COLLECTIVE BEHAVIOR

Autonomous task sequencing in a robot swarm

Lorenzo Garattoni and Mauro Birattari*

Robot swarms mimic natural systems in which collective abilities emerge from the interaction of individuals. So far, the swarm robotics literature has focused on the emergence of mechanical abilities (e.g., push a heavy object) and simple cognitive abilities (e.g., select a path between two alternatives). In this article, we present a robot swarm in which a complex cognitive ability emerged. This swarm was able to collectively sequence tasks whose order of execution was a priori unknown. Because sequencing tasks is an albeit simple form of planning, the robot swarm that we present provides a different perspective on a pivotal debate in the history of artificial intelligence: the debate on planning in robotics. In the proposed swarm, the two robotics paradigms—deliberative (sense-model-plan-act) and reactive (sense-act)—that are traditionally considered antithetical coexist in a particular way: The ability to plan emerges at the collective level from the interaction of reactive individuals.

INTRODUCTION

Swarm robotics (*1*–*4*) takes inspiration from collective behaviors of social animals to develop multirobot systems that, like their natural counterparts, are flexible, robust, and autonomous (*5*). A robot swarm comprises a large number of robots with limited capabilities. The interaction of the robots with each other and with the environment engenders emergent properties: Collectively, the swarm displays abilities that a single robot does not have. So far, research has focused on the emergence of geometrical/spatial properties and mechanical abilities: for example, aggregating (*6*), covering space (*7*, *8*), forming shapes (*9*, *10*), moving coordinately (*11*), overcoming obstacles (*12*), transporting objects (*13*), clustering objects (*14*), or assembling structures (*15*). Research has been also devoted to the emergence of simple cognitive abilities: for example, selecting an aggregation area (*16*–*18*), a behavior (*19*, *20*), a foraging source (*21*, *22*), or a path (*23*–*26*) between (typically two) alternatives. Here, we consider the emergence of a more complex cognitive ability: sequencing tasks. We present TS-Swarm, a robot swarm that sequences tasks autonomously. Several studies have already been devoted to swarms that, inspired by mechanisms of division of labor observed in insect societies (*27*–*29*), perform multiple tasks, transitioning from one to another (*8*, *30*–*32*). Nonetheless, in these previous studies, the correct order of execution and/or the transition conditions were known at design time. The designers could thus devise and hard-code in the robots the rules that trigger the transition from task to task. Contrary to previously demonstrated swarms, TS-Swarm sequences tasks autonomously and at run time. It can therefore operate even if the correct order of execution is unknown at design time. In TS-Swarm, the two robotics paradigms—deliberative (sense-model-plan-act) (*33*) and reactive (sense-act) (*34*)—that are traditionally considered antithetical (*35*) coexist: The ability to sequence tasks and, therefore, to plan a course of action emerges at the collective level from the interaction of reactive individuals.

Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle, Université libre de Bruxelles, Belgium.
*Corresponding author. Email: mbiro@ulb.ac.be

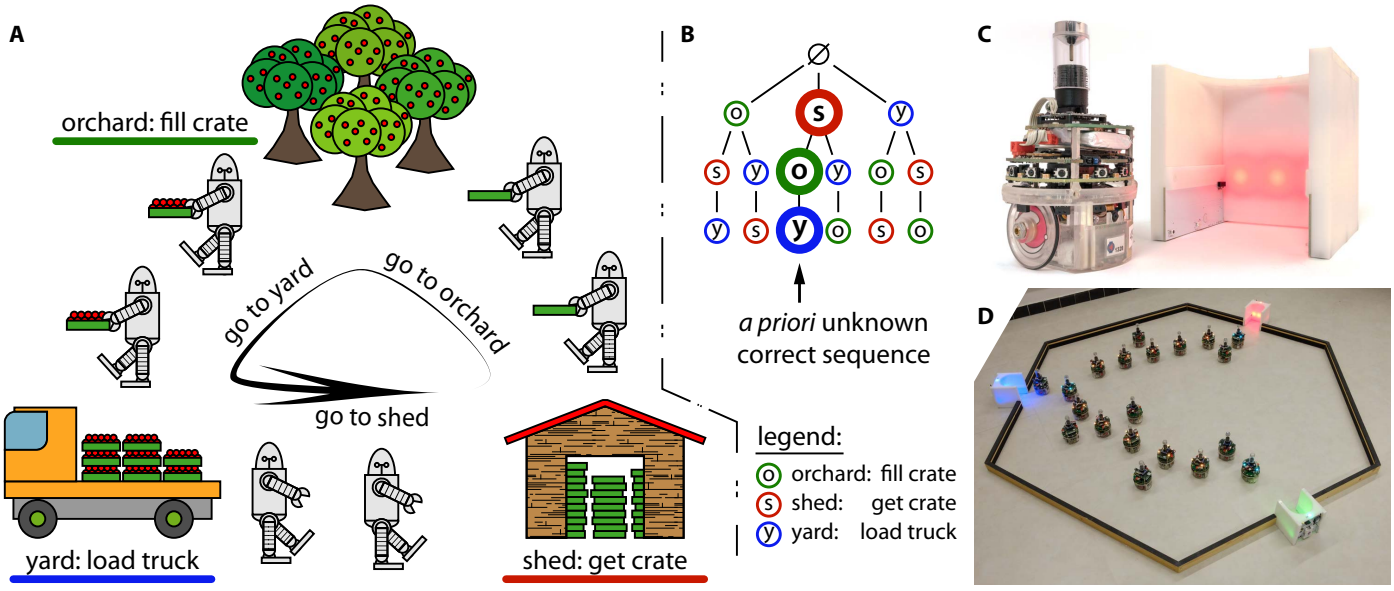

Fig. 1. From task sequencing to TS-Swarm. (**A**) An example of task sequencing. Three tasks must be performed in a specific order by an individual robot: Get a crate at the shed, fill the crate at the orchard, and load the crate onto the truck at the yard. The robots initially ignore the correct order of execution. They learn collectively from successes and failures; for example, a robot faces a failure if it reaches the orchard without a crate to fill or the truck with an empty one. The correct sequence must be repeated multiple times to fully load the truck. (**B**) Formal representation of the solution space. (**C**) An e-puck and a TAM. (**D**) TS-Swarm in its arena with three TAMs.

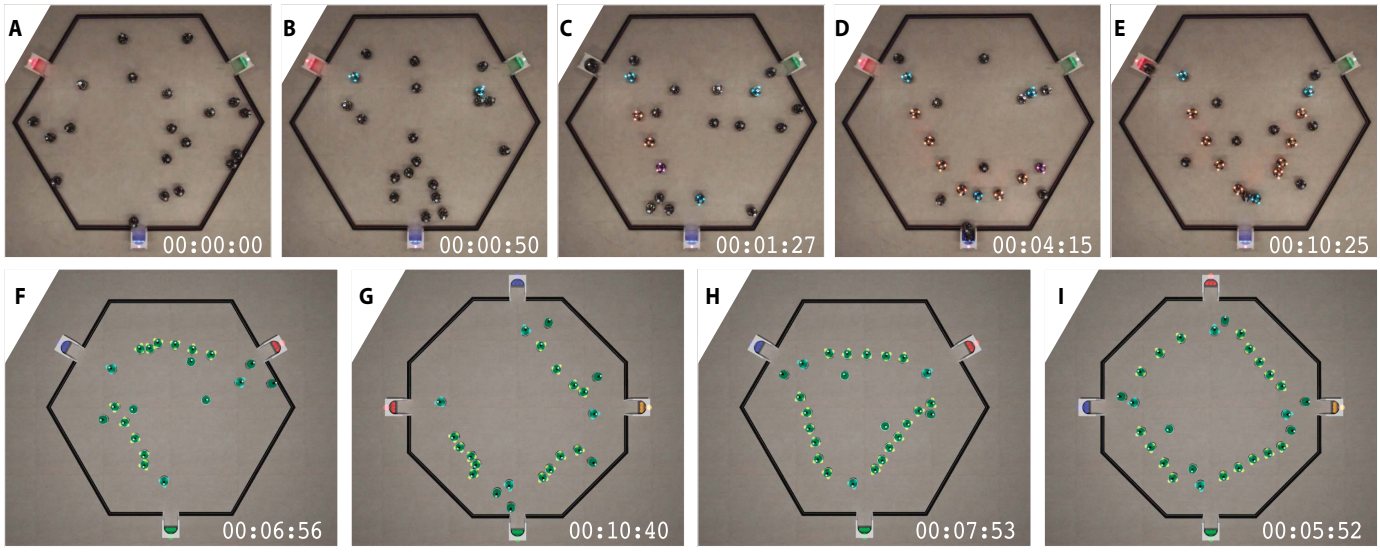

Fig. 2. Overhead snapshots. (**A** to **E**) Mark I_3, robot experiments (movie S1). (**F**) Mark I_3, simulation (movie S2, side by side with a run on the robots). (**G**) Mark I_4, simulation (movie S4). (**H**) Mark II_3, simulation (movie S5). (**I**) Mark II_4, simulation (movie S6).

