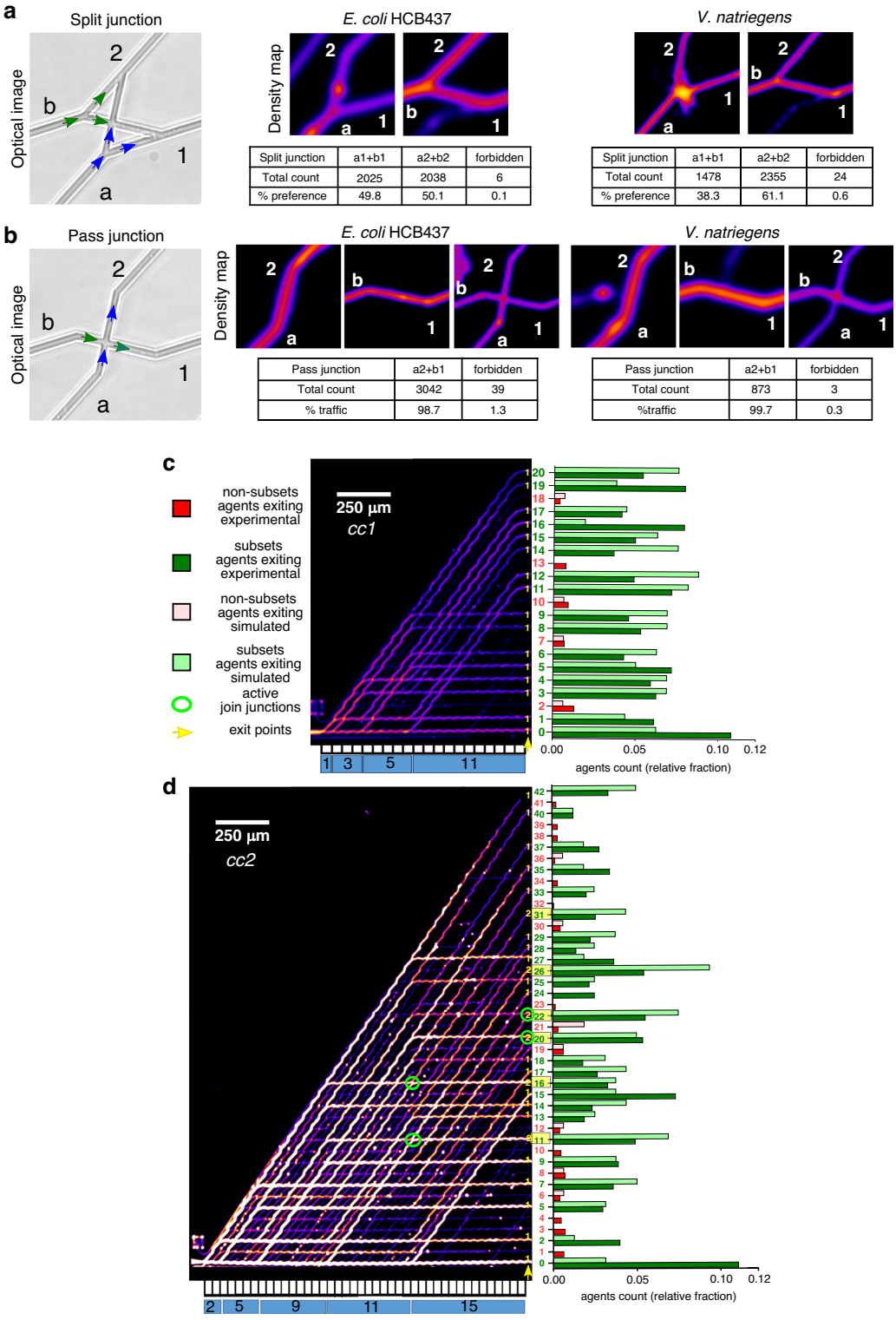
Decoding the SSP solutions from density maps

Our experimental demonstrations of bacteria exploring an SSP network show that the resulting density map contains sufficient information to solve both Q1 and Q2, even without prior knowledge of the employed set. However, each complexity class requires its own specific procedure to find the numbers that make the correct sums. A straightforward inspection of the density maps, which can be easily automated, will rapidly identify the SSP complexity class: the number of exits visited correctly equals 2^N for complexity class 1 networks, but it is smaller for complexity class 2. For complexity class 1, i.e., where each path to an exit representing a correct sum is unique, finding the numbers accruing to a correct sum consists of finding the junctions where the path is deflected, and then finding the numbers on the horizontal axis of the SSP network corresponding to these deflections. For complexity class 2, i.e., when more than one path leads to an exit representing a correct sum, finding the numbers accruing to a correct sum requires additional steps. The readout methodologies for both complexity classes, both easily made automatic, are presented in detail in SI-2.3 and SI-2.4 for SSP20@4 (Movie S3) and SSP42@5.

Non-dividing bacterial agents solving Subset Sum Problems

To explore the fabrication, operational, and readout limits of network biocomputation in the absence of multiplication of the agents, we tested the performance of solving SSP networks with an increasing number of instances and for both classes of complexity (Figs. 3c and d; S8-S10). The specifications of these computing networks, together with quantitative data calculated from the long-run experiments, are presented in SI-2.5, Table S4. This 'mini-Moore's Law' experimental plan, with examples provided in Figs. S8 and S9, did not reveal any fabrication, operational, or readout difficulties. Finally, these experiments demonstrated that bacteria exhibit lower error rates in junctions than other biological agents previously used for solving SSP networks[8]. Importantly, these errors are plateauing for larger SSP networks (SI-2.6, Fig. S10).

The lack of apparent difficulties regarding the fabrication, operation, and readout for bacteria exploring larger SSP networks, as well as the previously demonstrated



Split junction	a1+b1	a2+b2	forbidden
Total count	2025	2038	6
% preference	49.8	50.1	0.1

Split junction	a1+b1	a2+b2	forbidden
Total count	1478	2355	24
% preference	38.3	61.1	0.6

Pass junction	a2+b1	forbidden
Total count	3042	39
% traffic	98.7	1.3

Pass junction	a2+b1	forbidden
Total count	873	3
%traffic	99.7	0.3

Fig. 2 (See legend on next page.)

