Compacting and Grouping Mobile Agents on Dynamic Rings

Shantanu Das[1], Giuseppe Di Luna[2⋆], Linda Pagli[3], and Giuseppe Prencipe[3]

[1] Aix-Marseille University, CNRS, LIS, Marseille, France
[2] CINI, Italy
[3] Department of Computer Science, Università di Pisa, Pisa, Italy

Abstract. We consider computations by a distributed team of autonomous mobile agents that move on an unoriented dynamic ring network. In particular, we consider 1-interval connected dynamic rings (i.e. at any time, at most one of the edges might be missing). The agents move according to a Look-Compute-Move life cycle, under a synchronous scheduler. The agents may be homogenous (thus identical and monochromatic) or they may be heterogenous (distinct agents have distinct colors from a set of $c \geq 1$ colors). For monochromatic agents starting from any dispersed configuration we want the agents to form a compact segment, where agents occupy a continuous part of the ring and no two agents are on the same node – we call this the Compact Configuration Problem. In the case of multiple colors ($c > 1$), agents of the same color are required to occupy continuous segments, such that agents having the same color are all grouped together, while agents of distinct colors are separated. These formation problems are different from the classical and well studied problem of *Gathering* all agents at a node, since unlike the gathering problem, we do not allow collisions (each node may host at most one agent of a color).
We study these two problems and determine the necessary conditions for solving the problems. For all solvable cases, we provide algorithms for both the monochromatic and the colored version of the compact configuration problem, allowing for at most one intersection between the colored segments (which cannot be avoided in a dynamic ring). All our algorithms work even for the simplest model where agents have no persistent memory, no communication capabilities and do not agree on a common orientation. To the best of our knowledge this is the first work on the compaction problem in any type of dynamic network.

1 Introduction

Research in the field of distributed computing has always considered fault tolerance as an important aspect of algorithm design and there are many studies on algorithms tolerating e.g. failures of nodes or links in a network. However,

⋆ This work was performed while the author was affiliated with Aix-Marseille University, France.

in recent years researchers started to investigate so called *dynamic graphs*, that is graphs where the topological changes are not localized and sporadic; on the contrary, the topology changes continuously and at unpredictable locations, and these changes are not anomalies (e.g., faults) but rather an integral part of the nature of the system [4,9,10,17]. The study of distributed computations in such dynamic graphs has concentrated on problems of information diffusion, reachability, agreement, and other communication problems (see e.g., [1, 3, 6, 11, 14–16, 20]). These studies are on message passing networks under various different models of dynamic changes of topology. A general theoretical model for dynamic networks is the evolving graph model, where the network is modelled as a sequence of graphs each of which is a subgraph of the so-called footprint graph which represents the underlying topology. In order to allow useful tasks to be performed on such a network, we need to make some assumptions on the connectivity of the network. One natural way of modelling this is the k-interval connected dynamic graph model (See e.g. [17]). The most restricted of these models is the 1-interval connected dynamic graph model where the only assumption is that at each round, the instance of the graph is connected.

One of the ways of dealing with a highly dynamic environment is the use of mobile code, allowing processes to migrate from node to node on a network during the process of computation. This initiated research in algorithms for mobile agents, where an agent is an autonomous process that moves along the edges of the a network and can perform computations at the nodes of the network, using its own memory and state information, as well as the information stored in the nodes. Mobile agents can also represent agents moving in a dynamic environment. In this case, the agents may have some vision allowing them to see parts of the network and take decisions based on this knowledge. There are many different models for mobile agents depending on their capabilities of remembering (memory), visualizing (vision range), communication and computation abilities.

There has been a lot of research on mobile agents moving in static graphs. The fundamental problems studied are exploration and patrolling, where a team of agents has to visit all nodes of the graph, either once or periodically. A related problem is information dissemination or data collection from the nodes. Several coordination problems for teams of agents have been studied where the agents need to form a particular configuration. One of the most studied problem is rendezvous or gathering where all agents need to meet at a single node of the graph. This requires mechanisms for symmetry breaking as in the leader election problem in distributed computing. The problem has been studied both for agents with identities or anonymous (and thus identical) agents. For homonymous agents (where multiple agents share the same name or color), the problem of grouping the agents into teams with specific colors, is called the team assembling problem, and has been proposed and studied in [18] for agents moving freely in a plane. In the above problems, all agents of the same team must be at the same point or at the same node of the graph. However, it may not always be possible for a single node to host many agents at the same time. In this paper, we avoid multiple agents in the same node, but we want the agents in a team to be close to each other (e.g. to be able to exchange information and coordinate with each other). Motivated by this requirement, we define and study the Compact Configuration Problem (CCP) problem: starting from any configuration of mobile agents scattered in a graph G, the objective is to reach a configuration where each node contains at most one agent and the nodes occupied by agents of the same color induce a connected subgraph of G. To the best of our knowledge, this is the first time compaction problems have been studied at least for distributed teams of autonomous agents. As a preliminary investigation in this paper we consider one of the simplest topology - the ring network. In a ring, solving the CCP problem requires agents of the same team to occupy the nodes of a continuous segment of the ring, without any multiplicities. Although conceptually simple, a ring is highly symmetrical, and it is challenging to solve problems in the ring that require symmetry breaking. We assume that neither the nodes or the agents possess any unique identifiers, which makes the problem much harder. Moreover we consider the network to be dynamic where at any stage of the algorithm, some edges may be unavailable. In this paper the network is a 1-interval ring network, at most one edge of the ring may be missing at any round of the algorithm.

Previous studies of mobile agents in dynamic graphs has focussed on the fundamental problems of exploration, patrolling and gathering [5, 7, 8, 12, 13]. All these results consider t-interval connected graphs. Under weaker models of dynamicity, only weaker versions of gathering may be solved [2]. The problem of compaction is loosely related to the problem of *near-gathering* that has been studied recently in [19].

