HAL
archives-ouvertes.fr

Understanding the Power of Stigmergy of Anonymous Agents in Discrete Environments

Gianlorenzo d'Angelo, Xavier Défago, Nicolas Nisse

To cite this version:

Gianlorenzo d'Angelo, Xavier Défago, Nicolas Nisse. Understanding the Power of Stigmergy of Anonymous Agents in Discrete Environments. [Research Report] RR-8614, Inria. 2014. hal-01073368

HAL Id: hal-01073368

https://hal.inria.fr/hal-01073368

Submitted on 9 Oct 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Understanding the Power of Stigmergy of Anonymous Agents in Discrete Environments

Gianlorenzo D'Angelo, Xavier Défago, Nicolas Nisse

RESEARCH REPORT
N° 8614
October 2014
Project-Teams COATI

ISSN 0249-6399 ISRN INRIA/RR--8614--FR+ENG

Understanding the Power of Stigmergy of Anonymous Agents in Discrete Environments*

Gianlorenzo D'Angelo†, Xavier Défago‡, Nicolas Nisse§

Project-Teams COATI

Research Report n° 8614 — October 2014 — 28 pages

* This research was supported in part by JSPS KAKENHI Grant Number 26330020.
† Gran Sasso Science Institute (GSSI), L'Aquila, Italy
‡ School of Information Science, JAIST, Nomi, Japan
§ Inria & Univ. Nice Sophia Antipolis, CNRS, I3S, Sophia Antipolis, France

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Abstract: Communication by stigmergy consists, for agents/robots devoid of other dedicated communication devices, in exchanging information by observing each other's movements, similar to how honeybees use a dance to inform each other on the location of food sources.
Stigmergy, while a popular technique in soft computing (e.g., swarm intelligence and swarm robotics), has received little attention from a computational viewpoint, with only one study proposing a method in a continuous environment. An important question is whether there are limits intrinsic to the environment on the feasibility of stigmergy. While it is not the case in a continuous environment, we show that the answer is quite different when the environment is discrete.
This paper considers stigmergy in graphs and identifies classes of graphs in which robots can communicate by stigmergy. We provide two algorithms with different tradeoffs. One algorithm achieves faster stigmergy when the density of robots is low enough to let robots move independently. This algorithm works when the graph contains some particular pairwise-disjoint subgraphs. The second algorithm, while slower solves the problem under an extremely high density of robots assuming that the graph admits some large cycle. Both algorithms are described in a general way, for any graph that admits the desired properties and with identified nodes. We show how the latter assumption can be removed in more specific topologies. Indeed, we consider stigmergy in the grid which offers additional orientation information not available in a general graphs, allowing us to relax some of the assumptions.
Given an $N \times M$ anonymous grid, we show that the first algorithm requires $O(\mathcal{M})$ steps to achieve communication by stigmergy, where $\mathcal{M}$ is the maximum length of a communication message, but it works only if the number of robots is less than $\left\lfloor \frac{N \cdot M}{9} \right\rfloor$. The second algorithm, which requires $O(k^2)$ steps, where k is the number of robots, on the other hand, works for up to $N \cdot M - 5$ robots. In both cases, we consider very weak assumptions on the robots capabilities: i.e., we assume that the robots are anonymous, asynchronous, uniform, and execute deterministic algorithms.

Key-words: Mobile agents in graphs; Computational robots; Algorithms; Complexity; Information exchange; Theoretical computer science

Stigmergy par des robots anonymes dans des environnements discrets

Résumé : Dans cette article, nous identifions des classes de graphes dans lesquelles des robots peuvent communiquer par stigmergie, c'est-à-dire, peuvent échanger de l'information en observant les mouvements des autres robots.

Mots-clés : graphe, stigmergie, agents mobiles

1 Introduction

Unmanned automatic vehicles, flexible manufacturing systems, rescue robot teams, drone fleets, automated construction, and many other systems of which we will see more and more in the near future, share many common characteristics, among which the necessity to communicate. Indeed, without communication, there is no cooperation, or coordination.

In many cases, it is however not feasible, economical, or desirable for the robots to rely on explicit communication, such as wireless network, for their coordination. This may for instance be due to intrinsic limitations of the robots (e.g., cheap robots or payload limits with flying robots) or due to the nature of the environment (e.g., noise pollution in underwater applications).

Nevertheless, it may still be possible for the robots to exchange information by observing their respective actions on the environment. This notion is known as "stigmergy" and was originally defined by Grassé in the context of research on termites [1]. Stigmergy has become popular in computer science through the work on swarm intelligence and ant colony optimization of Bonabeau et al. [2]. Ant colony optimization relies on the notion of "pheromones" which is some piece of information left in the environment by the agents (robots/ants) that gradually decays over time.

In robot applications, implementing a mechanism akin to pheromones is impractical at best. In contrast, Karaboga et al. [7] proposed a variant called bee colony optimization in which agents (robots/bees) perform a dance, inspired by the waggle dance of honeybees [6], to communicate to their peers information such on the location (heading and distance) of a food source.

Both approaches have been applied to robots in the field of swarm robotics (e.g., work by Martinolli and Mondada [8]). In spite of a large literature presenting very interesting experimental studies of swarm optimization (see two decades of research on swarm robotics; e.g., [9, 10]), little has been done to investigate the fundamental limits of stigmergy itself.

In this paper, we define *stigmergy* as an exchange of information between the robots that compose a system devoid of explicit means of communication. A robot can learn some information from another one by observing its moves. More specifically, let $\{R_i\}_{(1 \leq i \leq k)}$ be a set of k robots, such that each robot carries a message m_i which is a string of symbols taken from a language $\mathcal{L} = \{\lambda_1, \cdots, \lambda_\ell\}$ consisting of $\ell \geq 1$ distinct symbols. For convenience, let $\mathcal{M}$ be the length in number of symbols of the longest message. There is a time after which the following must hold for any robot R_i:

S1. All robots know m_i;

S2. R_i knows that all robots know m_i;

S3. R_i enters a terminated-state in which it takes no further moves.

A protocol satisfying constraint S3 is said to be *quiescent*.

Dieudonné et al. [4] have proposed a method for communication by stigmergy in a continuous environment with chirality, under the semi-synchronous

model proposed by Suzuki and Yamashita [11]. Bouzid et al. [3] have proposed an extension called RoboCast, also in a continuous environment, under the asynchronous model of Prencipe et al. [5]. While their original motivation was somewhat different (generalize a method for robots to exchange their local coordinate systems originally proposed by Suzuki and Yamashita [11]), that approach is the first general method, combining explicit communication and orientation information, that we are aware of.

RoboCast heavily relies on the fact that the environment is continuous, and cannot possibly be adapted to a discrete environment (e.g., a graph). In particular, in sharp contrast with a discrete environment, robots evolving in a continuous environment can use a finite area to show a direction to the other robots by recursively moving halfway along a segment (Zeno's arrow). In other words, any segment of finite length is infinitely divisible, which is clearly not the case when the environment is discrete.

Considering stigmergy in a discrete environment raises several interesting questions. First, what are sufficient properties of the underlying graph such that robots can communicate by stigmergy? Second, what are the limits with respect to robots density such that robots retain enough freedom of movement to achieve stigmergy?

The contribution of this paper is twofold. First, we consider stigmergy in graphs and identify classes of graphs in which robots can communicate by stigmergy (Section 3). We provide two algorithms with different tradeoffs. One algorithm achieves faster stigmergy when the density of robots is low enough to let robots move independently. This algorithm works when the graph contains some particular pairwise-disjoint subgraphs. The second algorithm, while slower solves the problem under an extremely high density of robots assuming that the graph admits some large cycle. Both algorithms are described in a general way, for any graph that admits the desired properties and with identified nodes. Second, we consider stigmergy in the grid and provide concrete instances of the general concepts developed for the general graph (Section 4). The case of the grid is interesting because its nature offers additional orientation information not available in a general graph. This allows to relax some of the assumptions. For instance, nodes do not need to be identified anymore. In detail, given a $N \times M$ anonymous grid with k robots, we show that the first algorithm requires $O(\mathcal{M})$ steps while the second one requires $O(k^2)$ steps to solve the stigmergy problem. On the other hand, the first algorithm works only if k is less than $\lfloor \frac{N \cdot M}{9} \rfloor$, while the second one works for up to $N \cdot M - 5$ robots In both cases, we consider very weak assumptions on the robots capabilities: i.e. we assume that the robots are anonymous, asynchronous, uniform, and execute deterministic algorithms.

2 Model and Definitions

The environment consists of an undirected connected simple n-node graph. Nodes of the graph are *a priori* anonymous, but we will assume some prop-

erties about the position of robots, allowing us to break the symmetries (see below for more details and see Section 4 for the case of grids).

The system consists of $k \geq 2$ anonymous robots $R_1, \cdots, R_k$ evolving in the environment. The robots initially occupy k distinct nodes of an $n \geq k$ node graph. The robots have no explicit means of communication and can interact only through their actions in the environment.

Robots proceed asynchronously through activation cycles that consist of the *Look*, *Compute*, and *Move* operations [5]. In the *Look* operation, a robot obtains a snapshot of all nodes currently occupied by a robot and is able to distinguish its own current location. This information is used as input to the *Compute* operation where the robot executes a function to decide on a *Move* operation. The possible moves are to either move to an adjacent node or stay at the current location. The *Look* and *Move* operations are both atomic, which means that no robot can be observed while traversing an edge of the graph. Robots however become active asynchronously and the time elapsed between two successive *Look* and *Move* operations is arbitrary but finite.