(see figure on previous page)

Fig. 2 Bacterial traffic at the network junctions. Description of correct paths in Split (**a**) and Pass (**b**) junctions—blue and green arrows represent the correct paths, and the possible paths are labelled '**a'** and '**b'** for Split and for Pass junctions. Any path other than those indicated is forbidden and leads to computing errors. Middle and right panels. Bacterial distribution, for *E. coli* and *V. natriegens*, across Split (top row) and Pass-junctions (bottom row) visualized as a summation of bacterial trajectories (density maps). The distribution of paths, for *E. coli* and *V. natriegens*, for each junction type is presented as the number of agents that passed successfully through the respective junctions, their selected path at left/right splits, and the number of agents taking forbidden paths (illegal/U-turns) or discontinuing the computation, e.g., wall-immobilized bacteria. Bacteria solving complexity class 1 (**c**) and complexity 2 (**d**) SSP-encoded networks. The readout methodologies use density maps (left) and bar graphs (right) to process the trajectories, or the number of exits, respectively, taken by the computational agents (here, *E. coli*) exploring the SSP-encoded networks. The SSP20@4 with 210 nodes is a complexity class 1 network (**c**), and SSP42@5 with 903 nodes (**d**) is a complexity class 2 network. The sets of numbers are (1, 3, 5, 11) and (2, 5, 9, 11, 15) for complexity class 1 (**a**) and complexity class 2 (**b**), respectively. Exit numbers (indices) located at the right of the respective networks are marked in red or green for valid and invalid target sums, respectively. The yellow numbers at the base of the density maps correspond to the number of possible routes for reaching the corresponding exit. Green circles represent active Join junctions, where exits can be reached by more than one route, i.e., more than one valid subset combination. Bar graphs represent statistical averages of the ratio of agents in each exit to that of the total number of agents in the network for experimental (dark green and red) and simulated (light green and light red) data. Similarly, with the colors for exit indices, green represents correct exits, and red represents invalid exits. Solving the problem in the network for each combination of SSP input was tested at least in triplicate in each case. The total count of agents at the valid exits compared to that of the invalid exits was significantly different, tested by an unpaired Student's *t*-test and the Matt-Whitney satisfiability test ($P < 0.001$)

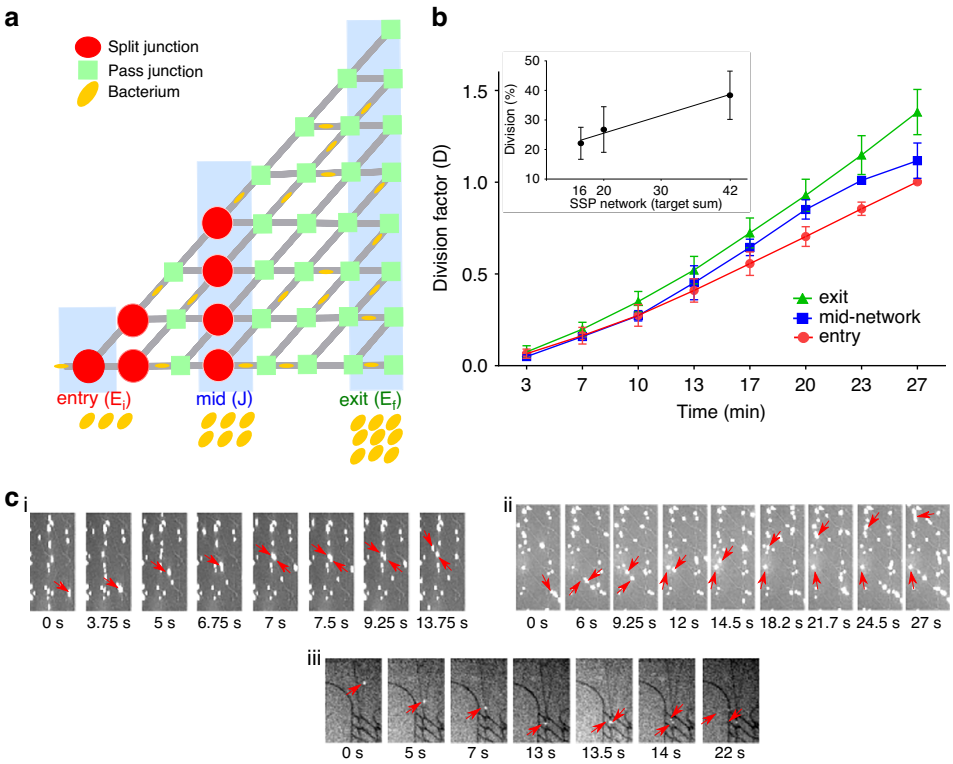


Fig. 3 Bacterial multiplication in a SSP network. a Conceptual representation of the methodology used for quantifying the time-resolved bacterial division. **b** Time-resolved quantitative estimation of the relative division factor for *V. natriegens* at the entry, middle, and exit regions of the SSP network. The inset represents the division percentage in three different-sized networks with different target sums. **c** Snapshots of bacterial division in the SSP network for *V. natriegens* (i and ii) and *E. coli* HCB437 (iii). *Single arrows:* Agents before the division; Double arrows: agents at the moment of cell division, and after dividing

similarity between the motility patterns of *E. coli* and *V. natriegens*[35], suggested upgrading the study to the demonstration of biocomputation in networks with agent multiplication using *V. natriegens*. Initial experiments, presented in detail in SI-2.6, aimed to solve small SSP instances, i.e., SSP 20@4 and SSP 42@5 (Figs. S10A and S10B, respectively), using density maps and exit counts. For these small instances, bacteria did not have significant opportunities for cell division, thus offering a species-specific baseline control. The successful computation by

V. natriegens exploring networks designed for *E. coli* suggests that the SSP network designs can accommodate various bacterial species with similar sizes and small differences in motility characteristics (e.g., speed and directionality bias) (Figs. S8 and S9). Following this benchmark experiment, all our division studies used *V. natriegens*, while a comparison with dividing *E. coli* was also drawn.

Computation of a combinatorial problem using multiplying agents

The theoretical concept of network computers using self-multiplication of computational biological agents was proposed[33] and recently compared with the theoretical performance of DNA and electronic computers[40]. As the number of possible solutions for an NP-complete problem increases exponentially with the size of the problem, and if all possible solutions must be visited, the logical conclusion is that to find all solutions, the computation will require an exponential increase in computing resources. Biocomputing with networks proposes that the computational resource needed to solve NP-complete problems encoded in microfluidic networks consists of a massively large number of motile agents, which perform logical operations when visiting junctions, similarly to Central Processing Units (CPUs). Bacteria are known to grow their numbers in constrained spaces that colonize, even in sub-micron microfluidic channels[41], which immediately recommends them, if motile, as exploring computational agents. Here, we present the experimental proof of concept of a computer whose microfluidic, SSP-encoded network is explored by bacteria with numbers growing exponentially during computation time. This approach is expected to reduce the exponential increase of computing time with the size of the NP-complete problem up to a polynomial relationship (an extensive discussion is presented in SI-3 and SI-4). This computer hardware technology, in which computing resources naturally grow to match a given problem size, proposes to 'defeat' an exponential, i.e., the NP-complete problem size, with another exponential, i.e., the computing resource growing exponentially during the computation process.

V. natriegens was the chosen computational agent for our proof-of-concept computer because of (i) its high division rate[39,42]; (ii) its straightforward fluorescent tagging and genetic manipulation, e.g., with mCherry protein-expressing plasmids[43]; and (iii) its similarity in size and motility characteristics (other than the small directional bias) with *E. coli*[44], thus making it possible to have a straightforward comparison of network computing with non- or slow-dividing agents. Importantly for both species, the division results in both bacterial cells moving initially in the same direction[45], (Movie S5 example 1 and 2, for *V. natriegens at pass junctions, example 3 for E. coli at a split junction*).