Our Contribution. In this paper we investigate the problem of compacting groups of mobile agents initially scattered on *dynamic rings*. We study the problem in two different scenarios: the agents either have all the same color ($c = 1$) or, there are $c > 1$ colors. We show that only local visibility is not sufficient for solving the problem even if the agents have unbounded memory. On the other hand, under global visibility, even oblivious agents (agents with no persistent memory) can solve the problem in all solvable instances. However, due to the dynamicity of the graph, we cannot always avoid intersections between the compacted segments. Our algorithms solve the CCP problem for many colors, with at most one intersection between two colored compact segments, while all other segments are separated. The results of this paper provide the full characterization of solvable instances for the above problems. Due the space limitations, some of the proofs have been omitted.

2 Preliminaries

Interval Connected Ring. A dynamic graph $\mathcal{G}$ is an infinite sequence of static graphs $(G_0, G_1, \ldots)$. For each round $r \in \mathbb{N}$ we have a graph $G_r : (V, E(r))$ where $V : \{v_0, \ldots, v_{n-1}\}$ is a set of nodes and $E : \mathbb{N} \to V \times V$ is a function mapping a round r to a set of undirected edges. Given a dynamic graph $\mathcal{G}$, its footprint

G is the graph obtained by the union of all graph instances $G = (V, E_\infty) = (V, \cup_{i=0}^{+\infty} E(i))$. A dynamic graph $\mathcal{G}$ is a 1-interval connected ring if its footprint is a ring and G_r is connected, for each round r. In this paper, we assume 1-interval connected ring such that at most one edge of the ring can be missing at any time; such an edge is arbitrarily chosen by an adversary. Throughout the paper we use the term *dynamic ring* to always mean such a network. The graph $\mathcal{G}$ is anonymous, i.e. all nodes are identical to the agents, the endpoints of each edge are unlabelled, and we do not assume any common orientation (i.e the ring is not oriented).

The agents. We consider a set of oblivious agents, $A = \{a_1, \ldots, a_k\}$ that are initially located on distinct nodes of a dynamic ring. The agents have no persistent memory, and each agent has an initial color in $[1, c]$ (when $c = 1$, all agents have the same color). When $c > 1$, we assume that the sets of agents having the same color all have the same size h, with $h > 2$. Also, we assume that the size of the ring is at least $2hc + c$. Also, we assume a total ordering on the colors; we call *max_color* the first color in this ordering. Note that the color of the agents is fixed at the beginning and it cannot be changed.

Agents follow the same algorithm executing a sequence of Look, Compute, Move cycles. In the Look phase of each cycle, the agent gets a *snapshot* of the environment. In the Compute phase the agent uses the information from the snapshot and the contents to compute the next destination, which may be the current node or one of its neighbours. During the Move phase an agent traverses an edge to reach the destination node. Given a direction of movement, we say that an agent a is *blocked* by the missing edge, if the edge adjacent to a, in the chosen direction of movement, is missing. We say that two agents *collide* if they occupy the same node at the same round. When two (or more) agents with distinct colors occupy the same node, we say that the collision is *admissible.*

The visibility of the agents may be either global or local:

– Global Snapshot: The snapshot obtained by an agent in round r contains the graph G_r (with the current location of the agent marked), and $\forall v \in G_r$, the colors of the agents (if any) that are located in node v.
– Local Snapshot: The snapshot obtained by an agent at a node v in round r contains the same information as in the Global snapshot, but restricted to a distance of R hops from node v.

Synchronous system. The system is *synchronous*, agents perform each (Look, Compute, Move) cycle in a discrete time unit called round. Rounds are univocally mapped to numbers in $\mathbb{N}$, starting from 0. All agents start the execution at round 0. In each round, each agent in A executes exactly one entire (Look,Compute, Move) cycle.

Configurations and other definitions. The configuration of the set of agents A at round r, is a function $C_r : A \to V$ that maps agents in A to nodes of V where agents are located. The term initial configuration indicates the configuration of

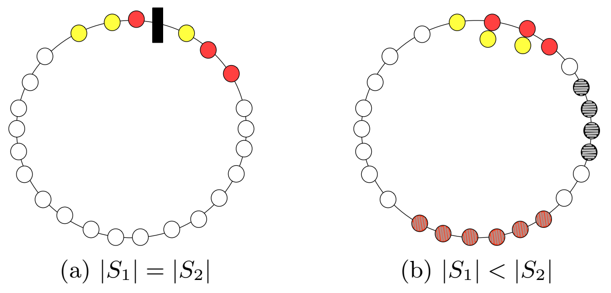


Fig. 1: (a) Impossibility with no overlap. (b) A Solution for ColoredCCP problem.

agents at round 0, when it is clear from the context we omit the round and we use C to indicate the current configuration. We use the notation $C_r(A)$ to indicate the set of nodes where agents in A are located at round r, and we use $G[C_r(A)]$ to indicate the subgraph induced by the locations of agents in A in graph G.

A *segment* indicates a set of nodes of $\mathcal{G}$ that have connected footprint and that do not form a cycle. Given a node $v \in \mathcal{G}$ we say that the node is *empty* at round r, if in C_r there is no agent on v. Similarly, we say that a segment of nodes is *empty* at round r if all nodes of the segment are empty. We say that a segment is *full* if each node of the segment contain agents of the same color. Given two full segments S_1 and S_2, let a be any agent in S_1 and b any agent in S_2; we define the *distance* between S_1 and S_2 as the smallest number of consecutive empty nodes between a and b. We say that a full segment S is *blocked* by the missing edge if the first agent in S is blocked according to the chosen direction of movement. Also, the "full segment" is said to move when all agents in the segment do a move in a given direction. Given two disjoint segments the distance between them is the minimum number of nodes between two endpoints of the segments.

Any given configuration at a round r can be represented by a sequence of n sets, representing the contents of the n nodes of the ring, starting from any given node. The configuration is said to be: (1) **periodic** if this sequence is periodic, (2) **Palindrome** if some cyclic rotation of this sequence is a palindrome, and (3) **Asymmetric** if it is neither Periodic nor Palindrome.

The Compact Configuration Problem. We introduce the problems we will investigate in the following. The first definition is for monochromatic agents.

