# Time Optimal Distance-$k$-Dispersion on Dynamic Ring

Brati Mondal Ms.
Department of Mathematics
Jadavpur University
Kolkata, India
bratim.math.rs@jadavpuruniversity.in

Pritam Goswami Mr.
Department of computer Science and Engineering
Sister Nivedita University
Kolkata, India
pgoswami.cs@gmail.com

Buddhadeb Sau Dr.
Department of Mathematics
Jadavpur University
Kolkata, India
buddhadeb.sau@jadavpuruniversity.in

## Abstract

In the problem of Dispersion by mobile robots, $l$ robots placed arbitrarily on nodes of a network having $n$ nodes are asked to relocate themselves autonomously so that each node contains at most $\lceil \frac{l}{n} \rceil$ robots. When $l \leq n$, each node of the network contains at most one robot. In this work, dispersion has been considered by introducing another variant called *Distance-$k$-Dispersion (D-$k$-D)*, where the robots have to disperse on a network in such a way that the shortest distance between any two pairs of robots is at least $k$ and there exist at least one pair of robots for which the shortest distance is exactly $k$ for a dynamic ring (1-interval connected ring) for rooted initial configuration. Note that, when $k = 1$ we have normal dispersion, and when $k = 2$ we have *Distance-2-Dispersion (D-2-D)*. We have proved the necessity of fully synchronous scheduler to solve this problem and provided an algorithm that solves D-$k$-D in $\Theta(n)$ rounds under a fully synchronous scheduler. So, the presented algorithm is time-optimal too. To the best of our knowledge, this is the first work that considers this specific variant.

## CCS Concepts

• **Theory of computation → Distributed algorithms**.

## Keywords

Dispersion, Dynamic ring, mobile robots, distributed algorithm.

**ACM Reference Format:**
Brati Mondal Ms., Pritam Goswami Mr., and Buddhadeb Sau Dr.. 2025. Time Optimal Distance-$k$-Dispersion on Dynamic Ring. In *26th International Conference on Distributed Computing and Networking (ICDCN 2025), January 04–07, 2025, Hyderabad, India.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3700838.3700843

## 1 Introduction

The problem of dispersion by a swarm of mobile robots is extensively studied in the field of distributed system. In *Dispersion* problem (introduced in [2]), the aim is to reposition $l$ robots on a graph with $n$ nodes in such a way that each node contains at most $\lceil \frac{l}{n} \rceil$ robots. So when $l \leq n$, each node must contain at most 1 robot. In this work, we have studied a special type of dispersion problem, called *Distance-$k$-Dispersion* aka D-$k$-D problem on a specific graph topology which is a 1-interval connected ring having $n$ nodes, assuming the initial configuration is a rooted configuration with $l \leq \lfloor \frac{n}{k} \rfloor$ mobile robots. In D-$k$-D, the robots disperse themselves autonomously in such a way so that the distance between any two occupied nodes is at least $k$ and there is at least one pair of occupied nodes whose distance is exactly $k$, where occupied nodes contain exactly one robot. For $k = 1$, the problem is equivalent to the dispersion problem. Also, Distance-2-Dispersion (introduced in [4]) problem is a special case of D-$k$-D where $k = 2$. So to the best of our knowledge, we consider Distance-$k$-Dispersion for the first time here on dynamic ring. An intriguing aspect of D-$k$-D is its application in maximizing the sensing area with the use of the fewest agents possible assuming the sensing distance of an agent is up to $k$ hop.

Here we have assumed 1-interval connectivity dynamism where the adversary can omit at most one edge in a certain round from the ring without hampering the connectivity. Each robot has a unique identifier and has weak global multiplicity detection and strong local multiplicity detection. We have shown that a solution of D-$k$-D is not possible on a 1-interval connected dynamic ring for a semi-synchronous (SSync) scheduler if the initial configuration has a multiplicity node. Also, a time lower bound ($\Omega(n)$ rounds) is given for the D-$k$-D algorithm. Then we have provided a deterministic and distributed algorithm D-$k$-D DynamicRing, which solves the D-$k$-D problem for rooted initial configuration by the robots under a fully synchronous (FSync) scheduler with chirality (i.e. the particular notion of orientation: clockwise or counter-clockwise). The algorithm D-$k$-D DynamicRing will terminate in $O(n)$ rounds and the algorithm is time optimal as its runtime matches with the lower bound provided. Observe that, if the robots have no chirality, then there are some known techniques in [1], applying that chirality can be achieved so that the robots can agree on a particular direction (clockwise or counterclockwise). Then one can easily apply the existing algorithm D-$k$-D DynamicRing to achieve D-$k$-D even considering the set of robots without chirality.