Examples of actual instances of agent multiplication in the network are presented in Fig. 3c and Movie S5. For a network computer operating with agent multiplication, the number of agents exiting the network is higher than the number entering (Fig. S3A and the respective quantification of the agent count at different stages: early, mid, and late stages of the network, Fig. 3a, b). Intuitively, the larger the SSP network with widely spaced combinations, the longer the time spent by the agents traversing the network, and thus the larger the opportunity for multiplication (Fig. 3b). The multiplication of bacteria in the network was quantified (Figs. 3 and S11) by threshold-based bacterial counters (SI-M2). The fraction of multiplied bacteria in the networks and the multiplication rates for each network with different sizes and cardinalities for both *E. coli* and *V. natriegens* are presented in Fig. S11.

To further quantify agent multiplication, we introduced an effective network-level growth rate (μ, h^{-1}) derived from entry–exit bacterial counts and residence-time estimates. The raw counts used for this analysis are reported in Table S5, while the corresponding effective growth rates across different SSP network sizes and bacterial species are summarized in Table S6. The calculation follows an exponential growth formalism detailed in SI-2.7. Across all SSP architectures, *V. natriegens* consistently exhibits higher effective growth rates than *E. coli*, reflecting its faster intrinsic division rate. Smaller SSP networks show lower effective growth rates due to shorter bacterial residence times, which limit opportunities for multiplication, whereas larger networks such as SSP100 provide residence times allowing more cycles of bacterial division (Table S6). This trend is further supported by the absolute bacterial counts at the entry and the exits of SSP100 networks, which show clear exponential amplification (Table S5).

The duration of the observation of computation (around 26 minutes for SSP42@5) offered *V. natriegens* ample time for two doubling cycles. Still, this computing time is not sufficient for the lower rate of division exhibited by *E. coli*. Indeed, *E. coli*, whose maximum doubling time is approximately 20 minutes at optimal conditions of 30 °C to 37 °C, and 30 minutes at room temperature[46], showed almost three times lower division rates than *V. natriegens*. In a very large network, e.g., prime numbers SSP 129@10, each agent would reside for at least 10 minutes before reaching the exits, leading to at least one bacterial division cycle, thus drastically increasing the agent population. Supplementary Materials and Methods M2 present the details of agent multiplication in the network and how a time-resolved quantification of agent division was derived. A discussion on the multiplication of bacterial 'CPUs' is detailed in SI-2.7, and examples of bacterial division in networks are presented in Movie S5.

Operational scaling of biocomputation in networks with motile agents

The scaling of biocomputation with networks using motile agents was previously assessed[47] from a 'roadmap' perspective, including through a comparison with other computing technologies, such as electronic computing and DNA computing. The estimation of the energy required for computation[8,13,33,40], concluded that biocomputation using networks explored by motile biological agents will consume orders of magnitude less energy per operation than solid-state computers. Importantly, bacteria exploring microfluidics networks with an excess of nutrients will operate in a chemotaxis-free environment[40], resulting in a considerably lower energy consumption[48]. While these estimations are encouraging, a conservative scaling exercise will also consider the *operational* aspects of biocomputation in a network with motile agents, such as the limits of using a very large number of agents, and resilience against errors, in the context of using bacteria as self-multiplying pseudo CPUs. Also, from an operational perspective, the rule for halting the computation leads to the revisiting of the estimated computing time.

Scaling assessment of the operational limits

The experiments using *E. coli* (Figs. 2b, c, S8 and S9, and Table S4) and *V. natriegens* (Fig. S10) demonstrated that bacteria can solve SSP instances of a series of prime numbers up to a total sum of 42, corresponding to cardinality $N = 7$, without any discernible operational problems. However, legitimate scaling questions arise regarding the further operational limits for network biocomputing related to the use of bacteria as computational agents (SI-3.2), especially concerning the depletion of the nutrients in confining spaces, the possible adherence of bacteria on walls leading to crowding (SI-3.1), and the multiplication-related filling of the networks towards the exits. Long-term experiments with bacteria moving in various microfluidics structures[35] did not show any decrease in bacterial viability for many hours, regardless of the confinement level. Furthermore, for larger microfluidic networks, nutrients can be delivered through a porous membrane placed on top of the computational network, e.g., as proposed[49] for motility assays using cytoskeletal filaments.

A more difficult problem could be the parasitic bacterial adhesion on the walls. Still, a previous study[35] studying the motility behavior of *E. coli* and *V. natriegens* (and three other bacterial species) demonstrated that the hydrophilization of PDMS walls was effective in cancelling bacterial adhesion. More importantly, this study showed that, while both *E. coli* and *V. natriegens* exhibit a motility pattern which can be classified as swimming parallel, but close to the walls, their movement is occasionally that of "wall escapers", and not that of "wall accumulators". This motility behavior inherently leads to avoidance of long-term contact with surfaces. A recent study[34] confirmed this observation on the optimum geometry of the junctions, enforcing a computationally correct and smooth bacterial movement.

For higher SSP instances, a more fundamental difficulty arises from the 'competition' between (i) agent multiplication, which is one of exponentially increasing computational resources, and which accelerates with the progression of computation, and (ii) the availability of physical space, which is the other exponentially increasing computational resource. This competition between the volume of biological CPUs and the volume of the computer could lead, in principle, to the agent crowding in the later stages of the network exploration, thus resulting in computational delays. Indeed, a recent study predicted that network biocomputing will shift from a polynomial relationship between problem size and computing time around cardinality 30 and 15 for SSP series comprising primary-, and unit- (only 1's) series, respectively[40]. While this theoretical analysis assumes that bacterial cells have rigid bodies, thus artificially overestimating crowding, it also suggests that experimental evidence of the operational limit of bacterial computing is fully warranted. This analysis also suggests that crowding occurs earlier in networks with a higher density of junctions per unit area, with Pascal's triangle geometry being the most compact.

Pascal's triangle is a triangular array, which is equivalent to an SSP instance comprising only the numbers 1, with cardinality equal to the number of 1 s, e.g., SSP 5@[1,1,1,1,1]. Interestingly, SSP networks with Pascal's triangle number series always present unit cells with Split and Join junctions (SI-3.2, Fig. S12). Conversely, other series, e.g., the prime numbers, also contain unit cells without Split and Join junctions, i.e., with a single Pass junction that can induce erroneous switching of the direction. Therefore, Pascal's triangle SSP networks are essentially error-free and consequently should be used as a litmus test for the operational capability of bacterial computing.

Initial tests solved smaller SSP instances, answering both Q1 and Q2 (Fig.S12), which demonstrated the operational viability of bacterial computing for networks with high junction densities. The maximum operational capability achieved on Pascal's triangle SSP networks was $N = 175$ (Fig. 4). While the experiments using Pascal's triangle networks are intended to test the operational limits of bacterial computing solely, it is noteworthy that for this SSP case instances with up to $N = 100$ could be solved, answering Q1 (Figs. 4, S13A–S14C), that is, running a network encoding a problem with 2^{100} solutions. The operation scalability and image acquisition for generating Q1-derived density map solutions are shown in Movie S6.