Hence, In contrast to the continuous case, we assume that moves are instantaneous, and hence any robot performing a Look operation sees all other robots at nodes and not on edges. Note that, in a discrete asynchronous environment this does not constitute a limitation to the model. In fact, an algorithm cannot take advantages from seeing robots on the edges as the adversary can decide to perform the Look operations only when the robots are on the nodes. On the other hand, if an algorithm takes advantage from the assumption that the robots always occupy nodes, the same algorithm can be applied by adding the rule that if a robot sees another robot on an edge, it just don't move (i.e. it waits until all the robots occupy only nodes).

Two robots moving to the same node are said to collide, resulting in their destruction. The system must ensure the *exclusivity property*. That is, initially all robots are occupying distinct nodes and no two robots are allowed to simultaneously occupy the same node. In particular, all protocols presented in this paper enforce the exclusivity property.

We are interested in how robots can communicate information to each other (albeit they have no direct means of communication). That is, each robot has a string of symbols that other robots must learn and a robot can learn this information only by observing the positions and moves of the other robots. Obviously, each robot requires some memory in order to, at least, remember the bits from other robots. Hence, when we refer to the memory of robots, we only consider the *control-memory*, i.e., the memory used to execute the process but not to store the desired information.

2.0.1 Pseudo-Synchronization.

We require algorithms to enforce a set of basic constraints which, when put together, provide properties useful for capturing robots movements. We call these rules *pseudo-synchronization*. A protocol is *pseudo-synchronized* if it satisfies the following three properties:

1. Initially, each robot takes a snapshot and then moves from its initial position;
2. After having moved with respect to a snapshot $\mathcal{S}$, a robot must move if and only if all other robots have moved from their position in $\mathcal{S}$;
3. A robot cannot cross the same edge twice consecutively.

In a pseudo-synchronized protocol, time can be divided into phases such that, for any $i \geq 1$, at the end of Phase i, each robot has moved exactly i times. In particular, such a protocol is non-blocking. For any robot R, let v_0^R be its initial position and let v_i^R be its position at the end of Phase i, for any $i \geq 1$.

For ease of presentation, in the following lemma, we assume that the robots can be distinguished (i.e. they have identifiers). This is only needed for detecting whether or not a robot has moved. In other sections, we show how to implement this property in various graph classes.

Lemma 1 *Let us consider a team of k robots executing a pseudo-synchronized protocol. If robots can be distinguished in the snapshots, then for any $i \geq 1$, at the end of Phase i, any robot has seen R in at least one node in $\{v_{i-1}^R, v_i^R\}$.*

Proof. The proof is by induction on i.

Let us first consider the case $i = 0$. When Robot R takes its first snapshot, some robots may have already executed their first move. Let A_0^R be the set of robots that have already moved before R takes its first snapshot, and let B_0^R be the set of other robots. Note that each robot in A_0^R must have moved exactly once since before moving they saw R in v_0^R and therefore, by Property 2 of a pseudo-synchronized protocol, they cannot move again while R has not left its current position. Hence, each robot R' in A_0^R is seen in $v_1^{R'}$ by R and each robot R' in B_0^R is seen by R in $v_0^{R'}$. Because of Property 1 of pseudo-synchronization, Robot R eventually reaches v_1^R.

Applying the same arguments to each robot, each robot R eventually reaches v_1^R before any robot executes a second move. Hence, there is a time when all robots have moved exactly once which defines the end of the first Phase. Moreover, at the end of Phase 1, each robot has seen any other robot R either in v_0^R or in v_1^R.

Let $i \geq 1$. By the induction hypothesis, at the end of Phase i, each robot R is occupying v_i^R and has seen any other robot R' either in $v_{i-1}^{R'}$ or in $v_i^{R'}$. Moreover, let $\mathcal{S}_i^R$ be last snapshot taken by robot R before moving from v_{i-1}^R to v_i^R. Note that, for any robot R', $\mathcal{S}_i^R$ shows R' either at $v_{i-1}^{R'}$ or at $v_i^{R'}$. We denote by $R \leq_i R'$ if $\mathcal{S}_i^R$ has not been taken after $\mathcal{S}_i^{R'}$. Note that $\leq_i$ is a partial order on the robots. Note also that, if $R \leq_i R'$ then $\mathcal{S}_i^R$ shows R' at $v_{i-1}^{R'}$.

For purpose of contradiction, assume that not all robots eventually move during Phase $i+1$. Let A_{i+1} be the set of robots that have moved during Phase $i+1$ and let B_{i+1} be the set of other robots. Let R be any minimal (for the $\leq_i$ relation) robot in B_{i+1}. Let consider $R' \in A_{i+1}$. By the induction hypothesis, $\mathcal{S}_i^R$ shows R' in $v_{i-1}^{R'}$ or $v_i^{R'}$ and, by definition of A_{i+1}, R' is now

occupying $v_{i+1}^{R'}$ (note that, by Property 2, R' cannot have moved more than once before all other robots have moved). Since, by Property 3 of pseudo-synchronization, $v_{i+1}^{R'} \notin \{v_{i-1}^{R'}, v_i^{R'}\}$, Robot R can detect that R' has moved. Let consider $R' \in B_{i+1}$. By minimality of R, $\mathcal{S}_i^R$ shows R' at $v_{i-1}^{R'}$. Moreover, R' is occupying $v_i^{R'}$. Hence, Robot R can detect that R' has moved. Therefore, R must eventually move.

Moreover, before its move in Phase $i+1$, any robot R' has been seen by R either in $v_i^{R'}$ or in $v_{i+1}^{R'}$. ■

3 Stigmergy in general graphs

In this section, we give two algorithms to solve the stigmergy problem in general graphs. The first algorithm allows the robot to communicate in parallel, thus requiring a small number of phases. On the other hand, this algorithm works only with a small number of robots. In contrast, the second algorithm, pertinent to some graph classes, allows a large number of robots to communicate but forces the robots to transmit their messages one-by-one, thus requiring more phases.

3.1 Parallel communication

In this section, we present a generic algorithm to solve the stigmergy problem. This algorithm is pseudo-synchronized and therefore, by Lemma 1, we can think of the time as being divided into discrete phases. The main advantage of this algorithm is that it allows the robots to transmit *in parallel*. That is, after a constant number of phases (independent of n and k), all robots can transmit one symbol to each other. On the other hand, our algorithm assumes that some disjoint subgraphs—the size of which depends on the number ℓ of possible symbols—are available for each robot. This algorithm hence requires a particular structure of the graph and the number of robots to be "small enough" when compared to the size of the graph. The other drawback of this protocol is that the control-memory required by each robot is linear in the number k of robots (this comes from the fact that each robot has to deal with all other robots in parallel).

First, let us define the graph structure that must be available to each robot. We assume that such a structure is available and we show how to obtain it in the case of grids (see Section 4.2).

A subgraph H of some graph is an *ℓ-keyboard* if it is connected and satisfies the following properties. The vertex-set of H contains a set of ℓ pairwise distinct nodes $X = \{x_1, \cdots, x_\ell\} \subseteq V(H)$. For any $1 \leq i \leq \ell$, node x_i has two neighbors u_i and v_i in $V(H) \setminus X$, with $\{u_i, v_i\} \neq \{u_j, v_j\}$ for any $1 \leq i < j \leq \ell$. For any $1 \leq i, j \leq \ell$, there is a path $P_{ij} = (w_1, \cdots, w_{h_{ij}})$ from v_i to u_j in $H \setminus X$ such that, for all $b \leq \ell$, if $u_b = w_h$ for some $h \leq h_{ij}$ then $v_b \notin \{w_{h+1}, w_{h+2}\}$. Finally, for any $1 \leq i \leq \ell$, there is a walk $Q_i = (a_1, a_2, \cdots)$ (i.e., an infinite sequence of nodes such that two consecutive nodes are adjacent) in $H \setminus X$ such

that $a_1 = v_i$ (Q_i starts from v_i), $a_h \notin \{a_{h+1}, a_{h+2}\}$ for any $h \geq 1$ (no edge is repeated consecutively in Q_i), and, for any $b \leq \ell$, if $u_b = a_h$ for some $h \geq 1$, then $v_b \notin \{a_{h+1}, a_{h+2}\}$.

An example of 3-keyboard in given in Figure 2 and explained in the associate Claim 1.

In the following theorem, we use the same notations as in the above paragraph. The superscript will refer to the index of the considered subgraph. In the hypotheses of the theorem, we assume that the robots can distinguish k disjoint ℓ-keyboards in the graph and that they are deployed in such a way that each robots is on a different ℓ-keyboard. However, for the case when the graph is a grid and k is small enough, we will show that the robots can univocally identify k disjoint ℓ-keyboards by using only the initially available information and can move to their own ℓ-keyboard accordingly.

Theorem 1 *Let G be any n-node graph with k pairwise disjoint ℓ-keyboards $G^1, \cdots, G^k$ and k robots. For each $1 \leq i \leq k$, assume that each robot R_i occupies a node in $\{v_1^i, \cdots, v_\ell^i\}$ and that it can distinguish each node in G^j, for any $j \leq k$. Finally, let p be the maximum length of some path P_{jr}^i ($i \leq k$; $1 \leq j, r \leq \ell$). Then, there exists a pseudo-synchronized algorithm that solves the stigmergy problem by k robots in $O(p\mathcal{M})$ phases and using $O(pk)$ control-memory per robot, if p is known to all robots. Otherwise, it requires $O(p\mathcal{M}+n)$ phases and $O(nk)$ control-memory per robot. If quiescence is not required,$O(k)$ bits of control-memory is sufficient per robot.*

Proof. Let us first define the following simple algorithm. Consider a robot R_i with some message m_i and starting in v_j^i ($j \leq \ell$). The message will be encoded into the walk followed by the robot. Except for its first move, R_i moves only if it has seen all other robots to change their position since the previous move of R_i. Note that this can always be detected by R_i because we assume that any robot can distinguish each node in G^h for any $h \leq k$. We now define the walk that R_i must follow by recursion on the length of m_i.