Table 1. Parameters of the scalability and robustness studies. We report the parameters that characterize each experimental setting in which each variant of TS-Swarm was studied. The scalability study was performed using the default number of robots in each setting. Between one setting and the following one, we doubled the surface of the arena in which the robots operate (see Materials and Methods). The robustness study was performed while varying the number of robots between −20 and +100% with respect to the default number of each setting.

Setting		Arena side (m)	Arena area (m^2)	Number of robots								Time cap (s)
				−20%	−10%	Default	+20%	+40%	+60%	+80%	+100%	
Mark I_3	0	0.90	2.10	16	18	**20**	24	28	32	36	40	2400
	1	1.27	4.21	23	25	**28**	34	39	45	50	56	3400
	2	1.80	8.42	32	36	**40**	48	56	64	72	80	4800
	3	2.55	16.84	45	51	**57**	68	80	91	103	114	6800
	4	3.60	33.67	64	72	**80**	96	112	128	144	160	9600
Mark I_4	0	0.66	2.10	18	20	**22**	26	31	35	40	44	100,000
	1	0.93	4.21	25	28	**31**	37	43	50	56	62	100,000
	2	1.32	8.42	35	40	**44**	53	62	70	79	88	100,000
	3	1.87	16.84	50	56	**62**	74	87	99	112	124	100,000
	4	2.64	33.67	70	79	**88**	106	123	141	158	176	100,000
Mark II_3	0	0.90	2.10	20	22	**25**	30	35	40	45	50	100,000
	1	1.27	4.21	28	31	**35**	42	49	56	63	70	100,000
	2	1.80	8.42	40	45	**50**	60	70	80	90	100	100,000
	3	2.55	16.84	56	63	**70**	84	98	112	126	140	100,000
	4	3.60	33.67	80	90	**100**	120	140	160	180	200	100,000
Mark II_4	0	0.66	2.10	22	24	**27**	32	38	43	49	54	100,000
	1	0.93	4.21	30	34	**38**	46	53	61	68	76	100,000
	2	1.32	8.42	43	49	**54**	65	76	86	97	108	100,000
	3	1.87	16.84	61	68	**76**	91	106	122	137	152	100,000
	4	2.64	33.67	86	97	**108**	130	151	173	194	216	100,000

We addressed the case in which m tasks must be performed in a specific order (and without repetitions) by an individual robot of the swarm. Each task must be performed in a certain area, and the correct order is a priori unknown. The sequence of tasks must be repeated multiple times by the same or by other robots. For an illustrative example, see Fig. 1 (A and B).

The characterizing feature of TS-Swarm is that some of the robots position themselves to form a chain that fulfills two functions: (i) to assist the navigation between the relevant areas and (ii) to identify/encode the order in which tasks must be performed. The chain enables robots with limited capabilities to accomplish a complex mission. Individually, the robots of TS-Swarm would be unable to navigate reliably from area to area or to perform the tasks in the correct order. They have a limited range of perception, are unaware of the position of the areas, and are unable to localize themselves in the environment. Moreover, the robots are not programmed to individually sequence tasks by reasoning symbolically on their order of execution.

Chaining has previously been adopted in swarm robotics to search the environment and to assist navigation (*31*, *36*–*43*). By forming a chain, robots act as waypoints to route other robots. In TS-Swarm, we generalized this scenario. The chain is both a routing mechanism and a means to identify/encode a task sequence. The robots in the chain act also as "logical waypoints" in the task space. By following them, other robots perform the tasks in the order encoded.

RESULTS

We implemented TS-Swarm on e-puck robots, and we used TAMs (task abstraction modules) to abstract tasks (see Fig. 1C and Materials and Methods). A TAM is a booth, which an e-puck can enter. For an e-puck, entering a TAM amounts to performing the task that it abstracts. A TAM is equipped with red-green-blue (RGB) light-emitting diodes (LEDs) and can display different colors. In the experiments, each task was identified by a unique color.

We developed four variants of TS-Swarm: Mark I_3, Mark I_4, Mark II_3, and Mark II_4. Mark I_3 assumes that (i) the tasks to be performed are $m = 3$ and (ii) a robot receives negative feedback as soon as it performs a task in an incorrect order and positive feedback otherwise. A robot receives feedback in the sense that, after performing a task, it becomes immediately aware of whether the task was performed in the correct order or not (see example in Fig. 1A). In practice, as we consider abstract tasks emulated by TAMs, a failure or a success in performing a task amounts to a message that a robot receives from a TAM via an infrared (IR) signal (see Materials and Methods).

After studying Mark I_3, we modified the aforementioned assumptions to make the sequencing problem harder. Mark I_4 assumes that the tasks are $m = 4$ and indicates how a larger number of tasks can be handled. Mark II_3 assumes that a robot must perform a complete sequence before receiving any feedback on whether the sequence is correct. Mark II_4 assumes that the tasks are $m = 4$, and a complete sequence must be performed before receiving any feedback. Because of the lack of an immediate feedback, the problem faced by Mark II_m is combinatorial. Its computational complexity is $O(m!)$. In all four variants, the swarm operates in a bounded, convex arena surrounded by walls. The TAMs are located at the boundaries of the arena (see Fig. 1D and Materials and Methods). We opted for such a scenario to simplify the construction of the chain so that we could focus our attention on the collective and distributed solution of the task-sequencing problem. The adoption of this scenario enabled us to implement the chain-based search in a way that presents only minor differences from what has already been described in the literature (*31*, *39*, *43*). In the following, we outline Mark I_3. We then sketch Mark I_4, Mark II_3, and Mark II_4 by highlighting their differences with respect to Mark I_3. Details are provided in the Supplementary Materials together with an extensive discussion of the empirical analysis.

Mark I_3 and Mark I_4

In Mark I_3, all robots execute the same control software but autonomously assume different roles depending on the contingencies that they encounter. A robot can be a runner, guardian, tail, or link. Initially, all robots are runners and move randomly in the arena. Upon encountering a task—more precisely, the TAM that abstracts it—a runner performs it and then remains in its proximity, becoming its guardian. From then on, no other runner will perform the task, unless directed to do so by its guardian. Eventually, three robots are the guardians of the three tasks. Two of them received negative feedback because their

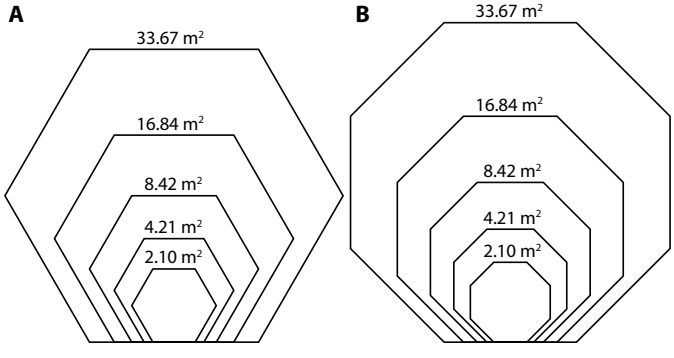

Fig. 3. Scalability and robustness analysis and the arenas. Shape and size of the arenas considered for the scalability and robustness study of (**A**) Mark I_3 (movie S3) and Mark II_3 and (**B**) Mark I_4 and Mark II_4.

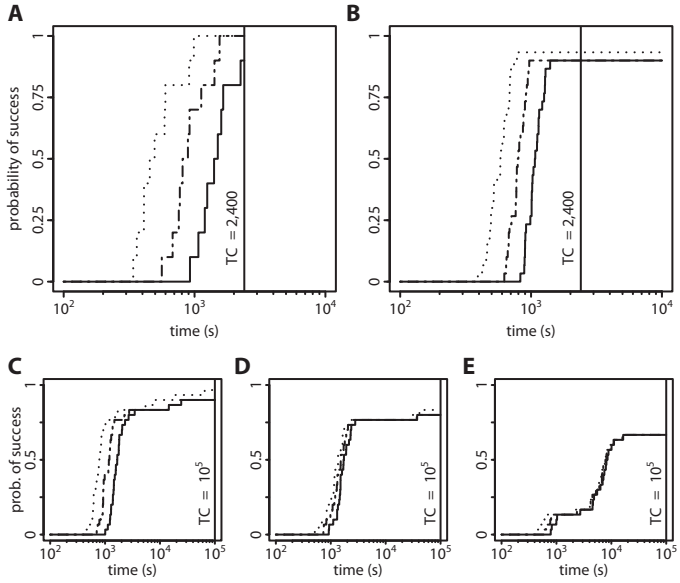

Fig. 4. Empirical assessments. Empirical run-time distributions for the execution of 1 (dotted lines), 5 (dot-dash lines), and 10 (solid lines) sequences. (**A**) Mark I_3, robot experiments. (**B**) Mark I_3, simulation. (**C**) Mark I_4, simulation. (**D**) Mark II_3, simulation. (**E**) Mark II_4, simulation.