Definition 1 (Compact Configuration Problem). *Given a dynamic graph $\mathcal{G}$ with footprint G and a set of agents A, we say that an algorithm solves the distributed* Compact Configuration Problem (CCP) *if and only if there exists a round r, when $G[C_r(A)]$ is connected and each agent occupies a distinct node.*

For multi-colored agents, we want agents of the same color to occupy continuous segments, while agents of distinct colors should be separated. Consider the example configuration in Figure 1a where agents of two colors are interleaved. Suppose the adversary blocks the edge marked with a dash, forever. In this case the only way to solve the problem would require two pairs of agents of two distinct colors to cross each other along the other continuous segment of the ring. Now during this process the agents would form an interleaved configuration in another part of the ring and now the adversary can block a new edge (releasing the previously blocked edge) in such a way that a similar configuration as in Figure 1a is created again.Thus it is not possible to completely segregate the agents of different colors under such an adversary. We therefore allow at most one overlap between two segments of different colors as in Figure 1b.

Definition 2 (Colored Compact Configuration Problem). *Given a dynamic graph $\mathcal{G}$ with footprint G and set of agents A_i having color $i \in [1, c]$, where $c \geq 2$, we say that an algorithm solves the distributed* Colored Compact Configuration Problem *(*ColoredCCP*) if and only if there exists a round r where, for each $i \in [1, c]$ except two colors j, p, each agent in A_i occupies a different node and $G[C_r(S_i)]$ is connected. Moreover, if $p \neq j$ it holds that $G[C_r(S_p)]$ and $G[C_r(S_j)]$ intersect.*

Intuitively, in the CCP problem we ask all agents, initially arbitrarily placed, to move so to form one full segment (i.e., with no empty nodes). While in the ColoredCCP problem, we require that all agents having the same color form one full segment, and that at most two of these full segments intersect. All the algorithms presented here allow only admissible collisions: i.e., at any point in time no two agents having the same color occupy the same node of the ring.

	$c = 1$, Global	$c = 2$, Global	$c > 2$, Global	Local
Asymmetric	✓(Sec. 3.1)	✓(Sec. 5)	✓(Sec. 4.1, 4.2)	× (Th. 2)
Palindrome	✓(Sec. 3.2)	✓(Sec. 5.2)	✓(Sec. 4.3)	× (Th. 2)

Table 1: Results for the CCP and ColoredCCP problems.

A summary of the results that we show in this paper is reported in Table 1. The first thing to notice is that solving CCP is impossible when the initial configuration is periodic as shown below (Theorem 1).

Theorem 1. *Given a dynamic ring $\mathcal{G}$, and a set of agents A initially placed on G_0 in a configuration that is periodic and disconnected, it is impossible to solve the* CCP *or the* ColoredCCP *problem, even if the agents have global visibility.*

Proof. In a periodic configuration, the ring can be partitioned into identical segments and none of these are full segments. In case no edge is ever missing, the symmetry between the agents in the two consecutive segments cannot be broken deterministically, thus agents in equivalent positions take the same action in each step and the resulting configuration remains periodic. Since any compacted configuration (with $k < n$) is not periodic, the theorem follows.

Therefore, in the following we assume that the initial configuration is either asymmetric or palindrome. These two cases are handled separately. However even for aperiodic configurations, the compaction problem cannot be solved in the local visibility model. In the following, the visibility graph of a configuration C is the defined as the graph $G_{vis} = (A, E)$, where A is the set of agents and there is an edge $(a, b) \in E$ whenever agent b is within distance R from a

Theorem 2. *In the local snapshot model, starting from a configuration C such that C is asymmetric and has a connected visibility graph, there is no correct algorithm that solves* CCP*, avoiding collisions. The result holds even if the agents have unbounded memory.*

Thus, in the following, we will consider the global snapshot model. We assume that the initial configuration is aperiodic (i.e. it is either asymmetric or palindrome). We will also assume that there are more than two agents in total (the special case of exactly $k = 2$ agents is handled separately in Section 6).

3 CCP with Global Snapshot

3.1 The Asymmetric Case

First, let us consider the case when the initial configuration is asymmetric. We denote by $\mathcal{E}_r$ the empty segment of maximum size in the configuration at round r. If initially there is only one empty segment of maximum size, we call this segment D. Otherwise, if there is more than one empty segment of maximum size, we can deterministically select one of these as segment D (since the initial configuration is asymmetric). Let S_1 and S_2 be the full segments of length at least 1 on the two sides of segment D (see Figure 2a). In case $|S_1| \neq |S_2|$, without loss of generality let $|S_1| < |S_2$; we define the *augmented* S_1, denoted by S_1^+, as the block of nodes constituted by the nodes in S_1 (all non empty), plus the empty node v close to S_1 and not in D, plus, if any, all agents between v and the next empty node (moving away from S_1, see Figure 3a).

The algorithm for solving CCP tries to increase the length of the empty segment D in each step, while preserving the asymmetric configuration. This is done by moving either S_1 or S_2 or both. The details are explained in Algorithm 1.

Lemma 1. *Starting from an asymmetric configuration, by executing Algorithm* One Color Connected Formation*, at any round $r \geq 0$:*

(i) $|\mathcal{E}_r| > |\mathcal{E}_{r-1}|$*, and*
(ii) The configuration is either asymmetric or solves CCP*.*

By previous lemma, since the size of $\mathcal{E}_r$ strictly increases at each round, we can state the following:

Theorem 3. *If the initial configuration is asymmetric, the agents executing Algorithm* One Color Connected Formation*, solve* CCP *within at most n rounds.*

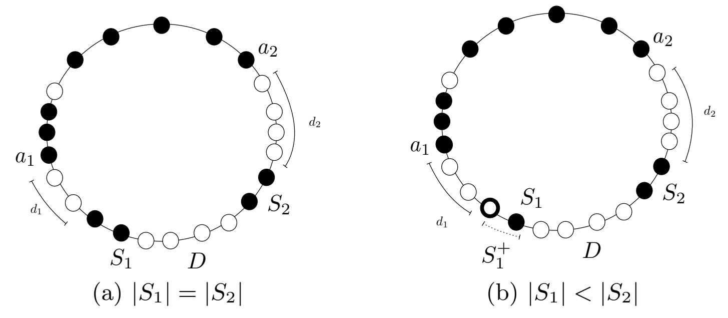


Fig. 2: Asymmetric initial configuration.