- If $m_i = \emptyset$, then R_i follows the walk Q_j^i until it enters in the terminated-state (the conditions for which are described later).
- Otherwise, $m_i = \alpha \odot m_i'$ where $\alpha = \lambda_h \in \mathcal{L}$ is the first symbol of m_i and m_i' the remaining part of it. In this case, R_i follows the path P_{jh}^i to reach u_h^i. Then, R_i goes to x_h^i and then to v_h^i. We call this sequence of two moves the *writing* of α by R_i. Then, recursively, R_i follows the path defined similarly for m_i'.

We must now prove that such a protocol, executed concurrently by all robots, actually allows them to communicate. First, let us note that, by definition of the protocol and of the walk followed by all robots, the algorithm is pseudo-synchronized. Hence, by Lemma 1, for any two robots R and R', Robot R sees R' occupying at least one node every two consecutive nodes that R' actually occupies. Second, each Robot R_i occupies x_h^i only when writing the symbol

λ_h. Similarly, Robot R_i follows two consecutive edges $\{u_h^i, w\}$ and then $\{w, v_h^i\}$ only when writing the symbol λ_h (in which case $w = x_h^i$).

By the above two remarks, Robot R sees R_i at x_h^i or at u_h^i and v_h^i consecutively if and only if R_i is writing λ_h ($h \leq \ell$). Therefore, following this protocol, each robot actually receives the messages of all other robots. Finally, while it has not fully transmitted its message, a robot needs at most $p + 2 \leq n$ phases to transmit one symbol. Hence, if a robot has not received (i.e., detected) any symbol from any other robot during $p + 2$ (or n if p is unknown) phases, it can safely enter a quiescent-state. Concerning the control-memory, each robot only needs to remember the last 2 (resp., $p + 2$, resp., n) nodes occupied by each other robot if quiescence is not required (resp., if quiescence is required and p known, resp., p unknown). ■

3.2 Serial communication

In this section, we present a second generic algorithm allowing robots to communicate in the Look-Compute-Move model. Contrary to the algorithm proposed in previous section, it requires a non-constant time to transmit one symbol and only one robot can transmit a symbol at a time. Moreover, we consider only binary symbols. On the other hand, in some graph classes, this new algorithm can be executed when the number of robots is of order the number of nodes of the graph. Moreover, the control-memory required by each robot is constant.

As in the previous case, we will assume some particular structure, namely a large (i.e., order of the number of robots) cycle whose nodes are distinguishable. Moreover, in this section, we assume that the robots are initially occupying some desired configuration. Section 4.3 details a protocol allowing the robots to reach such a configuration in the case of grids.

Let $C = (v_1, \cdots, v_{|C|})$ be a cycle in a graph. The configuration of C (i.e., the positions of the robots restricted to the nodes of C) is denoted by the binary string $(a_1, \cdots, a_r)$ where $a_i = 1$ if v_i is occupied and $a_i = 0$ otherwise for any $i \leq r$. Given a binary string $B = (b_1, \cdots, b_r)$, if $b_j = x \in \{0, 1\}$ for any $i \leq j \leq h$, we note B by $(b_1, \cdots, b_{i-1}, x^{h-i+1}, b_{h+1}, \cdots, b_r)$. Finally, x^0 corresponds to an empty symbol.

For instance, the configuration of the cycle depicted in Figure 4c is denoted by $(0, 1^2, 0^3, 1^2, 0^5, 1, 0^6, 1, 0, 1, 0)$ where v_1 is the top-left corner.

The algorithm described in the next theorem is not pseudo-synchronized because robots may slide twice consecutively along the same edge. However, this algorithm ensures that, for any configuration, exactly one robot will move. Therefore, we can define, similarly as for pseudo-synchronized algorithm, a *phase* of the process as a minimal sequence of moves such that all robots have moved at least once during this sequence. In the hypotheses of the theorem, we assume that the robots are aware of some information about a cycle C. In Section 4.3, we will show how the robots on the grid can deduce such information from their initial observation.

Theorem 2 *Let G be a graph with a cycle C and $3 \leq k \leq |C| - 5$ robots and let us assume that the robots are able to identify the first node v_1 of C and all the edges of C. Then, there exists an algorithm that, starting from configuration $(1, 0, 1^{k-2}, 0^{|C|-k-5}, 1, 0^4)$, solves the stigmergy problem by k robots in $O(k^2\mathcal{M})$ phases. Moreover, each robot needs a control-memory of size $O(1)$ bits.*

Proof. First, notice that, by the hypotheses, each robot knows the cycle C and its starting node v_1, but there is no *a priori* agreement on sense of orientation (in particular, only v_1 needs to have an identifier). In what follows, the sense of orientation is deduced from the starting node v_1 and from the fact that $3 \leq k \leq |C| - 5$. Moreover, the reached configurations will always induce a sense of direction coherent with the initial one. Hence, for ease of description, we will assume that the nodes of C are labeled $(v_1, \cdots, v_{|C|})$ where the labeling corresponds to the initial configuration $(1, 0, 1^{k-2}, 0^{|C|-k-5}, 1, 0^4)$. In particular, the robot initially occupying v_1 will be called R and the robot initially occupying $v_{|C|-5}$ will be called R_{last}. Finally, we denote by R_i the robot initially occupying v_{i+1}, for any $2 \leq i \leq k - 1$.

The algorithm proceeds as follows in order for R to communicate its message to all other robots. Note that, in this algorithm, exactly one robot moves at a time. The fact that all configurations reached are asymmetric implies that there is no ambiguity on which robot has to move.

We distinguish among three types of configurations: those used to communicate a bit 1 by robot R; those used to communicate a bit 0 by robot R; and those used to communicate the end of transmission of robot R. The idea can be informally summarized as follows: If R wants to communicate a bit 1, it moves from v_1 to v_2 and then back to v_1; If R wants to communicate a bit 0, it moves from v_1 to $v_{|C|}$ and then back to v_1; If R wants to communicate the end of its transmission, it first moves from v_1 to v_2 and then from v_2 to v_3. Each movement of robot R is followed by a movement of each other robot R_i whose aim is to make R aware that R_i has seen the movement of R. When R communicates the end of its transmission, robot R_{last} takes the place of R by moving to v_1 and, in turn, each robot plays the role of R, thus solving the stigmergy problem.

In detail, first R goes either to v_2 (which will mean either the transmission of a 1 or that R has terminated its transmission) or to $v_{|C|}$ (which means the transmission of a 0). Hence, the configuration become either $(0, 1^{k-1}, 0^{|C|-k-5}, 1, 0^4)$ or $(0^2, 1^{k-2}, 0^{|C|-k-5}, 1, 0^3, 1)$. Note that, since $3 \leq k \leq |C| - 5$ and v_1 is well identified, there is no ambiguity.

Then, R_{last} moves from $v_{|C|-5}$ to $v_{|C|-4}$. This move initiates the first phase of acknowledgment of all other robots. The configuration reached is either $(0, 1^{k-1}, 0^{|C|-k-4}, 1, 0^3)$ or $(0^2, 1^{k-2}, 0^{|C|-k-4}, 1, 0^2, 1)$. Again, because $3 \leq k \leq |C| - 5$ and v_1 is well identified, there is no ambiguity.

Then, for $i = k - 1$ down to 2, the robot R_i will acknowledge that it has seen R in v_2 (resp., in $v_{|C|}$) by moving from v_{i+1} to v_{i+2}. Note that, if R is in $v_{|C|}$, then R_i can already interpret it as a 0. On the other hand, if R is in v_2, R_i cannot know yet whether it corresponds to a 1 or to the

end of the message of R. During this phase, the robots pass sequentially, for $j = 0$ to $k-2$, through the configurations $(0, 1^{k-1-j}, 0, 1^j, 0^{|C|-k-5}, 1, 0^3)$ (resp., through the configurations $(0^2, 1^{k-2-j}, 0, 1^j, 0^{|C|-k-5}, 1, 0^2, 1)$). Because $3 \leq k \leq |C| - 5$ and v_1 is well identified, there is no ambiguity. Hence, eventually, the configuration $(0, 1, 0, 1^{k-2}, 0^{|C|-k-5}, 1, 0^3)$ (resp., the configuration $(0^3, 1^{k-2}, 0^{|C|-k-5}, 1, 0^2, 1)$) is reached.

Now, there are two cases depending on whether the considered phase corresponds to the transmission of a bit by R or if it corresponds to the end of the transmission of R.

- Let us first assume that R aims at transmitting a bit during this phase. Therefore, as already mentioned, R being in v_2 means it transmits a 1 and R being in $v_{|C|}$ means it transmits a 0. In both cases, all robots have seen (since they all have moved) the position occupied by R (v_2 or $v_{|C|}$) and therefore they know the corresponding bit. Then, R moves back to v_1, obtaining configuration $(1, 0^2, 1^{k-2}, 0^{|C|-k-5}, 1, 0^3)$.