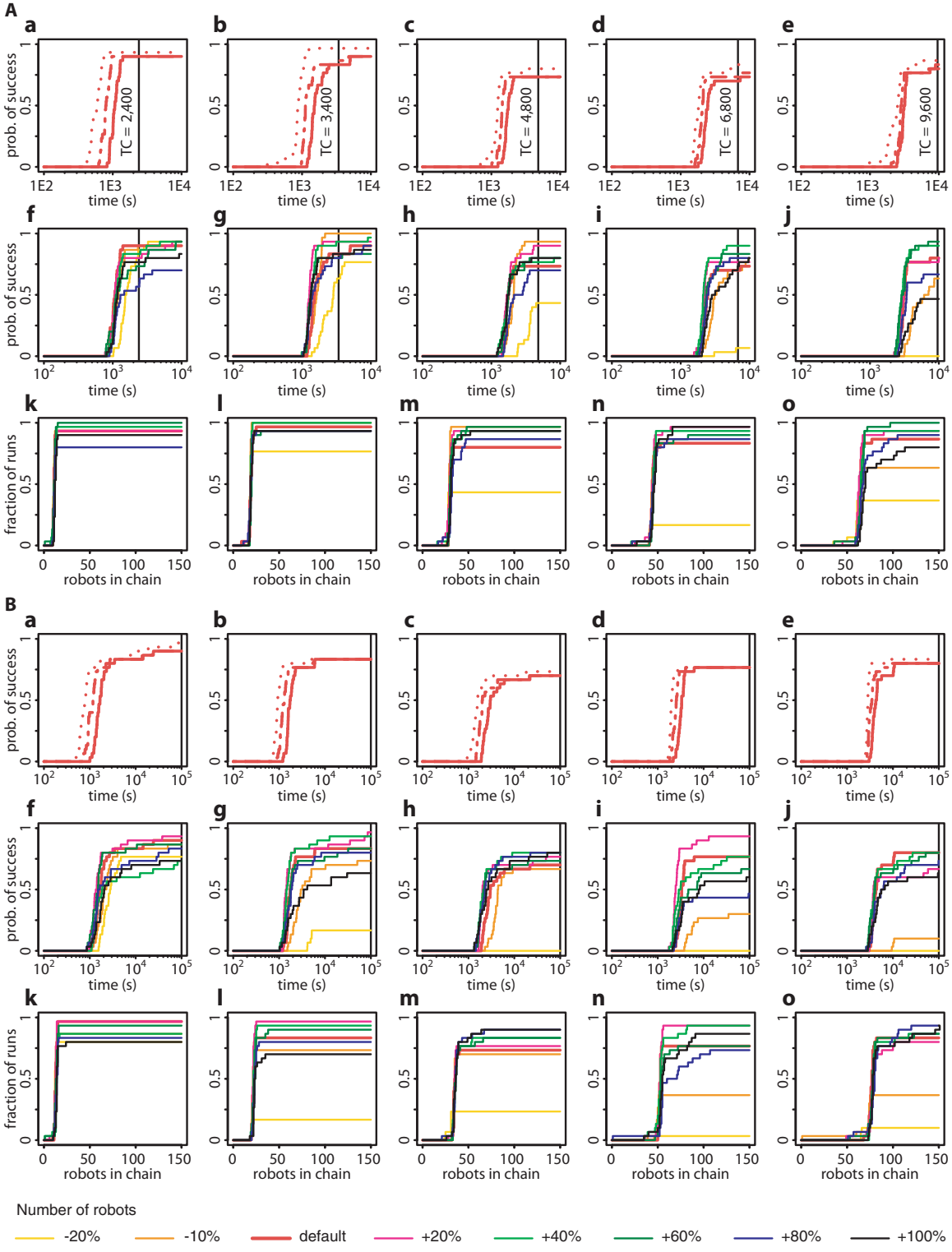

Fig. 5. Scalability and robustness of Mark I_3 and Mark I_4. (**A**) Mark I_3. (**B**) Mark I_4. (a to e) Scalability studies using the default number of robots in five arenas of different size (see Table 1 and Materials and Methods). Empirical run-time distributions for the execution of 1 (dotted lines), 5 (dot-dash lines), and 10 (solid lines) sequences. (f to j) Robustness to variation in the number of robots between −20 and +100% of the default number (see Table 1 and Materials and Methods). Empirical run-time distributions for the execution of 10 sequences. (k to o) Empirical distributions of the number of robots in the chain as a function of the total number of robots. Arena areas: 2.10 m^2 (a, f, and k), 4.21 m^2 (b, g, and l), 8.42 m^2 (c, h, and m), 16.84 m^2 (d, i, and n), and 33.67 m^2 (e, j, and o).

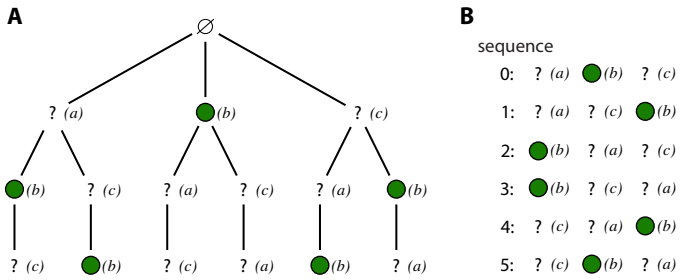

Fig. 6. Exploration of the sequence space in Mark II_3, as seen by the guardian of the green task. In this example, the green task is the second of the initial sequence. Its guardian ignores the colors of the first and last tasks; it only knows that its label is b, and therefore, its task is second. More precisely, this guardian is in the state in which it directs to its task the runners that have performed exactly one task. This guardian (as the others) directs the runners throughout the search process without knowing the sequence that is tested at each step. At the first step, its task is second—it directs to its task the runners that have performed exactly one task. At the following step, its task is third—it directs to its task the runners that have performed exactly two tasks. Then, its task is first—it directs to its task runners that have not yet performed any task and so on. (**A**) Permutation tree generated by the guardian of the green task on the basis of its partial knowledge of the initial sequence. (**B**) Sequences explored through depth-first search of the permutation tree.

task is not the first of the sequence. The other guardian received positive feedback—its task is the first one. This guardian initiates the construction of the chain. Runners that encounter the chain being built can contribute to its extension by positioning themselves at its end, one after the other. We refer to the last robot in the chain as the tail and to the others as the links. Tail and links align and keep a target distance between them so that the chain is stretched and straight. If the chain reaches a wall, then it turns, sweeping the environment. By extending and turning repeatedly, the chain eventually encounters another guardian. When this happens, the tail transfers its role to the guardian and becomes a link. The guardian initiates the construction of a new branch of the chain to ultimately include all the guardians. Robots that have not become chain members remain runners and navigate the environment by following the chain. When a runner reaches a guardian, it performs the guarded task if so directed. Guardians learn to direct runners via trial and error. As mentioned, a guardian received positive or negative feedback, depending on whether its task is the first to be performed. The guardian that received positive feedback will direct to its task the runners that have not yet performed any task. The other two guardians learn the correct policy with the help of the runners. The first time they are reached by a runner that has performed exactly one task, they ask it to perform their task and wait to be informed of the outcome. If the feedback is positive, then from then on, they will direct to their task the runners that have performed one task. If the feedback is negative, then they will direct to their task the runners that have performed two tasks. We empirically studied Mark I_3 with hardware and simulated experiments (Fig. 2, A, E, and F). Moreover, in simulated experiments, we assessed its scalability and robustness (Table 1 and Fig. 3). The results (Fig. 4, A and B, and Fig. 5A) show that Mark I_3 sequences three tasks reliably and operates correctly over a large range of conditions, without requiring any modification.

In Mark I_4, four tasks can be sequenced due to a minor difference relative to Mark I_3: A single counter that counts to four rather than three. We studied Mark I_4 in simulation (Figs. 2G and 3 and Table 1). The results show that the first assumptions of Mark I_3 can be overcome (Figs. 4C and 5B): More than three tasks can be sequenced.