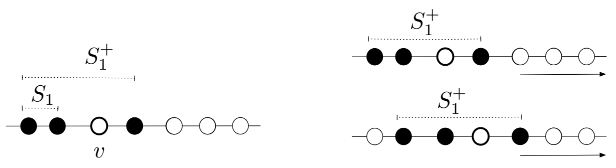
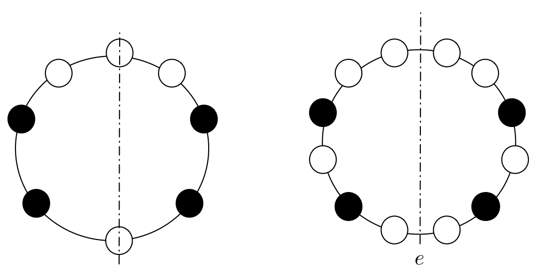


Fig. 3: (a) Definition of S_1^+ (b) Movement of S_1^+ (The arrows denotes the direction of movement)

3.2 The Palindrome Case

Let us now consider the case where the initial configuration C is palindrome i.e., there exists an axis of symmetry.



e
(a) Impossibility in the palindrome case.
(b) Possibility in the palindrome case.

Fig. 5: Example configurations for CCP in the palindrome case.

```
Algorithm 1 One Color Connected Formation
Pre-condition: Initial configuration is asymmetric.

Let $S_1$ and $S_2$ be the non-empty segments adjacent to the chosen empty segment $D$.
Let $a_1$ and $a_2$ be the agents closest to $S_1$ and $S_2$ respectively (going away from $D$).

1. If the smallest distance between $S_1$ and $S_2$ is strictly greater than one:
   (a) If $|S_1| = |S_2|$,
       – If neither $S_1$ nor $S_2$ is blocked, they both move away from $D$.
       – Otherwise, let $d_i$ be the distance between $S_i$ and $a_k$,
           • If $d_1 = d_2$, the segment that is not blocked moves away from $D$.
           • Otherwise, without loss of generality, let $d_1 < d_2$.
               ∗ If $S_1$ is not blocked, then $S_1$ moves away from $D$.
               ∗ If $S_1$ is blocked, then all agents not in $S_1$ move towards $S_1$ (pre-
                 serving the distance $d_2$).
   (b) If $|S_1| \neq |S_2|$, without loss of generality, let $|S_1| < |S_2|$ (refer to Figure 2b).
       $S_1^+$ and $S_2$ move away from $D$.
2. Else: let $v$ the only empty node separating $S_1$ and $S_2$. If the largest among the
   segments $S_1$ and $S_2$ is not blocked, this segment moves towards empty node $v$.
   Otherwise the other segment moves towards node $v$.
```

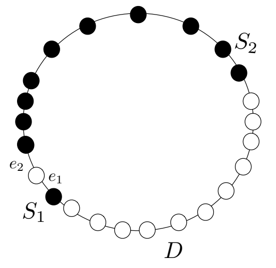


Fig. 4: Case 2 of Algorithm 1: the distance between S_1 and S_2 is 1.

Theorem 4. *Let the initial configuration be palindrome, aperiodic, and not compact. Then, if the axis of symmetry passes through two empty nodes, then* CCP *is not solvable.*

Proof. Let us assume that the problem is solvable, and that, by contradiction, the axis of symmetry of the initial configuration passes through two empty nodes (see Figure 5.(a)). If no edge is missing during the algorithm, the agents in both sides of the axis perform symmetric actions and the configuration stays palindrome with the same axis of symmetry. Since the agents avoid collision, no agent can move to the nodes on the axis; therefore, CCP cannot be solved in this case.

In Algorithm 2, we present a solution for CCP with more than 2 agents, when the initial configuration is aperiodic and palindrome, and the axis of symmetry either (a) passes through at least one edge, or (b) passes through at least one non empty node.

By Algorithm 2, and by Theorem 3, it follows that:

Theorem 5. *If the initial configuration is aperiodic and palindrome with more than two agents, and the axis of symmetry either (a) passes through at least one edge, or (b) passes through at least one non empty node, then* CCP *is solvable.*

Algorithm 2 One Color Palindrome

Pre-condition: Initial configuration is aperiodic and palindrome, with more than two agents. The axis of symmetry does not pass through two empty nodes.

(a) **If the axis of symmetry passes through at least one edge.** Since the configuration is aperiodic, we can elect a unique edge e that is crossed by the axis of symmetry ax. Once e has been elected, the two agents nearest to e that do not belong to a full segment containing e, are selected to move towards e. If none of these agents are blocked by a missing edge, the symmetry axis is preserved after the moves of the agents. Otherwise, if an agent cannot move because of a missing edge, the next configuration becomes asymmetric, and Algorithm 1 can be applied.

(b) **If the axis of symmetry passes through at least one non empty node.** In aperiodic configurations, it is always possible to elect one of the agents (agent a) among those that occupy the nodes crossed by the unique axis of symmetry.
   1. If the neighbors nodes of a are empty, a moves to one of the neighbors (chosen arbitrarily when both incident edges are available); After the move, the configuration becomes asymmetric and Algorithm 1 can be applied.
   2. If the two neighbors nodes of a are both occupied, and the axis of symmetry passes through another node occupied by agent b, and the two neighbors nodes of b are both empty, then a moves to one of the neighbors (chosen arbitrarily when both incident edges are available); After the move, the configuration becomes asymmetric.
   3. If no agent on the symmetry axis can move, since the configuration is palindrome, there must be two (full) segments of equal size to both the left and the right of a. These two segments move away from a by one position. Now, either the configuration becomes asymmetric (if one of the two segments cannot completely move because of a missing edge), or previous Case b.1 applies.