  Finally, all other robots acknowledge by moving back to their initial position. That is, for $i = 2$ to $k-1$, Robot R_i goes from v_{i+2} to v_{i+1}, and finally, R_{last} goes back from $v_{|C|-4}$ to $v_{|C|-5}$. The configurations met during this phase are: $(1, 0, 1^j, 0, 1^{k-2-j}, 0^{|C|-k-5}, 1, 0^3)$, for $j = 1$ to $k-2$, and finally the initial configuration is reached and R can start a new transmission.

- Let us assume that R wants to communicate the end of its transmission. The algorithm ensures all robots will be "rotated" in order to let the next robot, R_{last}, to transmit its own message.

  Note that, in that case, R must be on v_2 and the current configuration is $(0, 1, 0, 1^{k-2}, 0^{|C|-k-5}, 1, 0^3)$. Then, R moves to v_3 and the configuration $(0, 0, 1^{k-1}, 0^{|C|-k-5}, 1, 0^3)$ is reached. Then, R_{last} goes from its current position $v_{|C|-4}$ to v_1, passing through the configurations $(0^2, 1^{k-1}, 0^{|C|-k-5+j}, 1, 0^{3-j})$ for $j = 1$ to 3, and finally reaching $(1, 0, 1^{k-1}, 0^{|C|-k-1})$. For each configuration, there is no ambiguity because $3 \leq k \leq |C| - 5$ and v_1 is well identified.

  Finally, the robot occupying v_{k+1} moves to reach $v_{|C|-5}$, i.e., passing through the configurations, $(1, 0, 1^{k-2}, 0^j, 1, 0^{|C|-k-1-j})$ for $j = 0$ to $|C| - k - 5$. Here, the only possible ambiguity arises in the configuration $(1, 0, 1^{k-2}, 0, 1, 0^{|C|-k-2})$ where symmetry is broken because v_1 is well identified by assumption.

■

4 Stigmergy in grids

In this section, we focus on the special case of grids. Let G be an $N \times M$ grid occupied by k robots.

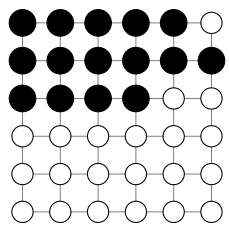

Figure 1: Configuration $\mathcal{C}^*$ in a 6× 6-grid with $k = 15$ robots

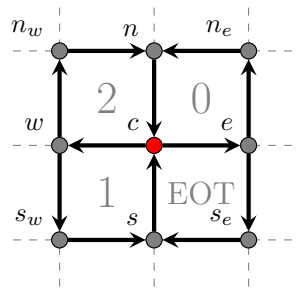

Figure 2: 3-*keyboard* in a grid.

We represent a *configuration* of the robots in G with an $N \times M$ matrix $\mathcal{C}$ with values in $\{0, 1\}$. The element $\mathcal{C}[i, j]$ is 1 if the corresponding position in the grid is occupied and it is 0 otherwise. As the robots have no sense of direction, from a configuration $\mathcal{C}$ they can obtain eight different matrix representations (depending on the first row/column). However, we consider only particular configurations, namely *asymmetric and aperiodic*, for which the representation is not ambiguous.

A configuration is *periodic* if it is invariant with respect to rotations of 90 or 180 degrees, where the rotation point coincides with the geometric center of the grid. A configuration is *symmetric* if it is invariant after a reflection with respect to a vertical, horizontal, or diagonal (in case of square grids) axis passing through the geometric center of the grid. In what follows, we will assume that the initial configuration is always *rigid*, i.e., aperiodic and asymmetric. Moreover, our algorithms ensure that, starting from a rigid configuration, the obtained configurations (after the robots' moves) are always rigid. For this purpose, our algorithms widely use the configuration $\mathcal{C}^*$ defined as follows.

Here we represent a configuration $\mathcal{C}$ as a *serialization* of the matrix $\mathcal{C}$, that is, as an array of the elements of $\mathcal{C}$. More precisely, a *row-by-row serialization* $R_{\mathcal{C}}$ of $\mathcal{C}$ is obtained by reading $\mathcal{C}$ starting from $\mathcal{C}[0, 0]$ and traversing it row by row, formally, $R_{\mathcal{C}} = (\mathcal{C}[0, 0], \mathcal{C}[0, 1], \ldots, \mathcal{C}[0, M-1], \mathcal{C}[1, 0], \mathcal{C}[1, 1], \ldots, \mathcal{C}[1, M-1], \ldots, \mathcal{C}[N-1, 0], \mathcal{C}[N-1, 1], \ldots, \mathcal{C}[N-1, M-1])$. Note that, when $\mathcal{C}$ is an aperiodic and asymmetric configuration, there is no ambiguity by defining $R_{\mathcal{C}}$ as the maximal row-by-row serialization in lexicographical order. Using this notation, let $\mathcal{C}^*$ be the configuration such that $R_{\mathcal{C}^*} = (1^k, 0^{N \cdot M - k})$ if $k < N$ or $R_{\mathcal{C}^*} = (1^{N-1}, 0, 1^{k-(N-1)}, 0^{N \cdot M - k})$ otherwise (see Fig. 1).

The next subsection shows that from any rigid initial configuration, the robots can reach configuration $\mathcal{C}^*$.

4.1 Achieving the starting configuration $\mathcal{C}^*$

Given any aperiodic and asymmetric configuration $\mathcal{C}$, let $R_{\mathcal{C}}$ be the maximal row-by-row serialization in lexicographical order and let c be the corner from which $R_{\mathcal{C}}$ starts.

In the following we first give an algorithm to reach C^* from any rigid configuration $\mathcal{C}$. We refer to Fig. 3 for a visualization of the algorithm.

R.1 If c is not occupied (i.e. no corner is occupied)

- **R.1.1** If there is no occupied node in the first row, $R_{\mathcal{C}} = (0^{N-1}, C')$:
  Move the robot in the first occupied node towards the first row (see Fig. 3a).
- **R.1.2** If there is at least an occupied node in the first row at distance greater than 1 from c, $R_{\mathcal{C}} = (0^x, 1, C')$, $1 < x < N$:
  Move the robot in the first occupied node towards c (see Fig. 3b).
- **R.1.3** If node next to c is occupied, $R_{\mathcal{C}} = (0, 1, C')$.
  Check whether moving towards c the robot in the node next to c leads to a symmetric configuration. In the affirmative case, let a be the first occupied node which is next to an empty node and is not in position $(0, 1)$, then move the robot in a according to the above rules (i.e. if a is not on the first row, move the robot towards the first row; if a is in the first row, move the robot towards c). Otherwise, move the robot in the first occupied node towards c (see Fig. 3c).

R.2 If c is occupied and the node $(0, 1)$ is not occupied.

- **R.2.1** If there are no occupied nodes on the first row:
  Move the robot in the first occupied node different from c towards the first row (see Fig. 3d).
- **R.2.2** If there are other occupied nodes on the first row:
  Move the robots on the first row towards c in order to have consecutive occupied nodes starting from c (see Fig. 3e).

R.3 If the first x nodes, $1 \leq x \leq N - 2$ nodes starting from c are occupied,

- **R.3.1** If there are other occupied nodes on the first row:
  Move the robots on the first row towards c (see Fig. 3f).
- **R.3.2** If no other nodes on the first row are occupied and all the corners different from c have less than x consecutive occupied nodes:
  Move the robot on the first occupied node of $R_{\mathcal{C}}$ next to a node which is not occupied towards the first node of $R_{\mathcal{C}}$ which is not occupied (see Fig. 3g).

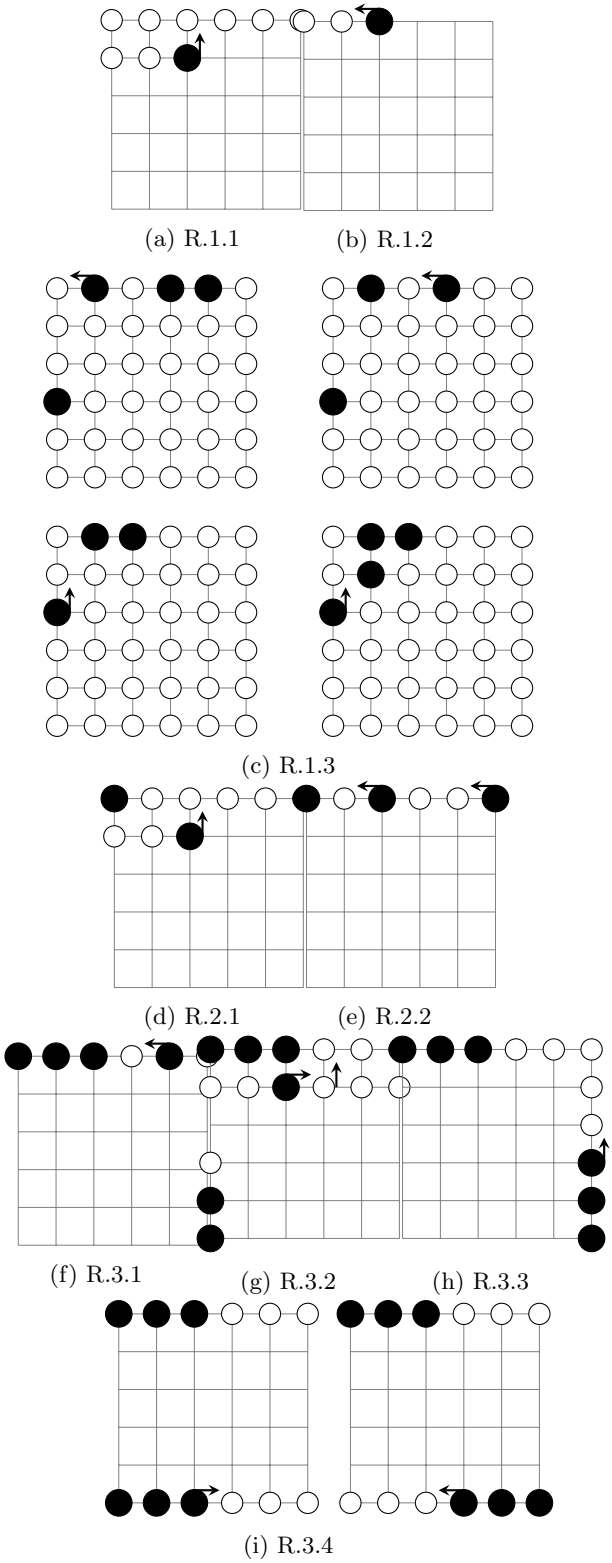

Figure 3: Algorithm to achieve C^*.