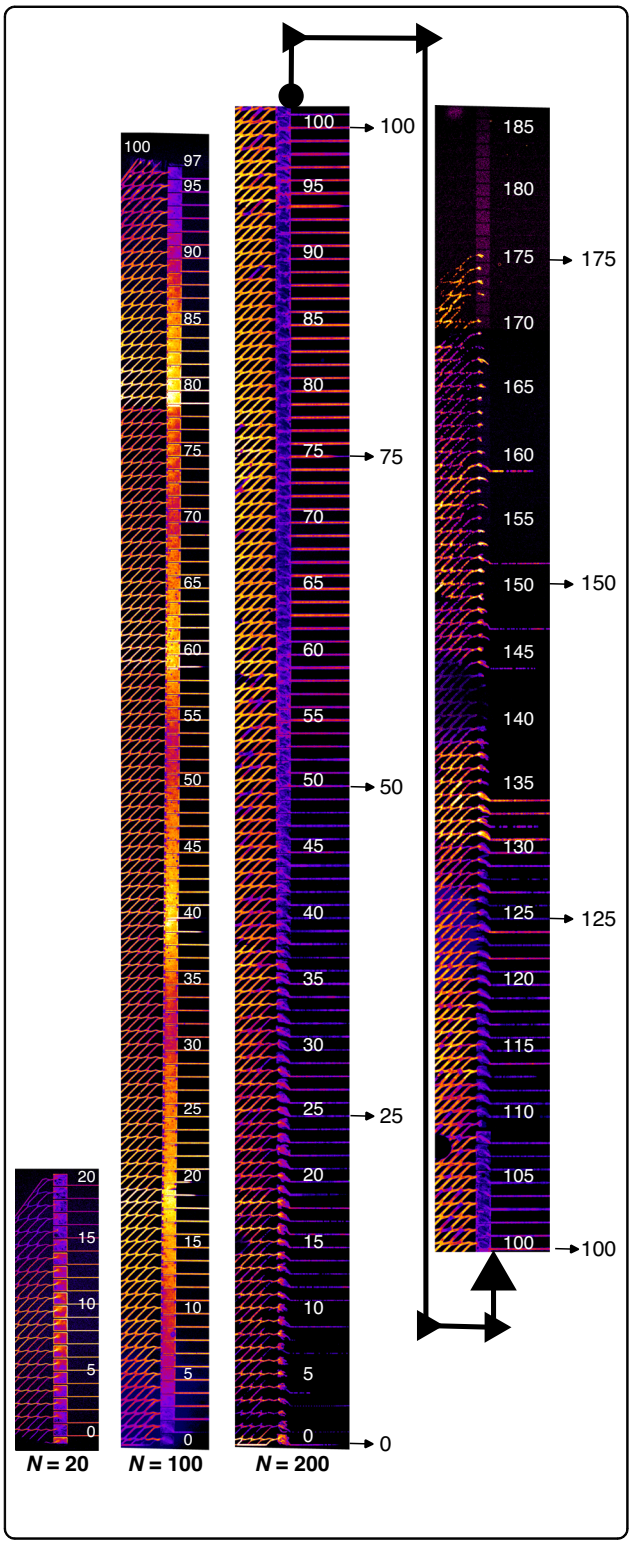


Fig. 4 Demonstration of solutions for bacteria exploring Pascal's triangle unit series for different networks, i.e., Pascal's triangle cardinalities of $N = 20$, $N = 100$, and $N = 200$. Density maps for the exits, only for cardinalities of $N = 20$, $N = 100$, and $N = 200$. For Pascal's triangle with cardinalities $N = 20$ and $N = 100$, all the exit lanes were discernible, based on the density maps representing bacterial trajectories. However, for Pascal's triangle with cardinality $N = 200$, the density map of the exits is visible only until exit #175 (due to an experimental defect, i.e., non-wetting of a region in the low downstream network, rather than bacterial crowding). Notes: 1. For $N = 200$, the density map had to be split into two sections for presentation purposes only. 2. Pascal's triangle cardinalities $N = 20$, $N = 100$, and $N = 200$, are approximately equivalent to the cardinalities of prime numbers series, $N = 6$, $N = 14$, and $N = 18$

At this experimental stage, none of the anticipated operational difficulties, i.e., depletion of nutrients, bacterial adhesion on microfluidic walls, or bacterial crowding in high densities of junctions and after long computation periods, were observed in Pascal's triangle networks before a total sum of S = 175 (equivalent to a cardinality, N, of 18 in SSP for prime numbers) (Fig. 4). Furthermore, while Pascal's triangle SSP networks encode a rather inconsequential problem, formally the problem is NP-complete. Fig. S14 shows the cardinalities (N) that were explored for solutions, by various non-conventional computing approaches, including recently studied photonics-based computing for SSP and the theoretical limits of the electronic computers based on the simulation demonstrated in this study (Fig. 6). This Figure further puts in perspective the landscape of progress in non-conventional computing using SSP as the model NP-complete problem (Fig. S14).

Scaling assessment considering the errors in junctions

We next sought to estimate the ability of the bacterial-driven computing to scale up, firstly using the present level of errors measured experimentally and in their current configuration, i.e., absent any substantial technological improvements or new problem encodings. For this purpose, we developed a stochastic spatial Monte Carlo model of device behavior (fully detailed in SI-3.3, with results presented in Movie S7, four examples of network simulations). Using this mathematical model applied for a given set of numbers making up the SSP instance, we can run simulations involving large numbers of agents to obtain statistics on device performance, e.g., histograms of the proportion of agents exiting at each exit node. The simulation also records whether an agent reached an exit by executing a legal path, or whether it made a traffic error, reflecting how statistics were gathered from the experimental setup (Fig. 5a).

We specifically simulated the performance of devices encoding SSP instances experimentally demonstrated for various values of errors, *E*, and using the number of agents from our estimates in SI-2.5, Table S4. Given a value for *E*, we can simulate the behavior of larger devices to predict whether such an error rate would allow devices to scale successfully, or alternatively, whether the 'signal' (correct