Mark II_3 and Mark II_4

In Mark II_3, runners must perform an entire sequence before receiving any feedback. Because of the lack of immediate feedback, which in Mark I_3 breaks the initial symmetry, all guardians initiate the construction of a branch of the chain immediately after assuming their role. Upon completion, the chain is a closed loop that, besides routing runners as Mark I_3's chain, has the additional function of relaying information. By exchanging messages via the chain, the guardians (i) establish an initial sequence, out of which they generate a permutation tree spanning all possible sequences, and (ii) direct the runners to collectively explore such tree via depth-first search. The guardians establish an initial sequence by ordering themselves via a leader election algorithm (*44*). Each guardian communicates its unique identifier (ID) that is relayed by the chain. The guardian with the largest ID takes the label c and sends a message that is relayed clockwise along the closed-loop chain. The message contains the label b. The first guardian that receives the message takes the label b and propagates label a, which is eventually taken by the last guardian. Each guardian generates the tree of the permutations of the sequence (a, b, c). The tree is then collectively explored by the swarm via depth-first search. As a first step, the guardians address the runners to the tasks guarded by a, b, and c, in this order; as a second step, to the tasks guarded by a, c, and b; as a third step to the tasks guarded by b, a, and c, and so on. A failure reported by a runner after completing a sequence triggers the transition to the following one. On the other hand, a success indicates that the correct sequence has been identified. The exploration of the permutation tree is distributed. Throughout the process, all robots act reactively (sense-act), and each guardian has only partial knowledge about the sequence being tested (see Fig. 6 and the Supplementary Materials).

In Mark II_4, four tasks are sequenced under the assumption that no immediate feedback is received after task execution; the only difference relative to Mark II_3 is a counter that counts to four rather than three.

We studied Mark II_3 and Mark II_4 in simulation (Figs. 2, H and I, and 3 and Table 1). The results show that the two assumptions of Mark I_3 can be overcome (Figs. 4, D and E, and 7): The task sequencing problem can be solved even if no immediate feedback is received by the robots, and the tasks are more than three.

DISCUSSION

Because sequencing tasks is an albeit simple form of planning, TS-Swarm provides a different perspective on a pivotal debate in the history of artificial intelligence: planning in robotics. This debate opposes two competing, antithetical paradigms: the deliberative and reactive (*35*). According to the former, an intelligent robot should necessarily plan a course of action by reasoning on a model (*33*). According to the latter, a robot is more effective in dealing with reality by simply reacting to contingencies, without relying on reasoning and representation (*34*). Although hybrid systems have been proposed, they conceptually juxtapose the two paradigms: Deliberative and reactive instances—operating sequentially or in parallel—interact but remain logically distinct (*45*, *46*). By contrast, TS-Swarm associates the two paradigms: The ability to plan a sequence of tasks emerges at the collective level from the interaction of robots that, at the individual level, behave reactively without relying on reasoning and representation.

Relations with multirobot/agent learning

The learning process performed by TS-Swarm bears some resemblance to others described in the multirobot and multiagent literature (*47*–*49*).

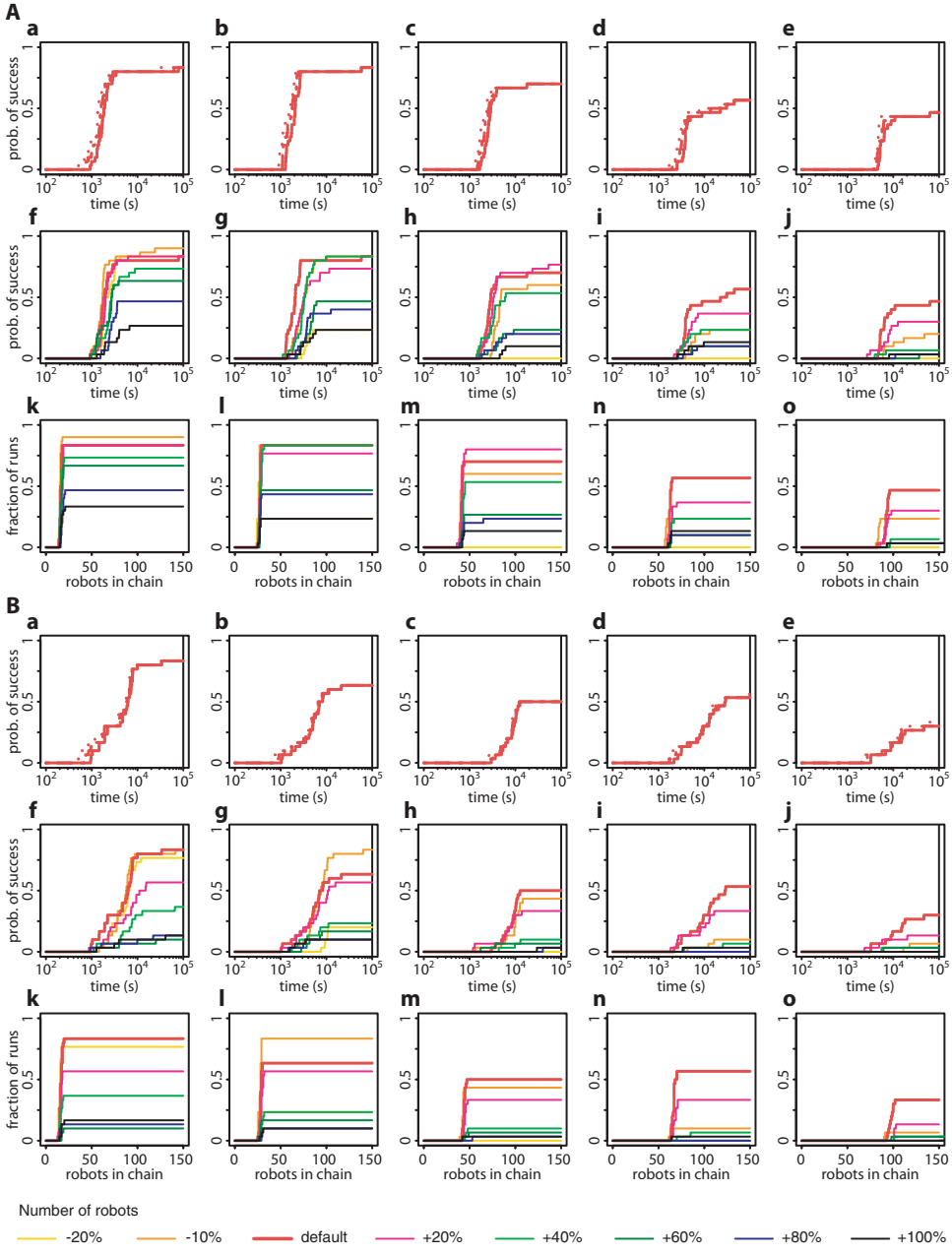

Fig. 7. Scalability and robustness of Mark II_3 and Mark II_4. (**A**) Mark II_3. (**B**) Mark II_4. See caption of Fig. 5.

TS-Swarm learns the correct sequence based on binary rewards: failures and successes experienced after performing tasks. No example of correct behavior is provided to the robots. The learning process performed by TS-Swarm can therefore be classified as reinforcement learning (*50*, *51*). More precisely, because no value function (*52*, *53*) is explicitly learned, the learning process of TS-Swarm could be seen as a form of direct policy search (*54*–*57*). In Mark I_m, feedback is received immediately after the execution of each single task. On the other hand, in Mark II_m, feedback is delayed and received only after the execution of a complete sequence. As a result, the sequencing problem presents a combinatorial nature: The resulting learning process is much more challenging. The robots learn collectively the correct sequence and the path to reach the areas where the task must be performed. A single learning process takes place, as opposed to collective systems, in which each agent/robot individually learns a behavior. In this sense, we can qualify the learning process of TS-Swarm as team learning (*58*, *59*). More precisely, because the behavior that is collectively learned is the same for all robots—the behavior that each robot (runner) must execute to perform the same correct sequence—the learning process can be qualified as a form of homogeneous team learning (*47*–*49*). Nonetheless, TS-Swarm differs from typical team learning systems (*60*–*62*) in that the single learning entity is the swarm as a whole, which has an immaterial and distributed nature: The robots operate in an independent manner, and no central entity exists that performs the learning process with a global view of the state of the system. Learning takes place at the collective level of the swarm: It is the swarm as a whole that searches the space of possible solutions. Moreover, once the correct solution is identified, the policy to produce it is eventually encoded by the chain in a collective and distributed way: Each guardian stores the part of policy that concerns the execution of its guarded task. Each runner implements the policy encoded by the chain on the basis of its own state, which is defined by the number of tasks performed and by which guardian is in its proximity, if any.