R.3.3 If no other nodes on the first row are occupied, there are x consecutive occupied nodes in the last column starting from the corner opposite to c w.r.t. the diagonal

Move the robot on the first node of the sequence of x occupied nodes on the last column towards its adjacent node not occupied on the same column (see Fig. 3h).

R.3.4 If no other nodes on the first row are occupied, there are x consecutive occupied nodes in the last row and the last column has less then x consecutive occupied nodes starting from the corner opposite to c w.r.t. the diagonal:

Move the robot on the last (first) node of the sequence of x nodes on the last row towards its adjacent node not occupied on the same row (see Fig. 3i).

R.4 If the first $N-1$ nodes starting from c are occupied and node $(0, N)$ is not occupied, then apply the same rules as above by using the first which is not fully occupied row instead of the first one.

R.5 If the first N nodes starting from c are occupied,

R.5.1 If all the nodes (i, N), $i \leq N$, are occupied, then move the node in position (x, j) to position $(x, j-1)$, where x is the first row which is not fully occupied and j is maximum.

R.5.2 If a node (i, N), $i \leq M$, is not occupied occupied, then move the node in position (i, N) to position $(i+1, N)$.

Theorem 3 *Given an $N \times M$ anonymous grid G with $\min\{N, M\} > 7$ and $k \leq N \cdot M - 5$ anonymous, asynchronous and uniform robots, there exists an algorithm that starting from any asymmetric and aperiodic configuration allows to achieve $\mathcal{C}^*$ in $O(kNM)$ phases. Moreover, no control memory is needed.*

Proof. Let C and C' be the configurations before and after each movement, then the statement follows from the following facts: (i) C' remains aperiodic and asymmetric (ii) the corner c which gives the maximal serializations $R_{\mathcal{C}}$ and $R_{\mathcal{C}'}$ is the same; (iii) the maximal serialization $R_{\mathcal{C}'}$ is smaller than $R_{\mathcal{C}}$ (but for rules R3.2, R3.4 and R.5 that will be addressed later in the proof) (iv) the maximal serialization $R_{\mathcal{C}^*}$ is the maximal possible among all the achieved configurations (but for those handled by Rule R.5).

Fact (iii) follows from the algorithm definition. Facts (i) and (ii) follow from Fact (iii) and by the observation that $R_{\mathcal{C}}$ is the unique maximum serialization. Fact (iv) follows from the definition of C^*. To conclude the first part of the proof it is enough to observe that when we apply rules R3.2, R3.4 and R.5, we still obtain that Facts (i)–(iii) hold after a finite number of steps instead of one step (i.e. C' is the configuration obtained after a finite number of movements instead of only one movement).

Therefore the algorithm works as follows. If the initial configuration is one of those handled by R.5, we apply the algorithm until we get a configuration in R.4. Note that R.5 can be applied only at the beginning since the algorithm does not achieve such configurations anymore. In any other rule, the algorithm increases the maximal serialization (possibly in more than one step in Rules R3.2, R3.4 and R.5) until it achieves the maximal possible that is C^*. In any of these steps the corner that gives the maximal serialization remains the same. ■

4.2 Parallel communication

In this section, we show how to fulfill the hypotheses used in Theorem 1 in the case of grids. Indeed, we show that, if k is small enough, the robots can univocally identify k disjoint 3-keyboards by using only the information on their initial disposal. Moreover, the robots can move to their own 3-keyboard by exploiting the algorithm in Theorem 3.

In detail, we consider any $N \times M$-grid containing at most $\lfloor NM/9 \rfloor$ robots in a rigid initial configuration. Informally, each robot will use some sub-grid, with at least 9 nodes, as a keyboard. Clearly, the less robots there are, the larger the sub-grid each robot can use. In what follows, to make the presentation easier, we consider a $3N \times 3M$-grid in which at most $NM - 2$ robots are initially occupying a rigid configuration. Our result can be extended to any $N \times M$-grid with at most $\lfloor NM/9 \rfloor - 2$ robots.

Following Theorem 1, we need first to assign some disjoint subgraphs to the robots. Each subgraph will be used as a keyboard by a robot. Then, we need to show that starting from any rigid initial configuration, each robot can uniquely identify the keyboard it is assigned to and then reach it before starting the communication phase. There are several difficulties here. First, we must ensure that each configuration achieved during the whole process is rigid, in order to avoid any ambiguity on the identification of the keyboards. Second, the main difficulty is to ensure that all robots identify the beginning of the transmission - roughly they must identify the "first" step when each robot has reach its own keyboard and is ready to transmit (i.e., the first step fulfilling the hypotheses of Theorem 1) - in order not to miss any information from any other robot.

We start with some easy claims that will be used to assign keyboards to robots. The next claim is illustrated in Fig. 2.

Claim 1 *A 3×3-grid can be used as a 3-keyboard.*

Proof. Let $G_{3\times3}$ be such a grid. Let c denote its central-node, and let n, e, s, w (North, East, South, West) be the neighbors of c. Finally, let n_e be the common neighbor (different from c) of n and e. We define similarly the nodes s_e, n_w and s_w (the last four nodes are the corners of $G_{3\times3}$).

Intuitively, the cycle (c, e, n_e, n) will be used to transmit a symbol 0, the cycle (c, w, s_w, s) can be used to transmit a symbol 1 and the cycle (c, w, n_w, n) to transmit a symbol 2. Finally, the cycle (c, e, s_e, s) will be used to signify

the end of the transmission. More formally, and using the same notation as in Section 3.1, let $X = \{x_1 = n_e, x_2 = s_w, x_3 = n_w\}$ and $u_1 = e, v_1 = v_3 = n, u_2 = u_3 = w$ and $v_2 = s$. Moreover, for any $i, j \leq 3$, $P_{i,j} = (v_i, c, u_j)$. Finally, $Q_i = (v_i, (c, e, s_e, s)^*) = (v_i, c, e, s_e, s, c, e, s_e, s, c...)$ for any $i \leq 3$. It is easy to check that this well defines a 3-keyboard. ■

Claim 2 *Let G be any $N \times M$-grid with* $\max\{N, M\} > 2$ *and* $2 \leq k \leq NM - 2$. *It is possible to obtain a rigid configuration by coloring exactly k nodes with* 0, *while any other is colored with* 1.

Proof. First, let us assume that $M \geq 4$ and $N \geq 5$. If $MN - k \geq 10$, then, color with 0 one of the corner and exactly one of its neighbor. The three other corners and all other neighbor of a corner (i.e., 10 nodes in total) are colored with 1. Regardless of the coloration of remaining nodes, the configuration is rigid. Otherwise, if $MN - k < 10$ (note that $k > 10$ because $MN \geq 20$), then start your coloration (i.e., for corners and their neighbors) as previously but reversing 0 and 1. Again, the configuration is rigid regardless of the coloration of remaining nodes.

The cases with smaller N and M can be dealt with using similar arguments. ■

Lemma 2 *Let G be any $3N \times 3M$-grid,* $\max\{N, M\} > 2$. *For any $2 \leq k \leq NM - 2$, there exist k vertex-disjoint 3-keyboards such that all configurations with exactly one robot per keyboard are rigid with a common orientation.*

Proof. Consider any orientation of G, i.e., choose the first row and first column. Let $v_{i,j}$ be the node in the ith row and j^{th} column of G ($1 \leq i \leq 3N$, $1 \leq j \leq 3M$). Let $G_{i,j}$ be 3×3-sub-grid with corners $\{v_{3i+1,3j+1}, v_{3i+3,3j+1}, v_{3i+3,3j+1}, v_{3i+3,3j+3}\}$ for any $1 \leq i \leq N$, $1 \leq j \leq M$. By previous claim, all the sub-grids $G_{i,j}$ are pairwise-disjoint 3-keyboards.

Let H be a $N \times M$-grid. By previous claim, it is possible to color the nodes black and white with exactly k black nodes and such that the obtained configuration $\mathcal{C}$ is rigid. For any $1 \leq i \leq N$, $1 \leq j \leq M$, let $h_{i,j}$ be the node in the ith row and j^{th} column of H.