---

4 ColoredCCP with Global Snapshot and $c > 2$.

In this section, we investigate the compaction problem for heterogenous agents having $c > 2$ distinct colors. Recall that $h = k/c$ is the number of agents of each color. Obviously there is nothing to solve when $h = 1$.

4.1 Asymmetric initial configuration and $h \geq 3$

The algorithm for this case builds segments around some specific points of the ring, called *rally points.* These points are identified during the execution of the algorithm, and to each color is assigned a specific rally point.

Definition 3. *We say that agents are forming a* compact line *if they are forming a full segment of size h around the rally point of their color. We say that agents are forming an* almost compact line *if they are forming a full segment of size $h-1$ around the rally point of their color; the only agent that is not part of the almost compact line is called a* dangling *agent.*

FC denotes the set of agents colored with max_color. We say that the current configuration is *correctly placed* if and only if both the following conditions hold:

(i) There are at least $c-2$ compact lines that do not overlap;
(ii) There is at most one almost compact line.

```
Algorithm 3 MULTI COLOR CONNECTED SEGMENT (First Step)
Pre-condition: Current configuration is not correctly placed and $FC$ is symmetric.
Let $a$ be the first agent in $FC$, according to the total ordering; $a$ will move of one step
to make $FC$ asymmetric.
```

The algorithm is split into three main steps, described in Algorithms 3, 4, and 6, respectively. Let us first describe the intuition behind each step.

- **First Step** (Algorithm 3). The main idea of the first step is to make an agent with color FC move in such a way that all agents with color FC become asymmetrically placed (this step is skipped if FC is already asymmetric). Once FC is asymmetric, we keep still the agents in FC until the last phase of the algorithm: these agents are used as reference points to univocally identify both the rally points and a unique orientation of the ring.
- **Second Step** (Algorithm 4). In the second step, the algorithm proceeds by making each color but FC to form a full segment around the respective rally point. This step lasts until the configuration becomes correctly placed. Note that it is not possible to wait until all agents not in FC form compact lines (i.e., with no dangling agents): in fact, one of the agents not in FC might become blocked by a missing edge, and the whole system become blocked forever.
- **Third Step** (Algorithm 6). Once the configuration is correctly placed, the only agents still to fix in order to solve the problem, are the agents in FC (that are still asymmetrically placed), and the only dangling agent (that has a color different from FC), if any. Note that, if there is no dangling agents, then there are $c-1$ compact lines, and no almost compact line.
  The idea is to use the compact lines formed so far to establish a global chirality of the ring, and a rally point for FC. In particular, the already formed compact lines do not move, hence the computed chirality can be kept; the other agents (i.e., those in FC and the dangling agent) move as done in the second step. The movements go on until either ColoredCCP is solved, or there are $c-1$ compact lines and one almost compact line. In

this second case, the only dangling agent and the almost compact line (by construction, all these agents have the same color) move one towards each other until they form a compact line.

Correctness: Since the initial configuration is asymmetric, we have the following:

Lemma 2. *If in the initial configuration FC is not asymmetric, by executing Algorithm 3 (Step One), within finite time agents in FC are placed asymmetrically on the ring.*

Once the agents in FC occupy asymmetric positions on the ring, it is possible to elect one of them as a leader, which provides a global orientation to the ring. Once a global orientation has been computed, the positions of agents in FC allow also to compute the rally points where all other agents will form their respective compact lines (Algorithms 4). Let us denote these points by rp_i, $0 \leq i \leq c-1$. To each rally point rp_i, $0 \leq i \leq c-1$, is assigned a color, c_i (color c_0 is *max_color*, and is assigned to FC): all agents of color c_i will gather around rp_i, as described in Routine Rally Points Connected Formation, reported in Algorithm 5. Given a rally point rp_i, let us call the *rally line* of color c_i a full segment of color c_i that is formed around rp_i. Extending Definition 3, we will call dangling any agent that is not part of a rally line.

Algorithm 4 Multi Color Connected Segment (Second Step)

Precondition: Current configuration is not correctly placed, and FC is asymmetric.

During this step, FC never moves until current configuration is correctly placed. Since FC is asymmetric, it can be used to establish an orientation of the ring; also let v_f the first node in FC according to this orientation.

1. **Rally Points Computation.** FC is now used to compute $c-1$ *rally points*, as follows: v_f is the first rally point, rp_0. The $i-th$ rally point rp_i is the node of the ring at distance $i * (2 \cdot h + 1)$ from rp_0 (in the clockwise direction; we assume the ring size is at least $2 \cdot h \cdot c + c$).
2. **Formation using Rally Points.** The rally points are now used by the other colored classes to form a line, by executing routine Rally Points Connected Formation in Algorithm 5.

Lemma 3. *Within finite time, by executing Routine* Rally Points Connected Formation *in Algorithm 5, the system reaches a configuration with $c-1$ almost compact rally lines.*

Proof. If $c-1$ rally lines are almost compact, the lemma trivially follows. Thus, let us assume that there exists at least one rally line, rl_i, that has at least two

Algorithm 5 Multi Color Connected Segment (Auxiliary routine)

There are c rally points, sorted according to the ring orientation.

Case:(Pattern 1) There exists a rally line rl_i of color different from max_color that is being formed around rally point rp_i that has at least two dangling agents. Given a dangling agent a, let p be the counter-clockwise path that connects a with its own rally line.

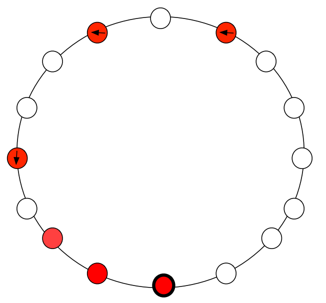
Movement (see Figure 6):

- If a is not the farthest agent from its rally line (according to the counter-clockwise direction), and on p there is a missing edge, then a does not move.
- If on p there is no missing edge, then a moves counterclockwise.
- If on p there is a missing edge, and a is the farthest agent from it rally line (according to the counter-clockwise direction), then a moves clockwise.