Limitations and possible improvements

Transmission of robot IDs limits scalability

The scalability of TS-Swarm is limited by the fact that robots include their ID in the range-and-bearing messages that they broadcast (see Materials and Methods and the Supplementary Materials). In addition, in Mark II_m, guardians use their ID in the leader election process.

Possible improvement. We could adopt locally unique IDs, which have been successfully demonstrated with a swarm of 1000 robots (*9*).

The number of tasks must be known at design time

All variants of TS-Swarm assume that the number of tasks to be sequenced is known at design time.

Possible improvement. We could let the swarm determine the number of tasks autonomously at run time. This would be straightforward in Mark II_m: When the closed-loop chain is established and the guardians order themselves using a leader election algorithm, the information on the number of tasks determined by the swarm in the environment is readily available to all the guardians.

Chains might overstep guardians

In some cases, the branch of chain being built fails to locate the guardian that it is supposed to reach. This may be due to several factors, such as (i) the lack of a sufficient number of robots to extend the branch up to the guardian or (ii) a temporary fault in the omnivision module of the tail. These factors cause the branch to overstep the guardian and eventually reach another branch of chain or the guardian from which the branch itself originates. As a result, the branch being built merges with another branch or collapses on itself. In both cases, the system fails.

Possible improvement. The tail could implement a mechanism to detect whether it is approaching another branch of chain or the originating guardian of its own branch. Should this happen, the tail could invert the sense in which the branch sweeps around its originating guardian. By alternating clockwise and counterclockwise sweeps, the branch being built would explore the environment more effectively and increase the chance to spot the guardian that it is supposed to reach.

Robots' movement is unsophisticated

Chain members and runners move in a simple and unsophisticated way. For simplicity, we implemented the links so that they stop moving upon being notified that the branch they form is established. If a runner bumps into a link, pushing it away from the correct position, then the continuity of the chain is broken, and the functionality of the whole system is compromised. This is more likely to occur when the swarm is composed of a large number of robots, the arena is large, and no immediate feedback is received by the robots (Mark II_m). In these cases, the branches are long and need to be functional for a long time to support the exploration of a large solution space.

Possible improvement. More refined movement mechanisms could be implemented to make the motion of chain members and runners more precise and reliable. Links could keep adjusting alignment and spacing even after their branch is established. In particular, they could benefit from a mechanism to regain their original position, should they detect that the functionality of their branch is compromised.

Chaining rests on restrictive assumptions

The chaining behavior works under the assumptions that (i) the arena is convex, (ii) the tasks are located along its perimeter, and (iii) there are no obstacles.

Possible improvement. We could relax these assumptions if the path between guardians were obtained by first covering the space with a lattice-like formation and then selecting the shortest path on this lattice. Robots that are not on the shortest path could leave; those that are on the shortest path would remain to act as waypoints. An approach to select the shortest path on a lattice has been demonstrated with a swarm of e-pucks (*63*). This approach is based on artificial pheromone.

The search strategy in Mark II_m is suboptimal

The transition from a candidate sequence to the following one in the permutation tree (Fig. 6) causes all the runners to abort the execution of the sequence being tested and start the execution of the following one from scratch. However, if the sequence being performed and the following one share an identical initial subsequence, then this solution is not optimal.

Possible improvement. We could implement a sort of backtracking for a more efficient exploration of the permutation tree. The runners that have performed only tasks contained in the identical initial subsequence could continue the execution of the remaining tasks of the following sequence without starting from scratch.

MATERIALS AND METHODS

The e-puck

The e-puck is a mobile two-wheeled differential drive robot designed for education and research (*64*). It is cylindrical in shape, with a diameter of 70 mm and a height of 50 mm. Its basic version is equipped with a PIC (peripheral interface controller) microcontroller and several sensors and actuators. The sensors are eight IR transceivers, which can be used to sense the presence of obstacles or to measure the intensity of ambient light, a color camera at the front of the robot, a microphone, and a three-axis accelerometer. The actuators are two stepper motors, which control the motion of the robot by differential steering (one motor for the left wheel and one for the right wheel), a ring of eight red LEDs, and a speaker.

The e-puck can be enhanced by the addition of various extension boards. For the research presented here, we extended the basic version of the e-puck with a range-and-bearing board (*65*), an omnivision module, and a Gumstix Overo board (Fig. 1C, left). The range-and-bearing board enables local communication between e-pucks via IR signals. It comprises 12 emitters and 12 receivers placed all around the body of the e-puck. The range-and-bearing board allows e-pucks to send and receive 4-byte messages at a rate of about 30 messages per second. Upon reception of a message, the board computes the distance (range) and angle (bearing) of the peer e-puck that sent the message. The omnivision module comprises an omnidirectional camera and three RGB LEDs and enhances the perception and local communication capabilities of the e-puck. Through the camera, an e-puck can see its neighboring peers and the TAMs. Moreover, it can perceive the color-coded status that the neighboring peers might display using their RGB LEDs. The Gumstix Overo board increases the computational capabilities of the e-puck and provides the flexibility and potential of a computer running Linux. It allows running C++ code, which is not possible on the PIC microcontroller of the basic version of the e-puck.

The basic version of the e-puck is powered by a rechargeable 3.7-V lithium-ion battery with a capacity of 1500 mA hour. The omnivision module houses a second battery with the same capacity to cope with the higher energy requirements of the extended e-puck. In a typical experiment, the full battery charge of an extended e-puck lasted about 40 min. We observed that, after about 45 min of continuous operation, the charge of the batteries was low. This negatively affected the behavior of the robots and, in particular, their ability to successfully transmit and receive messages through the range-and-bearing board.

The TAM

The TAM (*66*) is a device conceived for facilitating laboratory experiments with e-puck robots. A TAM represents an abstract task to be performed by an e-puck. The goal of the TAM is to abstract from task-specific details that are irrelevant to the objectives of an experiment. The TAM is particularly useful in experiments that focus on group dynamics rather than on the specific tasks performed by the individuals.

The TAM is a booth, which an e-puck can enter. For an e-puck, being into a TAM for a given time span amounts to performing the task abstracted by the TAM itself. The TAM has a cubical shape with sides of 120 mm (Fig. 1C, right). The TAM is controlled by a microcontroller (ATmega1284p, 16 MHz) and is equipped with two light barriers, three RGB LEDs, and an IR transceiver for short-range communication. Each TAM is powered by a rechargeable 3.7-V lithium-ion battery with a capacity of 1500 mA hour, the same battery used by the e-puck. In a typical experiment, a full battery charge lasts over 10 hours. The TAM is

also equipped with an XBee mesh network module that allows the synchronization of multiple TAMs. A group of TAMs can therefore be programmed to represent complex relationships between tasks. For example, a task could become activated only upon completion of another one, or a group of tasks could be performed successfully only in a specific order. The experimenter implements the logic that defines the relationship between tasks on a central computer. The computer dispatches commands to the TAMs to realize the relationships programmed by the experimenter. The TAMs and the central computer communicate wirelessly via the XBee mesh network module.

An e-puck perceives the colored LEDs of the TAM by using its omnidirectional camera. Different tasks are signaled by using different LED colors. An e-puck can decide to perform the task represented by a TAM by moving into it. The TAM detects the presence of the e-puck by means of its light barriers and reacts according to a logic defined by the experimenter. For example, upon the detection of an e-puck, the TAM could change the color of its LEDs or start communicating with the e-puck itself. The TAM and the e-puck communicate with each other through their IR transceivers. Communication between e-pucks and TAMs enables experiments in which e-pucks receive individual feedback for the tasks that they perform.

ARGoS

ARGoS (*67*) is a modular multirobot simulator and development environment conceived to be flexible and efficient. ARGoS provides a straightforward way to port control software developed in simulation to the robots without requiring any modification. To achieve this result, each sensor and actuator presents an interface with two back-end implementations: one for simulation and one for the robot. The control software of the robot directly interacts with this interface without having knowledge of which back-end implementation is being used. At link time, ARGoS makes sure that the appropriate back-end implementation is executed, depending on whether the execution is to take place in simulation or on the robot. ARGoS provides a number of physics engines. Some of them are kinematic engines that favor performance over realism; others are dynamics engines, in two or three dimensions, that require more computation but produce more realistic simulations. Because realism plays an important role in our simulations and the system that we propose comprises only robots that move on the ground, we used a dynamics engine in two dimensions in all the simulated experiments.