Now, in G, let us choose all 3-keyboards $G_{i,j}$ such that h_i is colored black in $\mathcal{C}$. This choice of keyboard satisfies the desired property because, from any configuration with exactly one robot per chosen keyboard, it is possible to recover the coloration $\mathcal{C}$ of H and then the orientation. ■

We can now present our algorithm that allows $2 \leq k \leq MN - 2$ robots to solve the stigmergy problem in $3M \times 3N$-grids starting from any rigid configuration. Because the initial configuration is rigid, the robots can be assigned distinct identifiers (each robot being assigned the same identifier by any other robot) and can choose a common orientation of the grid G (i.e., same first column and first row). Then, we ensure that both these identifiers and sense of orientation are preserved.

By Lemma 2 (and its proof), the robots can agree on a set of k pairwise-disjoint 3-keyboards with common orientation. More precisely, they first have to agree on a pre-computed rigid coloration of an NM-grid with exactly k black nodes (such a unique coloration is pre-computed for any N, M, k). Using this rigid coloration and their common orientation of G, the robots agree on the 3-keyboards as described in the proof of Lemma 2. Moreover, it is easy to give the keyboards distinct identifiers (for instance, ordering them according to the coordinates of their top-left corners). Therefore, just by the rigidity of the initial configuration: each robot with ID i (the identifier obtained from the rigid configuration) can univocally identify a 3×3-sub-grid (the keyboard with ID i) that will be its own 3-keyboard.

The next phase of the algorithm is that each robot reaches its own 3-keyboard. We emphasize that this happens without modifying the orientation of the grid. Therefore, the destination of each robot remains consistent during the whole process. Moreover, until every keyboard becomes occupied by exactly one robot, we ensure that, in any configuration, exactly one robot can move. More precisely, the robots first achieve the configuration $\mathcal{C}^*$ using the algorithm of Theorem 3 (we prove there that, in any configuration, exactly one robot moves and that the orientation never changes). In configuration $\mathcal{C}^*$, the robots are ordered row by row, and from left to right. Therefore, starting from this configuration, each robot (in order) will reach the center of the 3-keyboards.

The last phase starts as soon as each 3-keyboard is occupied by exactly one robot. First note that, from this step, according to the algorithm of Theorem 1, each robot will remain in its own keyboard and, by Lemma 2, each reached configuration will be rigid. When a robot R sees for the very first time that each robot occupies some node of their own keyboard, first, R updates some local variable that indicates that the last phase has started. There are two cases: either R is occupying the neighbor of a node in $\{v_1, v_2, v_3\}$ in which case R moves to one of these nodes, or not (which means that R is occupying a node in $\{v_1, v_2, v_3, e\}$), in which case R moves to the center c.

Lemma 3 *Let us consider an $3N \times 3M$ anonymous grid G with $\max\{N, M\} > 2$ and $k \leq NM - 2$ anonymous, asynchronous and uniform robots forming an initial rigid configuration.*

The algorithm above allows to reach a configuration such that k pairwise disjoint 3-keyboards $G^1, \cdots, G^k$ are well identified by each robot, with a common orientation. Moreover, for any $1 \leq i \leq k$, Robot R_i is occupying some node in $\{v_1^i, v_2^i, v_3^i, c^i\}$.

From Theorem 1 and Lemma 3, the next theorem follows.

Theorem 4 *Let G be a $3N \times 3M$ anonymous grid with $\max\{N, M\} > 2$. Then, there exists an algorithm that, starting from any rigid configuration, solves the stigmergy problem by $k \leq NM - 2$ anonymous, asynchronous, and uniform robots in $O(\mathcal{M})$ phases. Moreover, each robot needs a control-memory of size $O(k)$ bits.*

Proof. Lemma 3 ensures that the robots can achieve the initial configuration required by Theorem 1. There is actually a small difference between the hypotheses of Theorem 1 and those of Lemma 3. More precisely, after applying the algorithm of Lemma 3, some robot may occupy the center c of its keyboard, which it not allowed in Theorem 1. However, it is easy to see that the whole proof of Theorem 1 works as well from c because $c \in \bigcap_{1 \leq i,j \leq 3} P_{i,j} \cap \bigcap_{1 \leq i \leq 3} Q_i$. ■

4.3 Serial communication

In this section, we show how to fulfill the hypotheses used in Theorem 2 in the case of grids. In particular, we show how the robots can identify the cycle C and its first node v_1 by exploiting only the information on the robots' disposal on the grid. Moreover, the robots are able to reach the required initial configuration by using the algorithm in Theorem 3. We first introduce some notation. If one of N or M is even, then G is said to be *even* and it admits an Hamiltonian cycle. Otherwise, G is *odd* and it has a maximum cycle of size $N \cdot M - 1$ (number of nodes). If G is even, we assume w.l.o.g. that M is even. We represent a configuration $\mathcal{C}$ via a *serialization* $S_{\mathcal{C}}$ of $\mathcal{C}$ (an array of the elements of the matrix $\mathcal{C}$) that follows a maximum cycle. Remember that we only consider rigid configuration and, therefore, a particular corner $\mathcal{C}[0,0]$ can always be identified. In what follows, a configuration $\mathcal{C}$ is represented by its *maximum cycle serialization* $S_{\mathcal{C}}$, that is, by reading the matrix $\mathcal{C}$ starting from $\mathcal{C}[0,0]$ and traversing it according to the Hamiltonian cycle given in Fig. 4a if the grid is even, or according to the maximum cycle given in Fig. 4b if the grid is odd (e.g., see Fig. 4c). Note that, whenever $\mathcal{C}$ is rigid, $S_{\mathcal{C}}$ is uniquely defined.

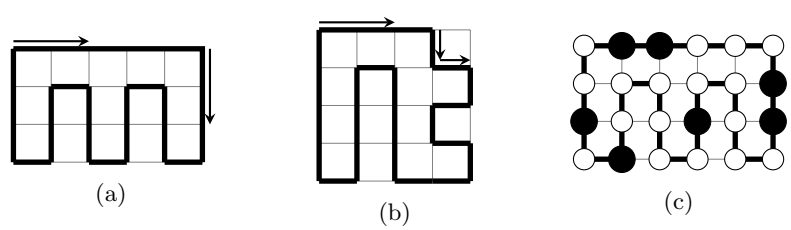

Figure 4: Maximum cycle serialization: (a) Hamiltonian cycle in a 4×6 grid. (b) Maximum cycle in a 5×5 grid. (c) Example on a 4×6 grid where white and black circles represent empty and occupied nodes, respectively and $S_{\mathcal{C}} = (0,1,1,0,0,0,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,0)$.

The given serialization allows us to fulfill the hypotheses of Theorem 2 in the case of grids. More precisely, Theorem 2 requires the robots to be able to distinguish a cycle in the grid G as well as a starting node v_1. It is then sufficient to show that, starting from any rigid configuration, the robots can actually identify a node v_1 and a cycle C. This can be done by the rigidity of the configuration. Indeed, by the above paragraph, in any rigid configuration,

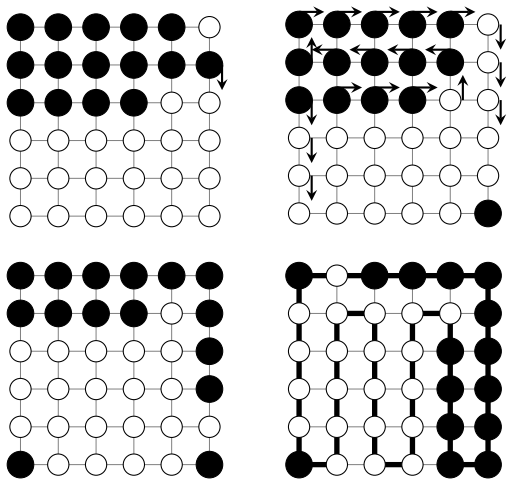

Figure 5: Algorithm to achieve $\mathcal{S}^*$ from $\mathcal{C}^*$.

the robots can identify a particular maximum cycle C with starting node v_1, and a particular corner of G (identified, again, via the rigidity of the configuration).

Moreover, the robots must be able to reach the configuration $\mathcal{S}^* = (1, 0, 1^{k-2}, 0^{|C|-k-5}, 1, 0^4)$ of C (where the representation is the maximum cycle representation). This is easily done as follows: first achieve the configuration $\mathcal{C}^*$ (following Theorem 3), and then reach configuration $\mathcal{S}^*$ (see Fig. 5).

Finally, it is easy to check that, in a grid, in the configurations reached during the algorithm described in the proof of Theorem 2, the cycle C and node v_1 remain well identified without ambiguity (see Figures 6–10 in Appendix). Altogether, we can state the theorem below, the proof of which is given in Appendix along with additional details on the algorithm.

Theorem 5 *Let G be an $N \times M$ anonymous grid G with* $\min\{N, M\} > 7$*. Then, there exists an algorithm that, starting from any rigid configuration, solves the stigmergy problem by $3 \leq k \leq N \cdot M - 5$ anonymous, asynchronous, and uniform robots in $O(k^2)$ phases. Moreover, each robot needs a control-memory of size $O(1)$ bits.*

5 Conclusion

The paper has presented the first study of stigmergic communication of mobile robots through a bee dance in a discrete environment. All algorithms assume the environment to be asynchronous, and apply a technique called pseudo-synchronization to cope with the asynchrony.

In particular, we have identified sufficient conditions on arbitrary graphs to allow stigmergy, and have proposed two algorithms. The first one requires that each robots has access to an ℓ-keyboard; a structure defined in the paper. It

allows the robots to communicate their bits in parallel and does not require the graph to be connected, but it is limited in the robot density that it can support. The second one requires the existence of a Hamiltonian path in the graph and solves the problem sequentially even when density is high.

We have also studied the problem further in a grid, allowing to relax some of the assumptions made for the general case. The grid indeed provides information on orientation unavailable in a more general graph, that the algorithm can exploit.