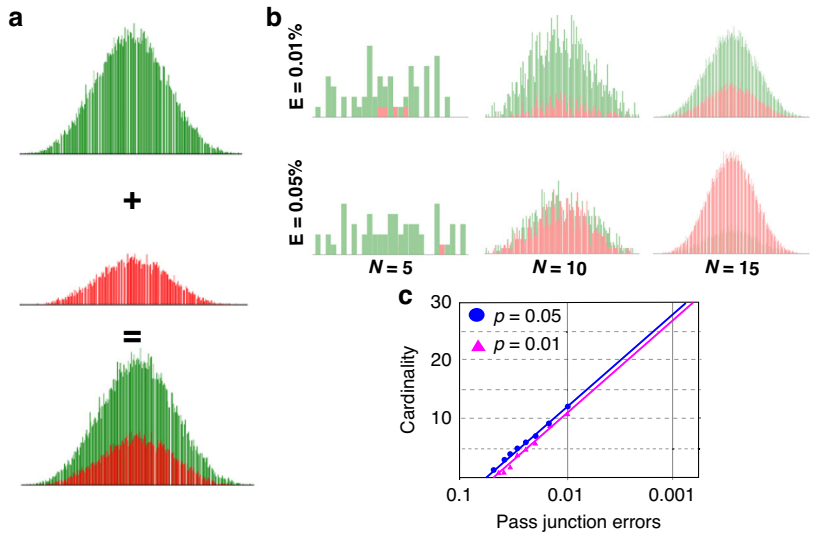


Fig. 5 A stochastic simulation model of the behavior of a biocomputational device supports the experimental results and predicts further device scalability. a The total exit counts (bottom) in both experiments and the model are generated by an additive combination of agents finding exits correctly (top) plus erroneous traffic (middle). **b** Combined exit count histograms from simulations of the model for sets with cardinality 5, 10 and 15, respectively and two representative Pass junction error rates, showing that for large problem sets, keeping junction error rate low is critical, due to the increasingly large number of agents needed to locate all possible legal exists. **c** Kolmogorov–Smirnov test - Simulation of SSP cardinality (x-axis), explorable at a given Pass junction error rate (y-axis) at two different *p*-values. *P*-value represents the difference between the distributions of agents (**a**) at the exits, exploring correctly (top, green) and agents exploring with errors (bottom, red). This is a global measure of how certain the biocomputing device can be that it has solved a given SSP instance correctly

sums being found) is buried in the 'noise' associated with agents making navigation errors at Pass junctions for SSP instances with cardinalities from 3 to 20 (SI-3.3, Movie S7 with four network simulations). This is corresponding to a problem with more than 1 million solutions, which is larger than any computation by non-electronic means[50]. The estimated cardinalities in reach of such biocomputation devices are presented in Fig. 5b, c. For a probability value of E approximating our experimental results, the signal can be clearly separated from the 'noise envelope' (with statistical certainties higher than 0.95 and 0.99, respectively) for much larger cardinalities than presently attempted experimentally. These results suggest that, notwithstanding several technological challenges[33], the computing devices demonstrated here can successfully solve problems two to three orders of magnitude larger than the original demonstration of network-based biocomputation[8] and at 1.5 orders of magnitude larger than demonstrated in this study. Furthermore, the achievable set cardinality for the two thresholds of *P*-value of <0.05 and <0.01, respectively (shown on the vertical axis in Fig. 5c), indicates that, for a Pass junction error rate of 0.01 (as reported here experimentally), sets with cardinalities of 10 to 15 can be solved, i.e., corresponding to problems with tens of thousands of solutions. To reliably solve SSP problems with cardinality 30 (and beyond), essentially error-free Pass junctions, e.g., E < 0.0001, are needed.

Computing time, volume, and energy, as a function of SSP cardinality

Various scaling estimations of the computing time by network-based biocomputing[8,33,40,47] concluded that the provision of a large enough number of agents, or better dividing agents, would eventually outrun fast, yet sequential electronic computers. The designs of the network-based biocomputer encoding an algorithm, e.g., the naïve inclusion-exclusion algorithm[31], for solving an NP-complete problem eliminates the exploration of non-solutions. Our preliminary experiments, which probed the operational limits of network computation with dividing bacteria did not reveal any blockages until cardinality 18 (for prime numbers). However, the use of an exponentially increasing computing resource, i.e., the number of agents, could eventually lead to the blockage of the channels.

To evaluate the scaling of the network computation using biological agents, our experiments were extrapolated by running stochastic simulations for *E. coli* and *V. natriegens* exploring virtual SSP networks (SI-4, Fig. S15), using experimentally measured parameters, i.e., velocity, division rate, directionality and error statistics in the junctions, and their statistical spread. The simulations were run for '0 volume' *E. coli* and *V. natriegens* for a low velocity/low division rate, i.e., velocities of 4 μm/s, and 7.5 um/s, respectively, and dividing every 30 minutes, and

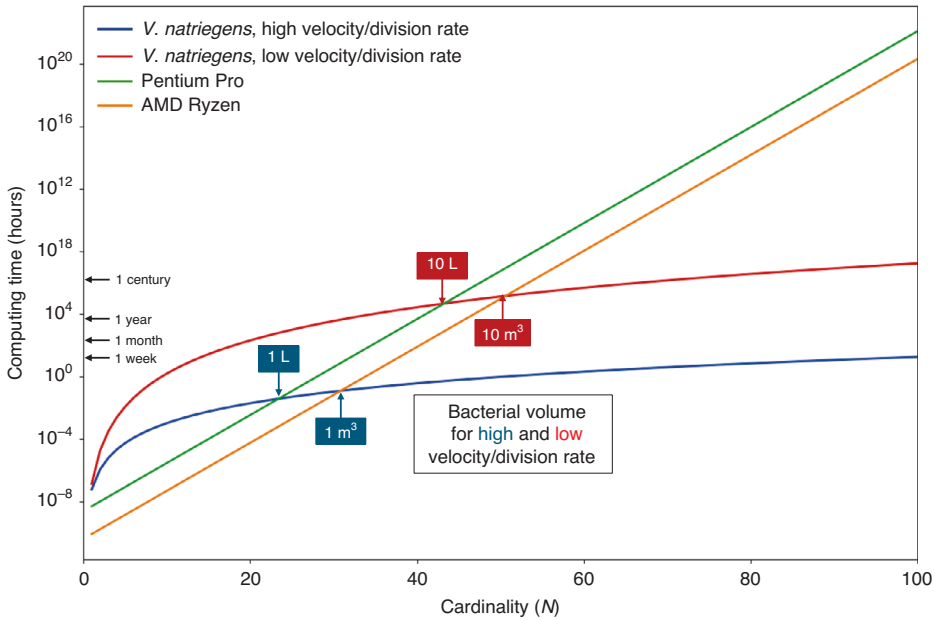


Fig. 6 Comparison of the computing time for network-based biocomputation and electronic computers. The network-based computation was performed with agents with low division rate and low velocities (*E. coli*, low temperature) and high division rate and medium velocity (*V. natriegens*, room temperature), and the benchmarked electronic computers are classical Pentium Pro and high-end AMD Ryzen Threadripper. The bacterial computer volumes for the points of estimated equal performance of the two versions of bacterial computers (high and low velocity/division rate) and electronic chips (Pentium Pro/AMD Ryzen Threadripper) are also figured

20 minutes, respectively; and for a high velocity/high division rate, i.e., velocities of 20 μm/s for both species; and dividing every 20 minutes, and 10 minutes, respectively (the error was conservatively set to 0.1%). The sequential nature of electronic computers resulted in stalling the simulations at cardinality 40 (for Pentium Pro chip). However, regression analysis ($R^2 > 0.99$) found a power law relationship between computing time of network computing with bacteria and SSP cardinality (of prime numbers). Subsequently, the simulated computing time for '0 volume' bacteria was further used to estimate the actual volume of bacteria and the energy needed for computation.