Case:(Pattern 2) For all rally lines of color different from max_color, there is at most one dangling agent; let m be the number of rally lines with exactly $h-1$ agents (i.e., only one dangling agent). Given a dangling agent a, let p be the counter-clockwise path that connects a with its own rally line.

Movement (see Figure 7):

- If a does not have the shortest distance to its own rally line among all distances of all other dangling agents from their own rally lines (according to clockwise direction), then a does not move.
- If the first edge on p is not missing, then a moves counter-clockwise.
- If there are $m-1$ dangling agents that are blocked by a missing edge, and a has the shortest distance to its own rally line among all distances of all other dangling agents from their own rally lines, then a moves clockwise.


(a) The dangling agents are not blocked. They move counter-clockwise towards their rally line.

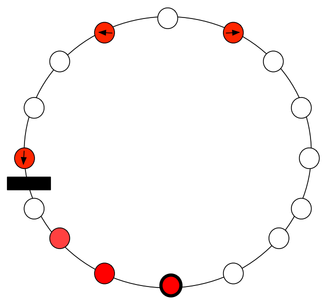
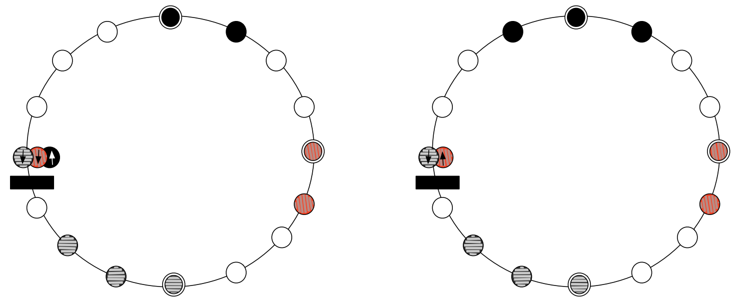

(b) The dangling agents are blocked. The last agent changes direction and move clockwise towards its rally line.

Fig. 6: Pattern 1 of Algorithm 5.


(a) The black agent switches direction.

(b) The vertical striped agent switches direction.

Fig. 7: Pattern 2 of Algorithm 5.

dangling agents. By construction, only Pattern 1 of Rally Points Connected Formation can be executed. Let us consider only agents having color c_i. Let a be the closest agent in the counter-clockwise direction to rp_i that has not reached rl_i yet. We will show that, within finite time, the size of rl_i increases. Note that, as long as a is not blocked, it will always move towards its own rally line, even if other agents are blocked.

Therefore, if a is never blocked by the missing edge, the statement trivially follows. Otherwise, let r be the furthest agent from rl_i: by Pattern 1, r switches direction, and starts moving towards rl_i. As long as a is blocked, r keeps approaching rl_i. If r becomes blocked, a does at least one step towards rl_i decreasing its distance from rl_i. Thus, within finite time, either a or r will join rl_i.

In conclusion, within finite time, rl_i becomes almost compact, and the lemma follows.

Lemma 4. *Let us assume that in the current configuration there exist $m > 2$ rally lines with exactly one dangling agent each, and $c - 1 - m$ compact lines. Within finite time, by executing Routine* Rally Points Connected Formation *in Algorithm 5, m decreases.*

Thus, by previous Lemmas 3 and 4, the following holds:

Lemma 5. *Within finite time, by executing Algorithm 4, the configuration becomes correctly placed.*

Finally, by executing Algorithm 6, agents are able to solve the problem. In particular, at the beginning of this step, there are at least $c - 2$ compact lines, at most one line with just one dangling agent, and finally the agents in FC, that still needs to be compacted. Thus, the agents that still need to be placed to correctly solve the problem, are those in FC and the dangling agent.

Algorithm 6 Multi Color Connected Segment (Third Step)

Precondition: Current configuration is correctly placed.

- Since agents in FC have to move, it is possible that the orientation of the ring that FC is establishing gets lost. Therefore, before moving any agent in FC, the other $c-1$ classes (one class per color) are used to establish a new orientation of the ring: in particular, let L_2 and L_3 be the set of agents colored with the second and third color. The agents in L_2 and L_3 are either both already compacted, or one of them (at most) forms an almost compact line. Without loss of generality, let us assume that L_2 forms a compact line. The new orientation of the ring follows the smallest distance from L_2 to L_3 (note that, by the definition of rally points, this distance is unique).
  Now, the rally point for FC, call it rp^*, is computed by taking the middle point of the largest segment between the lines that are not colored FC.
- The agents in FC and the dangling agent starts compacting, using rules described in Rally Points Connected Formation, as follows: agents in FC use rp^* as rally point, and the dangling agent uses as rally point the middle point of the almost compact line having its own color.
- If all lines are formed, and at most one has a dangling agent, the two portion of the last line to be compacted move towards each other.

Lemma 6. *If there are* 3 *or more colors, then, within finite time, by executing Algorithm 6 (Third Step), the* ColoredCCP *is solved.*

Combining all the results from this section, we have the following result:

Theorem 6. *Starting from an asymmetric initial configuration, with* $c \geq 3$ *and* $h \geq 3$*, algorithm* Multi Color Connected Segment *correctly solves the* ColoredCCP *problem.*

4.2 Asymmetric initial configuration and $h = 2$

In this section we focus on the case of agents with many colors ($c > 2$) but only two agents of each color ($h = 2$). In this case, agents execute again the three steps of previous section with a slight modification; The agents of the two maximum colors act as a single team having just one color. Thus, FC is the union of the agents having these two colors. This ensures that there are at least 3 agents in FC, such that the previous algorithm can still be executed.

At the end of the algorithm, agents of $c-2$ colors have formed compact lines and only the agents in FC form a segment where two colors are interleaved. More specifically, Configuration A in Figure 8 is, up to symmetries, the only possible interleaved configuration. At this point we run a simple separation procedure that separates the agents of distinct colors and forms the remaining two compact lines. As shown in Figure 8 from the configuration A, we can reach either configuration B or configuration C by swapping the agents on either edge

e_1 or edge e_2 (at least one of these edges must be available). Thus we reach a configuration where $c-1$ compact lines are already formed. For the two agents of the last color that is not compacted yet, these two agents can simply move towards each-other. Since there are at least 2 compact lines of other colors already formed, the configuration remains asymmetric after any movement of these two agents. Thus, eventually these two agents will reach adjacent nodes and thus, the ColoredCCP problem would be solved.