The paper provides sufficient conditions to solve the problem, but it is not known at this time whether these conditions are tight or what the lower bounds are. This leaves open the question to prove the necessity of the conditions, or to exhibit algorithms working under weaker assumptions.

References

[1] E. Bonabeau. Editor's introduction: Special issue on stigmergy. *Artificial Life*, 5(2):95–96, 1999.

[2] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[3] Z. Bouzid, S. Dolev, M. Potop-Butucaru, and S. Tixeuil. Robocast: Asynchronous communication in robot networks. In *Proc. OPODIS 2010*, number 6490 in LNCS, pages 16–31, 2010.

[4] Y. Dieudonné, S. Dolev, F. Petit, and M. Segal. Deaf, dumb, and chatting robots: Enabling distributed computation and fault-tolerance among stigmergic robots. In *Proc. 13th Intl. Conf. on Principles of Distributed Systems (OPODIS)*, volume LNCS 5923, pages 71–85, December 2009.

[5] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Hard tasks for weak robots: The role of common knowledge in pattern formation by autonomous mobile robots. In *Proc. 10th Intl. Symp. on Algorithms and Computation (ISAAC'99)*, volume 1741 of *LNCS*, pages 93–102, Chennai, India, December 1999. Springer.

[6] J. L. Gould. Honey bee recruitment: the dance-language controversy. *Science*, 189(4204):685–693, 1975.

[7] D. Karaboga and B. Akay. A survey: algorithms simulating bee swarm intelligence. *Artif. Intell. Rev.*, 31(1–4):61–85, 2009.

[8] A. Martinoli, A. J. Ijspeert, and F. Mondada. Understanding collective aggregation mechanisms: From probabilistic modeling to experiments with real robots. *Robotics and Autonomous Systems*, 29(1):51–63, 1999.

[9] F. Mondada, L. M. Gambardella, D. Floreano, S. Nolfi, J.-L. Deneuborg, and M. Dorigo. The cooperation of swarm-bots: physical interactions in

collective robotics. *IEEE Robotics & Automation Magazine*, 12(2):21–28, June 2005.

[10] E. Sahin and A. Winfield. Special issue on swarm robotics. *Swarm Intelligence*, 2(2–4):69–72, December 2008.

[11] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. Comp.*, 28(4):1347–1363, 1999.

A. Serial communication in grids.

In this section, we follow the algorithm outlined in Theorem 2 to devise an algorithm for the stigmergy problem which uses weaker assumptions in the case of grids. In particular, Theorem 2 assumes that: (i) each robot distinguishes each node of the cycle C; (ii) each robot distinguishes a unique starting node v_1 of the cycle C; (iii) for any node $v \in C$, each robot distinguishes the neighbors of v in the cycle C; (iv) the robots occupy a specified initial configuration. In the algorithm given in this section, we can assume that all the nodes and all the robots in the grid are anonymous and that the robots can initially occupy any aperiodic and asymmetric configuration. Moreover, robots have no sense of orientation and no memory of the orientations used in the previous snapshots. The algorithms given exploit the graph structure of the grid and the aperiodicity and asymmetry of the configurations to derive the hypotheses of Theorem 2.

In the following, we give the algorithm to solve the stigmergy problem starting from a set of some given configurations, and we observe that one of such configurations can be achieved from any rigid configuration by exploiting the algorithm given in the previous section. In particular, it is easy to reach configuration $\mathcal{S}^*$ from configuration $\mathcal{C}^*$ by using the algorithm described in Fig. 5.

In this section we use the maximum cycle serialization to represent a configuration. In the notation used in Theorem 2, the cycle used in the serialization is denoted by C while the upper left corner is denoted by v_1. We assume that we are in one of the configurations listed in the following. Note that each of such configurations is not ambiguous, that is, there exists one and only one pair of corner and serialization that matches the given pattern.

- Starting configuration (see e.g. Fig. 6): $\mathcal{S}^*$: $S_{\mathcal{C}}(\mathcal{S}^*) = (1, 0, 1^{k-2}, 0^{|C|-k-5}, 1, 0^4)$
- One configurations (see e.g. Fig. 7):
  - $\mathcal{O}.1$: $S_{\mathcal{C}}(\mathcal{O}.1) = (0, 1^{k-1}, 0^{|C|-k-5}, 1, 0^4)$
  - $\mathcal{O}.2$: $S_{\mathcal{C}}(\mathcal{O}.2) = (0, 1^{k-1}, 0^{|C|-k-4}, 1, 0^3)$
  - $\mathcal{O}.3$: $S_{\mathcal{C}}(\mathcal{O}.3) = (0, 1^x, 0, 1^{k-1-x}, 0^{|C|-k-5}, 1, 0^3)$, for $2 \leq x \leq k-2$
  - $\mathcal{O}.4$: $S_{\mathcal{C}}(\mathcal{O}.4) = (0, 1, 0, 1^{k-2}, 0^{|C|-k-5}, 1, 0^3)$
- Zero configurations (see e.g. Fig. 8):
  - $\mathcal{Z}.1$: $S_{\mathcal{C}}(\mathcal{Z}.1) = (0, 0, 1^{k-2}, 0^{|C|-k-5}, 1, 0^3, 1)$
  - $\mathcal{Z}.2$: $S_{\mathcal{C}}(\mathcal{Z}.2) = (0, 0, 1^{k-2}, 0^{|C|-k-4}, 1, 0^2, 1)$
  - $\mathcal{Z}.3$: $S_{\mathcal{C}}(\mathcal{Z}.3) = (0, 0, 1^x, 0, 1^{k-2-x}, 0^{|C|-k-5}, 1, 0^2, 1)$, for $1 \leq x \leq k-3$
  - $\mathcal{Z}.4$: $S_{\mathcal{C}}(\mathcal{Z}.4) = (0, 0, 0, 1^{k-2}, 0^{|C|-k-5}, 1, 0^2, 1)$
- Ack configurations (see e.g. Fig. 9):
  - $\mathcal{A}.1$: $S_{\mathcal{C}}(\mathcal{A}.1) = (1, 0, 0, 1^{k-2}, 0^{|C|-k-5}, 1, 0^3)$
  - $\mathcal{A}.2$: $S_{\mathcal{C}}(\mathcal{A}.2) = (1, 0, 1^x, 0, 1^{k-2-x}, 0^{|C|-k-5}, 1, 0^3)$, for $1 \leq x \leq k-3$

  - $\mathcal{A}.3$: $S_{\mathcal{C}}(\mathcal{A}.3) = (1, 0, 1^{k-2}, 0^{|C|-k-4}, 1, 0^3)$, for $1 \leq x \leq k-3$

- Break configurations (see e.g. Fig. 10):
  - $\mathcal{B}.1$: $S_{\mathcal{C}}(\mathcal{B}.1) = (0, 0, 1^{k-1}, 0^{|C|-k-5}, 1, 0^3)$
  - $\mathcal{B}.2$: $S_{\mathcal{C}}(\mathcal{B}.2) = (0, 0, 1^{k-1}, 0^{|C|-k-5+x}, 1, 0^{3-x})$, for $1 \leq x \leq 3$
  - $\mathcal{B}.3$: $S_{\mathcal{C}}(\mathcal{B}.3) = (1, 0, 1^{k-1}, 0^{|C|-k-1})$
  - $\mathcal{B}.4$: $S_{\mathcal{C}}(\mathcal{B}.4) = (1, 0, 1^{k-2}, 0^{|C|-k-1-x}, 1, 0^x)$, for $5 \leq x \leq |C|-k-2$

The robot r_i on the upper left corner (v_1) is the one allowed to communicate, as outlined in Theorem 2. To transmit a one (resp. zero) bit, the algorithm moves the robots according to the following sequence of configurations:

- from $\mathcal{S}^*$ to $\mathcal{O}.1$,
- from $\mathcal{O}.i$ to $\mathcal{O}.i+1$ (resp. from $\mathcal{Z}.i$ to $\mathcal{Z}.i+1$), $i = 1, 2, 3$,
- from $\mathcal{O}.4$ (resp. $\mathcal{Z}.4$) to $\mathcal{A}.1$
- from $\mathcal{A}.i$ to $\mathcal{A}.i+1$, $i = 1, 2$,
- from $\mathcal{A}.3$ to $\mathcal{S}^*$.

To communicate m_i, the algorithm transmits the sequence of bits representing m_i, followed by the sequence:

- from $\mathcal{S}^*$ to $\mathcal{O}.1$,
- from $\mathcal{O}.i$ to $\mathcal{O}.i+1$, $i = 1, 2, 3$,
- from $\mathcal{O}.4$ to $\mathcal{B}.1$,
- from $\mathcal{B}.i$ to $\mathcal{B}.i+1$, $i = 1, 2, 3$,
- from $\mathcal{B}.4$ to $\mathcal{S}^*$.

At this point, the next robot is allowed to communicate and, in turn, all the robots will be allowed to communicate.

All the above movements are done by following the maximum cycle corresponding to the serialization but for the case of even grids where $k = |C| - 2 \cdot M + 2$ or $k = |C| - 3 \cdot M + 4$ where the movement from $\mathcal{B}.4$ to $\mathcal{S}^*$ is done by avoiding the symmetric configurations $(1, 0, 1^{|C|-2\cdot M}, 0^{M-1}, 1, 0^{M-2})$ and $(1, 0, 1^{|C|-3\cdot M+2}, 0^{2M-3}, 1, 0^{M-2})$, respectively.