To assess the computing time required by solid-state-based computers, we also ran a 'brute force' computation solving increasingly large SSP instances with prime numbers, on various generations of computer chips, from Intel 286 to Pentium[51]. Despite using computer resources more efficiently than previously reported[8], the sequential nature of computing with electronic computers limited the size of SSP instances that could be solved to around cardinality 40 (also for Pentium Pro chip). Regression analysis using the operational parameters of emulated chips revealed an exponential relationship between computing time and cardinality ($R^2 > 0.999$), with Million Instructions Per Second (MIPS, a key performance parameter at peak performance) modulating this relationship, as described elsewhere[40]. This correlation allowed the estimation of the computing time needed to solve SSP by a high-performance chip, i.e., AMD Ryzen Threadripper 3990X (with more than 3 million MIPS).

Figure 6 presents the estimated computing time for *V. natriegens* for low velocity/low division rate, and high velocity/high division rate. Similar relationships for *E. coli* are presented in SI-4, Figs. S16 and S17 as a function of cardinality, as well as those estimated for electronic computers for solving the prime number series of SSP with inputs set of different cardinalities, from $N = 1$ to $N = 100$ (logarithmic scale). This scaling estimation suggests that network-based biocomputing, using *V. natriegens* as agents, with high velocity and division rate, will need the same time (few hours) as Pentium Pro, and AMD Ryzen, for cardinalities of approximately 28, and 38, respectively. For these intersection points, the bacterial volume required for solving the respective SSP problems is 1 L, and 1 m^3, respectively. In the more conservative scenario, i.e., low velocity and division rate, the intersection points are located at 45, and 55, respectively, where the required bacterial volume is 10 L, and 10 m^3, respectively (but the computing time is exceedingly long for all computers considered). Naturally, for 'harder' SSP, i.e., using the series of exponentially increasing numbers, the computing time for network-based biocomputing will grow exponentially due to the exponential increase in the

length of the network (and limited, constant bacterial velocity).

From the operation point of view, when the increase of the bacterial volume creates clogged regions due to insufficient volume of the local microfluidics network, the computation will switch from fast 'combinatorial', to massively parallel computation. At this point, the approximately $0.7 \cdot N^2$ (equivalent to sum(N) for prime numbers series) blockages will operate in a "Mother machine" mode[52], equivalent to the slow 'single file booting'. In this situation, the bacterial computer will operate with a 'clock frequency' in the range of 20 Hz for each of its approximately 500 separate 'cores', i.e., approximately 11 MHz for the intersection of high velocity/division rate scenario with Pentium Pro, compared to its 200 MHz.

Each bacterium will require approximately $2 \cdot 10^{-19}$ J/μm, which translates to an energy/operation for each 'biological CPU' of $2 \cdot 10^{-18}$ J/operation (for a distance of 10 μm between logical gates, and ten times more for 100 μm). On the electronic computers side, a recent estimation[53] of the energy requirements of present CPUs at $3.9 \cdot 10^{-12}$ to $1.4 \cdot 10^{-11}$ J/operation (Fig. S18). The actual consumption of energy, both by biological and by electronic computers, is considerably higher, as, for instance, *E. coli* uses only approximately 2% of its energy for motility. Also, electronic computers need more energy to operate beyond the actual CPU requirements.

Perspectives

The proof of concept of network-based biocomputing with motile and dividing bacterial agents suggests that, even in the absence of technological improvements, this computational paradigm is a good candidate for further development as an alternative to sequential electronic computers, which have benefited from seven decades of advances in hardware and software. To reach a higher potential, network-based biocomputing with motile and dividing agents could also benefit from improvements, many of which, at the present level of technology development, were reviewed recently[33,47].

Microfluidics network

One possible 'hardware'-based improvement would replace the static Split- and Pass-junctions with a type of dynamic junction, which will allow programming the blockage of the two lanes running alongside the central crossing, transforming a Split Junction into a Pass Junction. Consequently, the logic of any SSP series of a given total size can be programmed, possibly in real-time, and on a large scale[54]. The control of the opening of the junctions would also allow the implementation of 'traffic control', which could prevent crowding or push it further to higher cardinalities. Additionally, while the present level of errors allows, in principle, finding solutions for much larger problems than demonstrated here, their elimination would have considerable benefits, e.g., on the number of agents, computing and readout time. The planar Pass Junction could be replaced by 3D, highway-like junctions, to achieve error-free computing[55].

On a more strategic direction, network-based biocomputing will require other encoding schemes for other non-trivial mathematical problems challenging sequential computers. Also, the mitigation of the exponential increase of the volume of the dividing agents can be addressed by channels with increasing depths in areas where bacterial multiplication reaches high levels, as proposed elsewhere[41]. A more strategic approach is to translate the microfluidic network from the present planar geometry to a folded-planes, volume-based architecture. This new architecture will require, by necessity, microarrayed optical sensors. Finally, microarray digitization could include local heating/cooling elements to control the velocity and division rate of bacteria in zones where crowding reaches dangerous limits.

Agents

Network-based biocomputing would benefit greatly if each agent would not only perform as a CPU solving simple operations in logic junctions, but also as a memory registering these operations 'on the fly'[33,56]. Fortunately, and in contrast to other proposed agents, such as photons[11,13], or even cytoskeletal filaments[8], there is a large panoply of techniques available for the engineering of bacteria at the protein, DNA or RNA levels, or by modification on the cell surface with high-affinity labelling, including those that can be turned on and off by external light beams[57], thus making bacteria amenable to recording trajectory history. Finally, a more technologically achievable improvement consists of finding, or engineering bacteria with properties leading to a decrease in crowding, i.e., optimal division rate, but also high velocity, as demonstrated in the scaling analysis of computing time. Overall, we demonstrated here the proof of concept of a biocomputer using the exploration of microfluidic networks encoding mathematical problems by self-propelled, dividing bacteria. Using SSP—a classical NP-complete problem—as a benchmark test, the estimation of the computational performance showed that, if sufficiently scaled up, the network-based biocomputation could compete with the present electronic computers solving a non-trivial problem.

Materials and methods

Fabrication of SSP networks

The fabrication of the masters for the PDMS SSP network used e-beam lithography followed by RIE etching. A

flowchart is presented schematically in Fig. S1, and detailed information about the fabrication process, including the replication of the master by PDMS, is presented in Materials and Methods in supplementary text, SI MM-1.

Bacterial agents

Two bacterial species were used in this work, namely, *E. coli* HCB437 and *V. natriegens* (with characterization presented in Table S1). Details regarding bacterial culture and their use for network-biocomputation experiments are presented in Materials and Methods in the supplementary text, SI MM-2.

Image acquisition and analysis

All experiments were performed on a PDMS device mounted on an inverted Olympus IX83 Confocal microscope, using a Hamamatsu C11440 monochromatic digital camera. The MetaMorph® Microscopy Automation and Image Analysis Software (Molecular Devices) was used as an interface between the microscope and the computer for image acquisition and initial screening. Full details regarding image acquisition and analysis, including the post-processing of images and bacterial count, and post-processing for generating density maps, tracking trajectories (Fig. S4, Table S2), post-processing for quantification of bacterial multiplication in the network, is presented in Materials and Methods in the Supplementary text, SI MM-3.