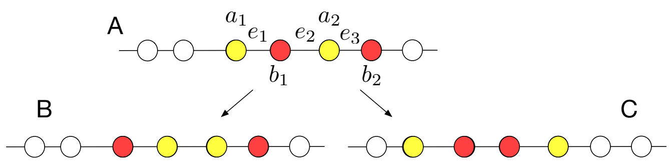


Fig. 8: Separating an interleaved line with $h = 2$ and two colors.

Theorem 7. *Starting from an asymmetric initial configuration, with $c \geq 3$ and $h = 2$, the modified algorithm* Multi Color Connected Segment *in this section correctly solves the* ColoredCCP *problem.*

4.3 Palindrome initial configuration for $c > 2$ colors

We now consider the only remaining case for ColoredCCP with $c > 2$ colors.

Theorem 8. *Starting from an initial configuration that is palindrome, aperiodic, and not compact, the* ColoredCCP *problem for $c > 2$ is not solvable if*

1. *the axis of symmetry passes through two empty nodes, or,*
2. *the axis of symmetry passes through one edge and one empty node, or,*
3. *the axis of symmetry passes through two edges and $c > 3$.*

Proof. We prove each of the statements independently.

1. The proof comes directly from Theorem 4.
2. By hypothesis, the configuration is palindrome; moreover, the symmetry axis intersects the ring on a node v and an edge e. Therefore, the agents can form the compact lines either around v or around e. If the lines are formed around v, since the ring not oriented, two agents with the same color would move to v, thus violating the no collision requirement of the problem. If the line would be formed around e, then there would be three compact lines of three different colors around e, intersecting, and thus violating the ColoredCCP specification.
3. Since the configuration is palindrome, then compact lines formable by agents have to be centred around the symmetry axis. By construction, it is only possible to form two disjoint compact lines. Since there are more than 3 colors, by the pigeonhole principle, these three compact lines will intersect, thus violating the specification of ColoredCCP.

Algorithm 7 solves the remaining cases when (a) the axis of symmetry passes through at least one occupied node, or, (b) there is an axis of symmetry passing through two edges, and $c = 3$. We can thus conclude that:

Theorem 9. *If the initial configuration is aperiodic and palindrome and either (a) the axis of symmetry passes through at least one occupied node, or (b) there is an axis of symmetry passing through two edges, and $c = 3$, then* ColoredCCP *is solvable.*

Algorithm 7 Algorithm Multi Color Palindrome

Pre-condition: Initial configuration is aperiodic and palindrome.

(a) **If the axis of symmetry passes through at least one occupied node.**
We follow the statements of Case (b) in Algorithm 2. In particular, since the configuration is not periodic, it is always possible to elect one among the agents that are on the axis of symmetry, let this agent be a. We distinguish the three possible cases:
1. If the neighbors nodes of a are empty, a moves of one position, and the configuration becomes asymmetric. Now, Algorithm of Section 4.1 can be run.
2. If the neighbors nodes of a are occupied, and the axis of symmetry passes through another node b, and the neighbors nodes of b are empty, then b moves of one position, and the configuration becomes asymmetric. Now, Algorithm of Section 4.1 can be run.
3. Finally, no node on the symmetry axis can move. In this case, since the configuration is palindrome, there must be two block of nodes of equal size to the left and to the right of a. These two block of nodes move away from a of one position. Now, either the configuration becomes asymmetric (one of the two block does not move because of a missing edge), or previous Case a.1 applies.

(b) **If the axis of symmetry passes through two edges, and $c = 3$.**
Let e be one of the edges intersected by the symmetry axis, elected as in Case (a) of Algorithm 2. The agents proceed as follows: at each round, only agents with maximum color are allowed to move. In particular, the two agents nearest to e that do not belong to a full segment containing e, move towards e. If no agent is blocked by an edge removal, the symmetry axis is preserved and eventually all agents with maximum color form a full segment around e. Otherwise, if an agent is blocked, the next configuration becomes asymmetric; thus we can apply the algorithm of Section 3.1.
Once we have a compact segment of the first color, following the same strategy, the second color in the order will form a full segment around the antipodal edge e' of e. Finally, the agents of the third color form a full segment around edge e, solving ColoredCCP.

5 Colored CCP with Global Snapshot and $c = 2$

5.1 Asymmetric Initial configuration

If $h = 2$, the agents act like they have the same a color, and form one full segment using the algorithm presented in Section 3.1. Once this full segment is formed, they separate using the technique described in Section 4.2.

If $h \geq 3$, algorithm Two Color Connected Segment (Algorithm 8) solves the problem. The algorithm is based on a modification of the strategy in Section 4.1: eventually two compact lines are formed, with possible overlap.

Algorithm 8 Two Color Connected Segment

1. **First Step** is the same as in Algorithm 3.
2. **Second Step** is the same as in Algorithm 4. Please note that, at the end of this step, agents in FC are not forming a full segment, while the agents with the other color form an almost compact line.
3. **Third Step**: at this point the dangling agent and the almost compact line will form a unique line by moving towards each other. Once this is done, the agents in FC form a compact line, by executing Algorithm 1.

By the discussion presented in previous Section 4.1, we have:

Theorem 10. *If $c = 2$, and the initial configuration is asymmetric, within finite time, Algorithm 8 solves* ColoredCCP.

5.2 Palindrome Initial configuration

First of all, by Theorem 4, we can state that:

Theorem 11. *Let the initial configuration be palindrome, aperiodic, and not compact. If the axis of symmetry passes through two empty nodes, then* Colored-CCP *is not solvable.*

Finally, following the lines of previous Algorithm 7, it is easy to show that (a) if the axis of symmetry passes through at least one occupied node, or (b) if $h \geq 2$, $c = 2$ and there is an axis of symmetry passing through at least one edge, then ColoredCCP is solvable.