The next lemma shows that the hypotheses of Theorem 2 are fulfilled by the above algorithm without assuming that the robots know the nodes of cycle C.

Lemma 4 *Given an $N \times M$ anonymous grid G with $\min\{N, M\} > 7$ and $k \leq N \cdot M - 5$ anonymous, asynchronous and uniform robots, the above set of configuration is such that: (i) each robot distinguishes each node of the cycle C; (ii) each robot distinguishes a unique starting node v_1 of the cycle C; (iii) for any node $v \in C$, each robot distinguishes the neighbors of v in the cycle C.*

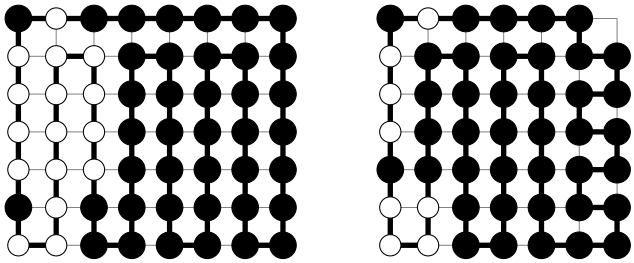

Figure 6: Starting configurations with $(N, M) = (7, 8)$ and $k = 40$ (left), and $(N, M) = (7, 7)$ and $k = 40$ (right).

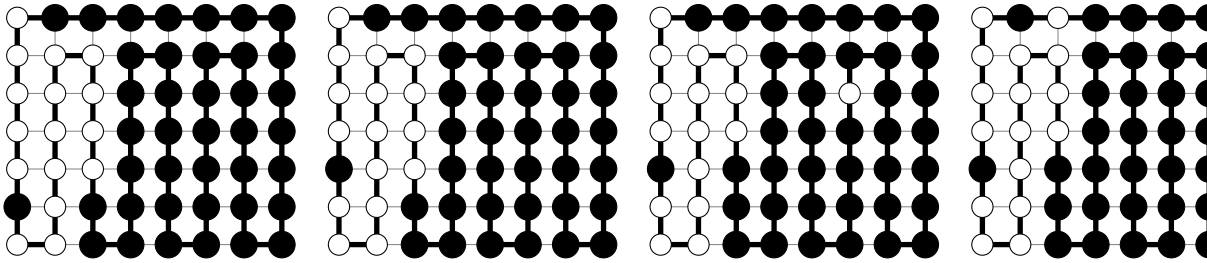

Figure 7: One configurations with $(N, M) = (7, 8)$ and $k = 40$.

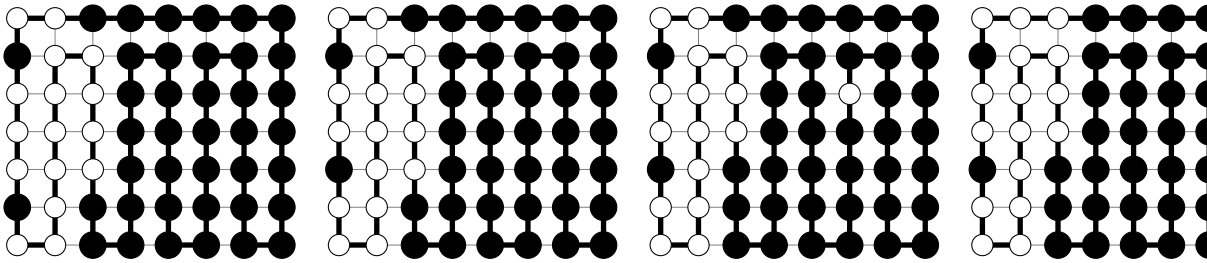

Figure 8: Zero configurations with $(N, M) = (7, 8)$ and $k = 40$.

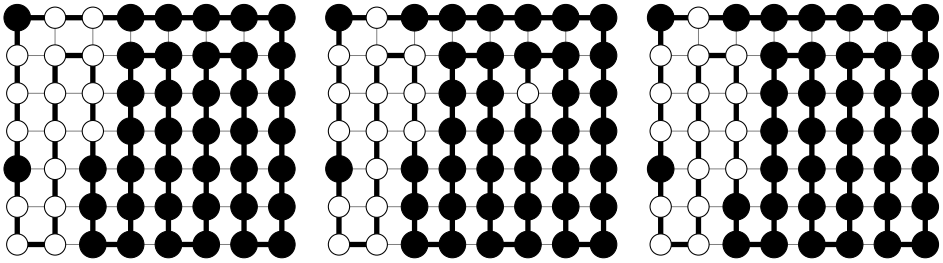

Figure 9: Ack configurations with $(N, M) = (7, 8)$ and $k = 40$.

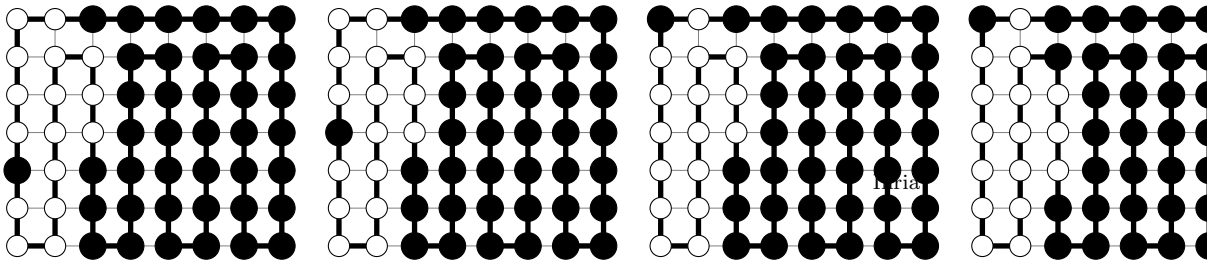

Figure 10: Break configurations with $(N, M) = (7, 8)$ and $k = 40$.

Proof. To show the statement, we prove that any node of the maximum cycle used in the serialization can always be univocally computed by each robot. Indeed, it is enough to observe that for $k \leq N \cdot M - 5$ each generated configuration is aperiodic and asymmetric and that it matches only one of the possible eight serializations of the grid. ■

The next theorem we show how to solve the stigmergy problem starting from $\mathcal{S}^*$, and, in the reminder of the section, we show how to achieve $\mathcal{S}^*$ configuration from any aperiodic and asymmetric configuration.

Theorem 5 *Let G be an $N \times M$ anonymous grid G with $\min\{N, M\} > 7$. Then, there exists an algorithm that, starting from any rigid configuration, solves the stigmergy problem by $3 \leq k \leq N \cdot M - 5$ anonymous, asynchronous and uniform robots in $O(k^2)$ phases. Moreover, each robot needs a control-memory of size $O(1)$ bits.*

Proof. The first part of the proof follows from Theorems 2 and 3, and Lemma 4. To show the second part of the proof, we observe that the defined configurations are all distinguished in a way that each robot can determine the phase of the algorithm at any point in time. Therefore, no state must be recorded and the only control-memory needed is for the robot that is communicating to record the bits that it sent. ■

In the next paragraph we comment on some particular special cases.

.0.1 Special cases with $N \leq 7$.

With small values of N there exist some special cases that can generate symmetries. Such cases are listed in the following.

- $(N, M) = (6, 6)$, $k = N \cdot M - 10 = 26$ or $k = N \cdot M - 14 = 22$, where $\mathcal{S}^*$ is symmetric;
- $(N, M) = (5, 2p)$, $k = N \cdot M - 5$ or $k = N \cdot M - 8$, where $\mathcal{O}.1$ is symmetric.
- $(N, M) = (4, 2p)$, $k = N \cdot M - 6$, where $\mathcal{O}.2$ is symmetric.
- $(N, M) = (4, 2p)$, $k = N \cdot M - 10$, where $\mathcal{O}.4$ is symmetric.
- $(N, M) = (7, 2p)$, $k = 7(2p - 2) + 2 = N \cdot M - 12$ or $k = 7(2p - 3) + 4 = N \cdot M - 17$, where $\mathcal{Z}.1$ is symmetric
- $(N, M) = (6, 2p)$, $k = 6(2p - 2) + 2 = N \cdot M - 10$ or $k = 6(2p - 3) + 4 = N \cdot M - 14$, where $\mathcal{Z}.2$ is symmetric
- $(N, M) = (6, 2p)$, $k = 6(2p - 3) + 2 = N \cdot M - 16$, where $\mathcal{Z}.4$ is symmetric
- $(N, M) = (5, 2p)$, $k = N \cdot M - 13$, where $\mathcal{A}.1$ is symmetric
- $(N, M) = (5, 2p)$, $k = N \cdot M - 14$, where $\mathcal{A}.2$ is symmetric

- $(N, M) = (5, 2p)$, $k = N \cdot M - 8$ or $k = N \cdot M - 11$, where $\mathcal{A}.3$ is symmetric

However, it is possible to avoid such symmetric configurations by considering specific movements for each of them. As an example, we can avoid to be symmetric by using a set of configurations where the last robot in the maximum cycle is shifted backwards by one position. In this case we have that, e.g., $S_C(\mathcal{S}^*) = (1, 0, 1^{k-2}, 0^{n-k-4}, 1, 0^3)$ instead of $S_C(\mathcal{S}^*) = (1, 0, 1^{k-2}, 0^{n-k-5}, 1, 0^4)$. Therefore, in the cases where the grids are rectangular, there is no ambiguity in determining the maximum cycle in any configuration. Coping with such configurations is out of the scope of this paper.

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399