Simulation and emulation of bacterial computation vs electronic computation

The simulation of the SSP device operation was developed in Scala on the IntelliJ IDEA 2018.3.2 development environment. The simulated data shown in Fig. 6 was generated on a Lenovo laptop with Microsoft Windows 10 OS and Intel(R) Processor Core i5-7200U CPU @ 2.50 GHz (2 Cores, 4 Logical Processors). Full details of the modelling and simulation are presented in the Materials and Methods section of the Supplementary text, SI MM-4.

Error analysis

If an agent is at a junction corresponding to a Pass junction, it will choose its next junction along its previous direction of travel (horizontal or diagonal) with probability 1-E, and it will deviate to the alternative direction of travel with probability E. If an agent reaches an exit node, the simulation terminates for that agent, and the program keeps a record of that exit having been reached. Full details are presented in Materials and Methods in the Supplementary text, SI MM-5.

Acknowledgements

We thank the Facility for Electron Microscopy Research (FEMR) at McGill University, the Faculty of Dental Medicine at the University of Montreal, and the Department of Geology at the University of Quebec at Montreal for help in electron microscopy characterization of bacteria. We also thank Prof. Howard C. Berg, MIT, USA. and Dr. Ankur B. Dalia, Indiana University, USA, for gifting the smooth swimming *Escherichia coli* HCB437 and *Vibrio natriegens* SAD1306, respectively. Last, but not least, we thank young Isabella-Maria Barbu for lending her voice to the explanatory animation.

Author details

[1]Department of Bioengineering, McGill University, Faculty of Engineering, Montreal, QC, Canada. [2]Molecular Sense Ltd., Liverpool, UK. [3]Department of Chemical Engineering, University of Chemistry and Technology, Prague, Czech Republic. [4]Kavli Institute of Nanoscience Delft, Delft University of Technology, Delft, The Netherlands. [5]School of Mathematical Sciences, Queensland University of Technology, Brisbane, QLD, Australia. [6]Peter Gorer Department of Immunobiology, School of Immunology and Microbial Sciences, Faculty of Life Sciences and Medicine, King's College London, London, UK

Author contributions

Ayyappasamy Sudalaiyadum Perumal: conceptualization, methodology, validation, formal analysis, investigation, writing—original draft, writing—review and editing, visualization; Falco C.M.J.M. van Delft conceptualization, methodology, validation, formal analysis, investigation, writing—original draft, writing—review and editing, visualization; Giulia Ippoliti: methodology, software, formal analysis, investigation, writing—review and editing, visualization; Ondřej Kašpar: methodology, validation, software, formal analysis, investigation, visualization; Viola Tokárová: methodology, validation, formal analysis, investigation, visualization; Jessica Li: methodology, validation, formal analysis, investigation, visualization; Monalisha Nayak: methodology, validation, formal analysis, investigation, visualization; Matthew Cho: formal analysis, investigation; Anja van Langen-Suurling: methodology, validation, investigation, visualization; Charles de Boer: methodology, validation, investigation, visualization; Frank Dirne: methodology, validation, investigation Dan V. Nicolau Jr. conceptualization, methodology, validation, formal analysis, investigation, writing—original draft, writing—review and editing, visualization, funding acquisition; Dan V. Nicolau conceptualization, methodology, validation, formal analysis, investigation, writing—original draft, writing—review and editing, visualization, supervision, project administration, funding acquisition.

Funding

Research supported by the Defense Advanced Research Projects Agency (DARPA), grant no. HR0011-16-2-0028; by the Canadian Natural Sciences and Engineering Research Council (NSERC) grant no. RGPIN-2016-05019; by Social Sciences and Humanities Research Council of Canada (SSHRC), grant NFRFE-2019-00129, by an Australian Future Fellowship (for DVN Jr); and by the European Union, grant no. 732482 (Parallel network-based biocomputation: technological baseline, scale-up and innovation ecosystem).

Data availability

All data are available in the main text or the supplementary materials.

Conflict of interest

The authors declare no competing interests.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41378-026-01341-x.

Received: 17 November 2025 Revised: 27 January 2026 Accepted: 2 April 2026
Published online: 11 August 2026

References

1. Hopfield, J. J. & Tank, D. W. Neural" computation of decisions in optimization problems. *Biol Cyber* **52**, 141–152 (1985).