We used ARGoS to develop control software for and to simulate e-pucks and TAMs. We extended the basic model of the e-puck that was originally provided in ARGoS by implementing models of the range-and-bearing board, the omnivision module, and the Linux board (*68*). We also created the model of the TAM, which was not originally provided by ARGoS (*68*).

Experimental design

The goal of the experiments that we present here is to demonstrate TS-Swarm and provide evidence that it is able to successfully sequence tasks in an autonomous and distributed way. First, we demonstrated Mark I_3 both in reality with a swarm of 20 e-puck robots and in simulation. Besides showing the effectiveness of Mark I_3, this first experiment also provided an assessment of the simulator. After having shown that the simulator satisfactorily predicted the behavior of TS-Swarm on the e-puck robots, we adopted the simulator to perform a number of studies that we would be unable to perform with real robots. These studies either involved a large number of robots (more than what we had available) or lasted longer than the battery life of the robots. In particular, we performed a study in which we assessed the scalability of Mark I_3 by running experiments in which the number of robots ranged from 20 to 80 and the surface of the arena in which they operate ranged from 2.10 to 33.67 m^2. We also performed a study in which we assessed the robustness of Mark I_3 to the number of robots comprised in the swarm (Table 1 and Fig. 3). Last, we performed three studies to demonstrate Mark I_4, Mark II_3, and Mark II_4. In these studies, each run of the system lasted 100,000 s (i.e., about 28 hours), which was much longer than the battery life of the e-puck robot. As we did for Mark II_3, also for these three variants, we studied their scalability and robustness (Table 1 and Fig. 3).

The focus of the research presented here is on how a swarm can sequence tasks in an autonomous and distributed way rather than on the specific tasks that it should sequence. For this reason, in these experiments, we considered abstract tasks represented by TAMs. Robots operated in a bounded arena delimited by walls, which were 42-mm high. The arena was a regular hexagon when the tasks to be sequenced were three, and a regular octagon when the tasks were four. The TAMs abstracting the tasks were distributed along the perimeter of the arena and positioned in the middle of alternate sides. Each task was associated with a color. When the tasks were three, the colors were red (R), green (G), and blue (B). When they were four, the fourth color was orange (O). In each experimental run, the initial position of the robots, the correct sequence, and the relative position of the tasks were decided randomly. In movies S1 to S6, for clarity, the correct sequence was always RGB when the tasks were three and RGBO when they were four.

In all experiments, the final goal of TS-Swarm was to perform the correct sequence of tasks 10 times within a given time cap. As a performance measure, we considered the time required to complete 1, 5, and 10 executions of the correct sequence. The first execution indicates that TS-Swarm has been able to solve the task-sequencing problem. The 10th execution determines the final success of the system and, therefore, the end of the experiment. The fifth execution represents the midpoint of the two previous measures and provides visual information on whether the execution time grows linearly with the number of correct sequences performed.

Statistics

We report the performance of TS-Swarm via its empirical run-time distribution. Given one of the four variants of TS-Swarm (i.e., Mark I_3, Mark I_4, Mark II_3, or Mark II_4), a specific experimental setting (e.g., a setting characterized by the number of robots, the surface of the arena, and the time cap), and a target objective (i.e., the execution of 1, 5, or 10 correct sequences), we performed k independent runs and observed, for each run, the time required to attain the target objective. The empirical run-time distribution is the empirical distribution of these observations.

Formally, let TC be the time cap of each run, $j \in \{1, \ldots, k\}$ be the index of a run, r_j be the run-time of run j, and $k^t \leq k$ be the number of successful runs, that is, those runs $j : r_j < \mathrm{TC}$. The empirical run-time distribution is defined as $\mathrm{RTD}(t) = \hat{P}_s(\tau \leq t) = \#\{j \mid r_j \leq t\}/k$. Here, $\hat{P}_s(\tau \leq t)$ is an estimate of the probability that the system attains its target objective in an amount of time τ that is less than or equal to t. In other words, the empirical distribution $\mathrm{RTD}(t) = \hat{P}_s(\tau \leq t)$ is an estimate of the probability of success of the system over time (up to TC). For a given target objective and in a given experimental setting, the success ratio of the system within the time cap TC is $S_{\mathrm{TC}} = k^t/k$.

Experiments with Mark I_3

To complete a sequence, a robot must perform three tasks in a specific order, which was a priori unknown. Upon the execution of each task, the robot immediately received feedback—a success, if it performed the task in the right order; a failure, otherwise. In case of failure, the robot must restart the execution of the sequence from the beginning.

Robot experiments

We ran Mark I_3 10 times with 20 e-pucks. The experiments were performed in a controlled environment with a flat surface and uniform light conditions. The arena where the robots operated was a regular hexagon with sides of 0.9 m. A camera operating at about 3 frames per second was mounted on the ceiling with its axis lying on the vertical line passing through the center of the arena. We present the results of 10 consecutive runs. The performance of Mark I_3 in each of these 10 runs concurs to the statistics presented: No observed result was discarded for any reason whatsoever. Once a run started, it was accounted for in the statistics. The statistics therefore include also the failures. In table S3, we report the laboratory notebook, which includes the record of all the information that we collected during each of the 10 runs. A run was terminated either at the 10th execution of the correct sequence or at a time cap of 40 min (2400 s). Results are reported in Fig. 4A. A typical run is displayed in movie S1.

Assessment of the simulator

Alongside the experiments with the robots, we performed similar experiments in simulation using ARGoS, with the idea of producing an assessment of the simulation environment. The control software used in the two sets of experiments was the same: After performing the robot experiments, we ported the control software back to the simulated environment without any modification. Because performing experiments in the simulated environment is much less time-consuming than performing them in reality, we gathered results on 30 simulated runs. Moreover, because battery life is not a concern in simulation, we extended the duration of runs beyond the time cap of 40 min. Results are reported in Fig. 4B. Movie S2 shows a typical run in simulation, side by side with one on the robots.

Scalability study

We performed simulated experiments in five experimental settings. In each setting, we doubled the surface of the arena with respect to the previous one. We also increased the number of robots by a factor of $\sqrt{2}$. The rationale was that, by increasing the surface of the arena by a factor of 2, the distance between the TAMs increased by a factor of $\sqrt{2}$. We therefore expected the number of robots that became chain members to grow roughly by the same factor. By increasing the swarm size by a factor of $\sqrt{2}$, we expected that the robots would be sufficiently many to connect all the TAMs. The control software adopted in the scalability study was exactly the same in all the settings. The parameters that characterize the five settings are given in Table 1. We ran Mark I_3 30 times in each of the five settings (Fig. 3). Results are reported in Fig. 5A (a to e). Highlights of the scalability study are given in movie S3.

Robustness study

We used the same five experimental settings considered in the scalability study. For each of the five settings, we varied the number of robots with respect to the one adopted in the scalability study. We considered both a smaller (−10 and −20%) and a larger number of robots (+20, +40, +60, +80, and +100%). For each experimental setting and each number of robots tested, we report the run-time distribution for the successful execution of 10 sequences and the empirical distribution of the number of robots in the chain. We ran Mark I_3 30 times for each number of robots considered in each of the five settings (Table 1 and Fig. 3). Results are reported in Fig. 5A (f to o).

Experiments with Mark I_4

To complete a sequence, a robot must perform four tasks in a specific order, which was a priori unknown. The arena was a regular octagon with sides of 0.66 m. To connect the four TAMs, Mark I_4 needed to establish three branches of chain: one more than the two that Mark I_3 needed to establish. For this reason, we considered a swarm of 22 robots rather than 20 as we did for Mark I_3. We also increased the time cap to 100,000 s (i.e., about 28 hours). We ran Mark I_4 30 times in simulation. Finally, we studied the scalability and the robustness of Mark I_4 (Table 1 and Fig. 3). Results are reported in Figs. 4C and 5B. A typical run is displayed in movie S4.

Experiments with Mark II_3

We considered a scenario in which a robot needed to complete an entire sequence of tasks before being notified of a possible error. The tasks to be sequenced were three. The correct sequence was a priori unknown. The arena was the same as for the experiments with Mark I_3: a regular hexagon with sides of 0.9 m. Because Mark II_3 needed to build a closed-loop chain, the number of robots that it required was larger than the one required by Mark I_3. We considered a swarm of 25 robots rather than the 20 of the experiments performed with Mark I_3. Because Mark II_3 must explore a relatively large space of solutions, we increased the time cap to 100,000 s. We ran Mark II_3 30 times in simulation. Finally, we studied the scalability and the robustness of Mark II_3 (Table 1 and Fig. 3). Results are reported in Figs. 4D and 7A. A typical run is displayed in movie S5.