6 Special Case: Compaction of $k = 2$ agents

For the CCP problem, we assumed that there are $k > 2$ agents throughout this paper and we now consider the remaining case. For the case of $k = 2$ agents of the same color, the CCP cannot be solved in dynamic rings using oblivious agents. Any configuration with two agents is a palindrome configuration. Thus, if the axis of symmetry passes through two nodes (i.e the distance between the agents is even on both sides), then the problem in not solvable due to previous results. On the other hand if the symmetry axis passes through an edge, when the agents try to approach this edge, one agent may be blocked, so the resulting configuration would have the axis of symmetry passing through a node and the agents would not be able to solve the problem. Thus, some form of persistent memory is needed to solve the problem. If the agents have some persistent memory of at least one bit, then during the execution of the algorithm when one of the agents is blocked, then this agent can be elected, by setting a flag in the memory of this agent; during the rest of the algorithm, the agent can simply approach each-other until they are compacted or there is exactly one empty node between them. If there is only one empty node between the agents and none of the agents are blocked then the leader agent can move to the empty node to solve the problem. On the other hand, if none of the agents are blocked during the execution of the algorithm, then both agents can move in synchronous steps (maintaining the same symmetry axis) and eventually reaching the two end-points of the edge through which the axis passes. Thus we can state the following result:

Theorem 12. *For exactly two agents,* ColoredCCP *is solvable if and only if (i) the initial configuration has the axis of symmetry passing through at least one edge and (ii) the agents have persistent memory.*

7 Conclusions

In this paper we introduced and studied the Compact Configuration Problem and the Colored Compact Configuration Problem for a set of autonomous mobile agents on a dynamic ring networks. We showed that both the problems can be solved only if the initial configuration is aperiodic. The results of this paper provides the exact characterization of the solvable initial configurations for the CCP and ColoredCCP problems. We also showed that having persistent memory is not necessary for solving the problem (except in the special case of two agents). It would be interesting to determine what additional capabilities of the agents would allow them to the solve the ColoredCCP problem without any overlaps. Future investigations on this problem could also consider other graph topologies under either the same or a more relaxed model for dynamicity. Another interesting issue is to consider less synchronous models where all agents may not start at the same time and they may not be active at the same time.

References

1. M. Biely, P. Robinson, U. Schmid, M. Schwarz, and K. Winkler. Gracefully degrading consensus and k-set agreement in directed dynamic networks. 2nd International Conference on Networked Systems *(NETSYS)*, pages 109–124, 2015.
2. M. Bournat, S. Dubois, and F. Petit. Gracefully degrading gathering in dynamic rings. In *Proc. 20th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS) 2018*, pages 349–364, 2018.

3. A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Measuring temporal lags in delay-tolerant networks. IEEE Transactions on Computers, 63(2):397–410, 2014.
4. A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. Int. Journal of Parallel, Emergent and Distributed Systems, 27(5):387–408, 2012.
5. S. Das, G. A. Di Luna, and L. A. Gasieniec. Patrolling on dynamic ring networks. *CoRR*, abs/1808.04349, 2018.
6. G. Di Luna and R. Baldoni. Brief announcement: Investigating the cost of anonymity on dynamic networks. *In* Proc of the the 34th Symposium on Principles of Distributed Computing (PODC), pages 339–341, 2015.
7. G. Di Luna, S. Dobrev, P. Flocchini, and N. Santoro. Live exploration of dynamic rings. 36th IEEE International Conference on Distributed Computing Systems *(ICDCS)*, pages 570–579, 2016.
8. G. Di Luna, P. Flocchini, L. Pagli, G. Prencipe, N. Santoro, and G. Viglietta. Gathering in dynamic rings. *Theoretical Computer Science*, (in press), 2018.
9. P. Flocchini, A. M. Enriques, L. Pagli, G. Prencipe, and N. Santoro. Point-of-failure shortest-path rerouting: Computing the optimal swap edges distributively. *IEICE Transactions on Information and Systems*, E89-D(6):700–708, 2006.
10. P. Flocchini, L. Pagli, G. Prencipe, N. Santoro, and P. Widmayer. Computing all the best swap edges distributively. *Journal of Parallel and Distributed Computing*, 68(7):976–983, 2008.
11. B. Haeupler and F. Kuhn. Lower bounds on information dissemination in dynamic networks. 26th International Symposium on Distributed Computing *(DISC)*, pages 166–180, 2012.
12. D. Ilcinkas, R. Klasing, and A.M. Wade. Exploration of constantly connected dynamic graphs based on cactuses. Proc. 21st Int. Coll. Structural Inform. and Comm. Complexity (SIROCCO), pages 250–262, 2014.
13. D. Ilcinkas and A.M. Wade. Exploration of the t-interval-connected dynamic graphs: the case of the ring. *In* Proc. 20th Int. Coll. on Structural Inform. and Comm. Complexity (SIROCCO), pages 13–23, 2013.
14. A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans. Automat. Contr.*, 48(6):988–1001, 2003.
15. F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. 42th Symposium on Theory of Computing *(STOC)*, pages 513–522, 2010.
16. F. Kuhn, Y. Moses, and R. Oshman. Coordinated consensus in dynamic networks. 30th Symposium on Principles of Distributed Computing *(PODC)*, pages 1–10, 2011.
17. F. Kuhn and R. Oshman. Dynamic networks: Models and algorithms. SIGACT News, 42(1):82–96, 2011.
18. Z. Liu, Y. Yamauchi, S. Kijima, and M. Yamashita. Team assembling problem for asynchronous heterogeneous mobile robots. *Theor. Comput. Sci.*, 721:27–41, 2018.
19. L. Pagli, G. Prencipe, and G. Viglietta. Getting close without touching: Near-gathering for autonomous mobile robots. Distributed Computing, 28(5):333–349, 2015.
20. W. Ren and R.W. Beard. Consensus seeking in multi-agent systems under dynamically changing interaction topologies. *IEEE. Trans. Automatic Control*, 50(5):655–661, 2005.