2. Comon-Lundh, H., Cortier, V., Zălinescu, E. Deciding security properties for cryptographic protocols. Application to key cycles. *ACM Transactions on Computational Logic* **11**, (2010).
3. Nam, G. J., Sakallah, K. A. & Rutenbar, R. A. A new FPGA detailed routing approach via search-based Boolean satisfiability. *IEEE Trans Computer-Aided Des Integr Circuits Syst* **21**, 674–684 (2002).
4. Pierce, N. A. & Winfree, E. Protein design is NP-hard. *Protein Eng* **15**, 779–782 (2002).
5. Blelloch, G. E., Maggs, B. M. Parallel algorithms. In: *Algorithms and theory of computation handbook: special topics and techniques*) (2010).
6. Garey, M. R., Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co. (1979).
7. Adleman, L. M. Molecular computation of solutions to combinatorial problems. *Science* **266**, 1021–1024 (1994).
8. Nicolau, D. V. Jr et al. Parallel computation with molecular-motor-propelled agents in nanofabricated networks. *Proc Natl Acad Sci* **113**, 2591–2596 (2016).
9. Petersen, W., Arbenz, P., Petersen, W., Arbenz, P. 1Basic Issues. In: *Introduction to Parallel Computing: A practical guide with examples in C*). Oxford University Press (2004).
10. Nicolau, D.V. Jr, Burrage, K., Nicolau, D. V. Computing with motile bio-agents. In: *Biomedical Applications of Micro-and Nanoengineering III*). International Society for Optics and Photonics (2006).
11. Oltean, M. & Muntean, O. Solving the subset-sum problem with a light-based device. *Nat Comput* **8**, 321–331 (2009).
12. Nicolau, D. V. et al. Molecular motors-based micro- and nano-biocomputation devices. *Microelectron Eng* **83**, 1582–1588 (2006).
13. Xu, X.-Y. et al. A scalable photonic computer solving the subset sum problem. *Sci Adv* **6**, eaay5853 (2020).
14. Karp, R. M. Reducibility among combinatorial problems. In: *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*) (2010).
15. Song, B., Song, T. & Pan, L. A time-free uniform solution to subset sum problem by tissue P systems with cell division. *Math Struct Computer Sci* **27**, 17–32 (2017).
16. Karp, R. M. Reducibility among combinatorial problems. In: *Complexity of computer computations*). Springer (1972).
17. Filar, J. A., Haythorpe, M. & Taylor, R. Linearly-growing reductions of Karp's 21 NP-complete problems. *Numer Algebra, Control Optim* **8**, 1–16 (2018).
18. Mertens, S. Phase Transition in the Number Partitioning Problem. *Phys Rev Lett* **81**, 4281–4284 (1998).
19. Lenstra, J. K., Kan, A. R., Brucker, P. Complexity of machine scheduling problems. In: *Annals of discrete mathematics*). Elsevier (1977).
20. Costa, A. M., Cordeau, J. F. & Laporte, G. Steiner tree problems with profits. *INFOR* **44**, 99–115 (2006).
21. Chang, W.-L. & Guo, M. Solving the set cover problem and the problem of exact cover by 3-sets in the Adleman–Lipton model. *J Biosyst* **72**, 263–275 (2003).
22. Impagliazzo, R. & Paturi, R. On the complexity of k-SAT. *J Computer Syst Sci* **62**, 367–375 (2001).
23. Henkel, C. V., Bäck, T., Kok, J. N., Rozenberg, G. & Spaink, H. P. DNA computing of solutions to knapsack problems. *BioSystems* **88**, 156–162 (2007).
24. Kate, A. & Goldberg, I. Generalizing cryptosystems based on the subset sum problem. *Int J Inf Security* **10**, 189–199 (2011).
25. Caprara, A., Kellerer, H. & Pferschy, U. The multiple subset sum problem. *SIAM J Optim* **11**, 308–319 (2000).
26. Coster, M. J. et al. Improved low-density subset sum algorithms. *Computational Complex* **2**, 111–128 (1992).
27. Horowitz, E. & Sahni, S. Computing partitions with applications to the knapsack problem. *J ACM (JACM)* **21**, 277–292 (1974).
28. Schroeppel, R. & Shamir, A. A T=O(2^n/2), S=O(2^n/4) algorithm for certain NP-complete problems. *SIAM J Comput* **10**, 456–464 (1981).
29. Howgrave-Graham, N., Joux, A. New generic algorithms for hard knapsacks. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*). Springer (2010).
30. Pisinger, D. An exact algorithm for large multiple knapsack problems. *Eur J Operational Res* **114**, 528–541 (1999).
31. Björklund, A., Husfeldt, T. & Koivisto, M. Set partitioning via inclusion-exclusion. *SIAM J Comput* **39**, 546–563 (2009).
32. Saketh, K. H., Jeyakumar, G. Comparison of dynamic programming and genetic algorithm approaches for solving subset sum problems. In: *Advances in Intelligent Systems and Computing*) (2020).
33. Van Delft, F. C. et al. Something has to give: scaling combinatorial computing by biological agents exploring physical networks encoding NP-complete problems. *J. Interface focus*, 1-19 (2018).
34. Van Delft, F. C. M. J. M. et al. Design and fabrication of networks for bacterial computing. *N. J. Phys.* **23**, (2021).
35. Tokárová, V. et al. Patterns of bacterial motility in microfluidics-confining environments. *Proc. Natl. Acad. Sci. USA* **118**, (2021).
36. DiLuzio, W. R. et al. Escherichia coli swim on the right-hand side. *Nature* **435**, 1271 (2005).
37. Darnton, N. C., Turner, L., Rojevsky, S. & Berg, H. C. On torque and tumbling in swimming Escherichia coli. *J Bacteriol* **189**, 1756–1764 (2007).
38. Wolfe, A. J., Conley, M. P., Kramer, T. J. & Berg, H. C. Reconstitution of signaling in bacterial chemotaxis. *J Bacteriol* **169**, 1878–1885 (1987).
39. Weinstock, M. T., Hesek, E. D., Wilson, C. M. & Gibson, D. G. Vibrio natriegens as a fast-growing host for molecular biology. *Nat methods* **13**, 849–851 (2016).
40. Sudalaiyadum Perumal, A. et al. As good as it gets: a scaling comparison of DNA computing, network biocomputing, and electronic computing approaches to an NP-complete problem. *N J Phys* **23**, 125001 (2021).
41. Männik, J., Driessen, R., Galajda, P., Keymer, J. E. & Dekker, C. Bacterial growth and motility in sub-micron constrictions. *Proc Natl Acad Sci* **106**, 14861–14866 (2009).
42. Aiyar, S. E., Gaal, T. & Gourse, R. L. rRNA promoter activity in the fast-growing bacterium Vibrio natriegens. *J Bacteriol* **184**, 1349–1358 (2002).
43. Dalia, T. N. et al. Multiplex genome editing by natural transformation (MuGENT) for synthetic biology in Vibrio natriegens. *J ACS Synth Biol* **6**, 1650–1655 (2017).
44. Allen, R. D. & Baumann, P. Structure and arrangement of flagella in species of the genus Beneckea and Photobacterium fischeri. *J Bacteriol* **107**, 295–302 (1971).
45. Wang, P. et al. Robust growth of Escherichia coli. *Curr Biol* **20**, 1099–1103 (2010).
46. Den Blaauwen, T., Buddelmeijer, N., Aarsman, M. E., Hameete, C. M. & Nanninga, N. Timing of FtsZ assembly in Escherichia coli. *J Bacteriol* **181**, 5167–5175 (1999).
47. Van Delft F. C. M. J. M. et al. Roadmap for network-based biocomputation. *Nano Futures* **6**, (2022).
48. Kempes, C.P. et al. Drivers of bacterial maintenance and minimal energy requirements. *Front. Microbiol.* 31 (2017).
49. Roman, H. N., Juncker, D. & Lauzon, A. M. A microfluidic chamber to study the dynamics of muscle-contraction-specific molecular interactions. *Anal Chem* **87**, 2582–2587 (2015).
50. Braich, R. S., Chelyapov, N., Johnson, C., Rothemund, P. W. & Adleman, L. Solution of a 20-variable 3-SAT problem on a DNA computer. *Science* **296**, 499–502 (2002).
51. Van Delft, F. C. et al. Something has to give: scaling combinatorial computing by biological agents exploring physical networks encoding NP-complete problems. *J. Interface Focus*, (2018).
52. Ollion, J., Elez, M. & Robert, L. High-throughput detection and tracking of cells and intracellular spots in mother machine experiments. *Nat Protoc* **14**, 3144–3161 (2019).
53. Shankar, S. Energy Estimates Across Layers of Computing: From Devices to Large-Scale Applications in Machine Learning for Natural Language Processing, Scientific Computing, and Cryptocurrency Mining. In: *2023 IEEE High Performance Extreme Computing Conference (HPEC)*). IEEE (2023).
54. Unger, M. A., Chou, H. P., Thorsen, T., Scherer, A. & Quake, S. R. Monolithic microfabricated valves and pumps by multilayer soft lithography. *Science* **288**, 113–116 (2000).
55. Chiu, D. T., Pezzoli, E., Wu, H., Stroock, A. D. & Whitesides, G. M. Using three-dimensional microfluidic networks for solving computationally hard problems. *Proc Natl Acad Sci* **98**, 2961–2966 (2001).
56. Martins, D. P. et al. Microfluidic-Based Bacterial Molecular Computing on a Chip. *IEEE Sens J* **22**, 16772–16784 (2022).
57. Olesińska-Mönch, M. & Deo, C. Small-molecule photoswitches for fluorescence bioimaging: engineering and applications. *Chem Commun* **59**, 660–669 (2023).