Experiments with Mark II_4

We considered a scenario similar to the one considered for Mark II_3, with the only difference that the tasks to be sequenced were four. The arena was the same as for the experiments with Mark I_4: a regular octagon with sides of 0.66 m. We considered a swarm of 27 robots—more than those considered in the experiments with Mark I_4 because Mark II_4 needed to build a closed-loop chain. As we did in the experiments with Mark II_3, we set the time cap to 100,000 s. We ran Mark II_4 30 times in simulation. Last, we studied the scalability and the robustness of Mark II_4 (Table 1 and Fig. 3). Results are reported in Figs. 4E and 7B. A typical run is displayed in movie S6.

SUPPLEMENTARY MATERIALS

robotics.sciencemag.org/cgi/content/full/3/20/eaat0430/DC1

Section S1. Detailed description of TS-Swarm
Section S2. Discussion of the results
Section S3. Highlights of movie S1
Fig. S1. State machine of TS-Swarm.
Fig. S2. Encoding of the range-and-bearing message in Mark I_3.
Fig. S3. Guardians.
Fig. S4. State machine of a guardian.
Fig. S5. Guardian protocol (G protocol), sequence diagram.
Fig. S6. Motion of a link.
Fig. S7. Tail.
Fig. S8. Tail protocol (T protocol), sequence diagram.
Fig. S9. Construction and motion of a branch of chain.
Fig. S10. Runners.
Fig. S11. Trajectory followed by the runners around the chain.
Fig. S12. Motion of a runner along a branch of the chain.
Fig. S13. The chain in Mark I_3 and Mark I_4.
Fig. S14. The chain in Mark II_3 and Mark II_4.
Fig. S15. Exploration of the space of possible sequences in Mark II_3.

Fig. S16. Encoding of the range-and-bearing message in Mark II_3.
Fig. S17. Exploration of the space of possible sequences in Mark II_4.
Fig. S18. Number of chain members in Mark I_3.
Fig. S19. Comparison between Mark II_3 and Mark II_4.
Table S1. Guardian protocol (G protocol), description of messages.
Table S2. Tail protocol (T protocol), description of messages.
Table S3. Laboratory notebook.
Movie S1. Mark I_3: Experiment with robots.
Movie S2. Mark I_3: Reality and simulation, side by side.
Movie S3. Mark I_3: Scalability study.
Movie S4. Mark I_4: Four tasks.
Movie S5. Mark II_3: Delayed feedback.
Movie S6. Mark II_4: Four tasks with delayed feedback.

REFERENCES AND NOTES

1. G. Beni, From swarm intelligence to swarm robotics, in *Swarm Robotics* (Springer, 2005), pp. 1–9.
2. E. Şahin, Swarm robotics: From sources of inspiration to domains of application, in *Swarm Robotics* (Springer, 2005), pp. 10–20.
3. M. Dorigo, M. Birattari, M. Brambilla, Swarm robotics. *Scholarpedia* **9**, 1463 (2014).
4. G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield, B. J. Nelson, B. Scassellati, M. Taddeo, R. Taylor, M. Veloso, Z. L. Wang, R. Wood, The grand challenges of *Science Robotics*. *Sci. Robot.* **3**, eaar7650 (2018).
5. S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, E. Bonabeau, *Self-Organization in Biological Systems* (Princeton Univ. Press, 2001).
6. M. Gauci, J. Chen, W. Li, T. Dodd, R. Groß, Self-organized aggregation without computation. *Int. J. Rob. Res.* **33**, 1145–1161 (2014).
7. M. Schwager, J. McLurkin, D. Rus, Distributed coverage control with sensory feedback for networked robots, in *Robotics: Science and Systems* (MIT Press, 2006), p. 007.
8. M. Duarte, V. Costa, J. Gomes, T. Rodrigues, F. Silva, S. M. Oliveira, A. L. Christensen, Evolution of collective behaviors for a real swarm of aquatic surface robots. *PLOS ONE* **11**, e0151834 (2016).
9. M. Rubenstein, A. Cornejo, R. Nagpal, Programmable self-assembly in a thousand-robot swarm. *Science* **345**, 795–799 (2014).
10. N. Mathews, A. Christensen, R. O'Grady, F. Mondada, M. Dorigo, Mergeable nervous systems for robots. *Nat. Commun.* **8**, 439 (2017).
11. C. Virágh, G. Vásárhelyi, N. Tarcai, T. Szörényi, G. Somorjai, T. Nepusz, T. Vicsek, Flocking algorithm for autonomous flying robots. *Bioinspir. Biomim.* **9**, 025012 (2014).
12. R. O'Grady, R. Groß, A. L. Christensen, M. Dorigo, Self-assembly strategies in a group of autonomous mobile robots. *Auton. Robots* **28**, 439–455 (2010).
13. M. Rubenstein, A. Cabrera, J. Werfel, G. Habibi, J. McLurkin, R. Nagpal, Collective transport of complex objects by simple robots: Theory and experiments, in *AAMAS '13 Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems* (IFAAMAS, 2013), pp. 47–54.
14. M. Gauci, J. Chen, W. Li, T. Dodd, R Groß, Clustering objects with robots that do not compute, in *AAMAS '14 Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems* (IFAAMAS, 2014), pp. 421–428.
15. J. Werfel, K. Petersen, R. Nagpal, Designing collective behavior in a termite-inspired robot construction team. *Science* **343**, 754–758 (2014).
16. J. Halloy, G. Sempo, G. Caprari, C. Rivault, M. Asadpour, F. Tâche, I. Saïd, V. Durier, S. Canonge, J.-M. Amé, C. Detrain, N. Correll, A. Martinoli, F. Mondada, R. Siegwart, J.-L. Deneubourg, Social integration of robots into groups of cockroaches to control self-organized choices. *Science* **318**, 1155–1158 (2007).
17. S. Garnier, J. Gautrais, M. Asadpour, C. Jost, G. Theraulaz, Self-organized aggregation triggers collective decision making in a group of cockroach-like robots. *Adapt. Behav.* **17**, 109–133 (2009).
18. A. Özdemir, M. Gauci, S. Bonnet, R. Groß, Finding consensus without computation. *IEEE Robot. Autom. Lett.* **3**, 1346–1353 (2018).
19. G. Pini, A. Brutschy, M. Frison, A. Roli, M. Dorigo, M. Birattari, Task partitioning in swarms of robots: An adaptive method for strategy selection. *Swarm Intell.* **5**, 283–304 (2011).
20. E. Castello, T. Yamamoto, F. D. Libera, W. Liu, A. F. T. Winfield, Y. Nakamura, H. Ishiguro, Adaptive foraging for simulated and real robotic swarms: The dynamical response threshold approach. *Swarm Intell.* **10**, 1–31 (2016).
21. Á. Gutiérrez, A. Campo, F. Monasterio-Huelin, L. Magdalena, M. Dorigo, Collective decision-making based on social odometry. *Neural Comput. Appl.* **19**, 807–823 (2010).
22. G. Valentini, E. Ferrante, H. Hamann, M. Dorigo, Collective decision with 100 kilobots: Speed versus accuracy in binary discrimination problems. *Auton. Agent. Multi Agent Syst.* **30**, 553–580 (2016).
23. T. Schmickl, K. Crailsheim, Trophallaxis within a robotic swarm: Bio-inspired communication among robots in a swarm. *Auton. Robots* **25**, 171–188 (2008).
24. M. A. Montes de Oca, E. Ferrante, A. Scheidler, C. Pinciroli, M. Birattari, M. Dorigo, Majority-rule opinion dynamics with differential latency: A mechanism for self-organized collective decision-making. *Swarm Intell.* **5**, 305–327 (2011).
25. A. Reina, G. Valentini, C. Fernández-Oto, M. Dorigo, V. Trianni, A design pattern for decentralised decision making. *PLOS ONE* **10**, e0140950 (2015).
26. A. Scheidler, A. Brutschy, E. Ferrante, M. Dorigo, The k-unanimity rule for self-organized decision making in swarms of robots. *IEEE Trans. Syst. Man Cybern.* **46**, 1175–1188 (2016).
27. E. O. Wilson, Caste and division of labor in leaf-cutter ants (Hymenoptera: Formicidae: *Atta*). *Behav. Ecol. Sociobiol.* **7**, 143–156 (1980).
28. T. D. Seeley, *The Wisdom of the Hive* (Harvard Univ. Press, 1996).
29. E. Bonabeau, G. Theraulaz, J.-L. Deneubourg, Fixed response thresholds and the regulation of division of labor in insect societies. *Bull. Math. Biol.* **60**, 753–807 (1998).
30. M. J. B. Krieger, J.-B. Billeter, L. Keller, Ant-like task allocation and recruitment in cooperative robots. *Nature* **406**, 992–995 (2000).
31. S. Nouyan, R. Gross, M. Bonani, F. Mondada, M. Dorigo, Teamwork in self-organized robot colonies. *IEEE Trans. Evol. Comput.* **13**, 695–711 (2009).
32. T. Schmickl, R. Thenius, C. Moslinger, J. Timmis, A. Tyrrell, M. Read, J. Hilder, J. Halloy, A. Campo, C. Stefanini, L. Manfredi, S. Orofino, S. Kernbach, T. Dipper, D. Sutantyo, CoCoRo—The self-aware underwater swarm, in *Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2011 Fifth IEEE Conference* (IEEE Press, 2011), pp. 120–126.
33. N. J. Nilsson, "Shakey the robot" (Tech. note 323, SRI AI Center, 1984).
34. R. A. Brooks, Intelligence without representation. *Artif. Intell.* **47**, 139–159 (1991).
35. R. R. Murphy, *Introduction to AI Robotics* (MIT Press, 2000).
36. S. Goss, J.-L. Deneubourg, Harvesting by a group of robots, in *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life* (MIT Press, 1992), pp. 195–204.
37. A. Drogoul, J. Ferber, From Tom Thumb to the dockers: Some experiments with foraging robots, in *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior* (MIT Press, 1992), pp. 451–459.
38. B. Werger, M. Matarić, Robotic food chains: Externalization of state and program for minimal-agent foraging, in *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior* (MIT Press, 1996), pp. 625–634.
39. S. Nouyan, M. Dorigo, Chain based path formation in swarms of robots, in *Ant Colony Optimization and Swarm Intelligence, 5th International Workshop* (Springer, 2006), pp. 120–131.
40. S. Nouyan, A. Campo, M. Dorigo, Path formation in a robot swarm: Self-organized strategies to find your way home. *Swarm Intell.* **2**, 1–23 (2008).
41. V. Sperati, V. Trianni, S. Nolfi, Self-organised path formation in a swarm of robots. *Swarm Intell.* **5**, 97–119 (2011).
42. F. Ducatelle, G. A. D. Caro, C. Pinciroli, F. Mondada, L. Gambardella, Communication assisted navigation in robotic swarms: Self-organization and cooperation, in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE Press, 2011), pp. 4981–4988.
43. M. Dorigo, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy, D. Burnier, A. Campo, A. L. Christensen, A. Decugnière, G. Di Caro, F. Ducatelle, E. Ferrante, A. Förster, J. Martinez Gonzales, J. Guzzi, V. Longchamp, S. Magnenat, N. Mathews, M. Montes de Oca, R. O'Grady, C. Pinciroli, G. Pini, P. Rétornaz, J. Roberts, V. Sperati, T. Stirling, A. Stranieri, T. Stützle, V. Trianni, E. Tuci, A. E. Turgut, F. Vaussard, Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robot. Autom. Mag.* **20**, 60–71 (2013).
44. E. Chang, R. Roberts, An improved algorithm for decentralized extrema-finding in circular configurations of processes. *Commun. ACM* **22**, 281–283 (1979).
45. R. C. Arkin, Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Rob. Auton. Syst.* **6**, 105–122 (1990).
46. A. Saffiotti, K. Konolige, E. H. Ruspini, A multivalued logic approach to integrating planning and control. *Artif. Intell.* **76**, 481–526 (1995).
47. T. Haynes, S. Sen, Evolving behavioral strategies in predators and prey, in *IJCAI 1995: Adaption and Learning in Multi-Agent Systems* (Springer, 1996), pp. 113–126.
48. R. P. Sałustowicz, M. A. Wiering, J. Schmidhuber, Learning team strategies: Soccer case studies. *Mach. Learn.* **33**, 263–282 (1998).
49. M. Quinn, L. Smith, G. Mayley, P. Husbands, Evolving teamwork and role-allocation with real robots, in *ICAL 2003 Proceedings of the Eighth International Conference on Artificial Life* (MIT Press, 2003), pp. 302–311.
50. R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, 1998).
51. J. Kober, J. A. Bagnell, J. Peters, Reinforcement learning in robotics: A survey. *Int. J. Rob. Res.* **32**, 1238–1274 (2013).

52. R. S. Sutton, Learning to predict by the methods of temporal differences. *Mach. Learn.* **3**, 9–44 (1988).
53. C. J.C. H. Watkins, P. Dayan, *Q*-learning. *Mach. Learn.* **8**, 279–292 (1992).
54. L. Baird, A. Moore, Gradient descent for general reinforcement learning, in *Advances in Neural Information Processing Systems* (MIT Press, 1999), pp. 968–974.
55. J. Baxter, P. L. Bartlett, Reinforcement learning in POMDPs via direct gradient ascent, in *ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning* (Morgan Kaufmann Publishers Inc., 2000), pp. 41–48.
56. C. W. Anderson, "Approximating a policy can be easier than approximating a value function" (Tech. rep. CS-00-101, Colorado State University, 2000).
57. M. T. Rosenstein, A. G. Barto, Robot weightlifting by direct policy search, in *IJCAI'01 Proceedings of the 17th International Joint Conference on Artificial Intelligence* (Morgan Kaufmann Publishers Inc., 2001), pp. 839–844.
58. L. Panait, S. Luke, Cooperative multi-agent learning: The state of the art. *Auton. Agent. Multi Agent Syst.* **11**, 387–434 (2005).
59. L. Buşoniu, R. Babuška, B. De Schutter, A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Syst. Man Cybern. C* **38**, 156–172 (2008).
60. P. Stone, M. Veloso, Multiagent systems: A survey from a machine learning perspective. *Auton. Robots* **8**, 345–383 (2000).
61. L. E. Parker, Decision making as optimization in multi-robot teams, in *ICDCIT 2012: Distributed Computing and Internet Technology* (Springer, 2012), pp. 35–49.
62. J. Girard, M. R. Emami, Concurrent Markov decision processes for robot team learning. *Eng. Appl. Artif. Intell.* **39**, 223–234 (2015).
63. A. Campo, Á. Gutiérrez, S. Nouyan, C. Pinciroli, V. Longchamp, S. Garnier, M. Dorigo, Artificial pheromone for path selection by a foraging swarm of robots. *Biol. Cybern.* **103**, 339–352 (2010).
64. F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, A. Martinoli, The e-puck, a robot designed for education in engineering, in *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions* (IPCB, 2009), pp. 59–65.
65. Á. Gutiérrez, A. Campo, M. Dorigo, J. Donate, F. Monasterio-Huelin, L. Magdalena, Open e-puck range & bearing miniaturized board for local communication in swarm robotics, in *2009 IEEE International Conference on Robotics and Automation* (IEEE Press, 2009), pp. 3111–3116.
66. A. Brutschy, L. Garattoni, M. Brambilla, G. Francesca, G. Pini, M. Dorigo, M. Birattari, The TAM: Abstracting complex tasks in swarm robotics research. *Swarm Intell.* **9**, 1–22 (2015).
67. C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, M. Dorigo, ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intell.* **6**, 271–295 (2012).
68. L. Garattoni, G. Francesca, A. Brutschy, C. Pinciroli, M. Birattari, "Software infrastructure for e-puck (and TAM)" (Tech. rep. TR/IRIDIA/2015-004, IRIDIA, Université libre de Bruxelles, 2015).

Acknowledgments: We thank A. Roli, M. Brambilla, and G. Lucy for reading a preliminary version of the article. M.B. dedicates his work to the memory of his father. **Funding:** The project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 681872). M.B. acknowledges support from the Belgian *Fonds de la Recherche Scientifique*, of which he is a Senior Research Associate. **Author contributions:** The authors devised the system together. L.G. realized it and performed the experiments. The authors wrote the manuscript together. M.B. conceived and directed the project. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data generated and discussed are included in the article and in the Supplementary Materials.

Submitted 18 January 2018
Accepted 20 June 2018
Published 18 July 2018
10.1126/scirobotics.aat0430

Citation: L. Garattoni, M. Birattari, Autonomous task sequencing in a robot swarm. *Sci. Robot.* **3**, eaat0430 (2018).

Science Robotics

Autonomous task sequencing in a robot swarm

Lorenzo Garattoni and Mauro Birattari

Sci. Robot. **3** (20), eaat0430. DOI: 10.1126/scirobotics.aat0430

Science Robotics (ISSN 2470-9476) is published by the American Association for the Advancement of Science. 1200 New York Avenue NW, Washington, DC 20005. The title *Science Robotics* is a registered trademark of AAAS.