## 2 Model and Problem Definition

**Model:** Let $\mathcal{R}$ be a ring with $n$ consecutive nodes $v_0, v_1, v_2, \ldots, v_{n-1}$ in a certain direction (either clockwise or counterclockwise direction), where the nodes $v_i$ is connected to both $v_{i-1 \pmod n}$ and $v_{i+1 \pmod n}$ by edges. These nodes are unlabeled. A dynamic ring with $n$ nodes is considered here. Nodes have no memory. The 1-interval connectivity dynamism (from [3]), where the adversary can omit at most one edge from the ring in a certain round without losing the connectivity of the ring, is assumed here. let $\{r_1, r_2, \ldots, r_l\}$ be a set of $l$ robots reside on the nodes of ring $\mathcal{R}$. The robots are identical i.e. physically indistinguishable and homogeneous i.e. the


*ICDCN 2025, January 04–07, 2025, Hyderabad, India*

ACM ISBN 979-8-4007-1062-9/25/01
https://doi.org/10.1145/3700838.3700843

robots execute the same algorithm. Robots can reside only on the node but not on the path connecting two nodes. Each robot has a unique label (ID) which is a string of constant length. Robots have $\log n$ bits of memory for storing their ID, where $n$ is the number of nodes of the graph. But this memory is not updatable. So robots can't store anything else persistently. One or more robots can occupy the same node. If two or more robots are on the same node they are called co-located.

The robots can communicate with each other only when they are co-located. In each round, co-located robots in a certain node can detect the minimum ID robot of that node. Robots have the ability of **strong local multiplicity detection** which means the robot can detect how many robots are present in its node. Robots have **weak global multiplicity detection** i.e. robots can detect if there exists no robot, one robot, or more than one robot in a particular node within its visibility range. The visibility of a robot is defined as the number of hops the robot can see another node which is denoted as $\phi$. Here $\phi \geq n/2$ i.e. robots can see all the nodes of the dynamic ring. Robots can detect whether the edge between two nodes is missing or not and they can move one node to another node if the edge between them is not missing. The particular notion of orientation (clockwise or counterclockwise) of the robot in the ring is called chirality. Here clockwise (CW) and counterclockwise (CCW) notion of direction is used in the usual sense. The activation of the robot controlled by an entity is called scheduler. Here robots work under fully synchronous (Fsync) scheduler where all robots get activated at the same time and perform look-compute-move synchronously. In **look** phase, robots take a snap of their views to collect information and communicate with the robots of their own node, then perform computations on the basis of their look phase and determine whether to move or not and in which direction in the **compute** phase. The robots that are decided to move in the compute phase, move to their determined direction in **move** phase. Robots can move in their determined direction only if the edge in that direction is not missing.

Initially, all the robots are on the same node, which is called **root node**. The ring is always connected. There are two types of nodes: **occupied node** and **null node**. If a node contains a robot then it is called **occupied node**. If there is no robot on the node then the node is called **null node**. According to the numbers of robots present on the node **occupied node** is of two types: **multiplicity node**, **singleton node**. **Multiplicity node** contains multiple number of robots and **singleton node** contains only one robot. The robot on a singleton node can be called **singleton robot**.

**Problem Definition:** Let $l$ robots reside on an $n$ node dynamic ring. Initially, the robots are co-located at a particular node. Without loss of generality let us assume that node is $v_0$. In the dispersion problem, the robots reposition themselves autonomously in such a way that at each node there is at most one robot. In *distance-$k$-dispersion* (D-$k$-D) problem, the robots reposition to achieve a configuration that satisfies the following:
(i) The occupied node contains exactly one robot.
(ii) The distance between any two occupied nodes is at least $k$.
(iii) There exists at least one pair of nodes whose distance is exactly $k$.

## 3 Impossibility Result and Lower Bound

Theorem 3.1. *Distance-$k$-Dispersion on the dynamic ring is not possible for a semi-synchronous scheduler if the initial configuration has a multiplicity node on a 1-interval connected ring.*

Theorem 3.2. *Any dispersion algorithm on a ring with n nodes from a rooted configuration takes $\Omega(n)$ rounds.*

Corollary 3.3. *Any dispersion algorithm on a 1-interval connected ring with n nodes takes $\Omega(n)$ rounds to terminate.*

Corollary 3.4. *Any D-$k$-D algorithm on a 1-interval connected ring with n nodes takes $\Omega(n)$ rounds to terminate.*

## 4 Definitions and Preliminaries

In this section let us describe some useful definitions.

*Definition 4.1 (Configuration).* A *configuration* at a round $t$ can be denoted as $C(t) : (V, t, f)$ where $f$ is a function from $V$ to $\{0, 1, 2\}$ such that

$$f(v) = \begin{cases} 0, & \text{when v is null node at round t} \\ 1, & \text{when v is single node at round t} \\ 2, & \text{when v is multiplicity node at round t} \end{cases}$$

An induced sub-graph of the ring $\mathcal{R}$ containing the vertices starting from a node $v_x$ in the direction $\mathcal{D}$ ending at the node $v_y$ is called an *arc* and it is denoted as $(v_x, v_y)_{\mathcal{D}}$. For any two nodes $v_x$ and $v_y$ on the ring $\mathcal{R}$, the *distance between the nodes* $v_x$ and $v_y$ in a direction $\mathcal{D}$ is the number of edges from the node $v_x$ to $v_y$ in direction $\mathcal{D}$ and it is denoted as $d_{\mathcal{D}}(v_x, v_y)$. Two occupied nodes $v_x$ and $v_y$ on the ring $\mathcal{R}$, are said to be *consecutive occupied nodes* (Fig 1) if there exists a direction $\mathcal{D}$ (either clockwise or counterclockwise direction) such that $(v_x, v_y)_{\mathcal{D}}$ does not contain any other occupied node other than $v_x$ and $v_y$.

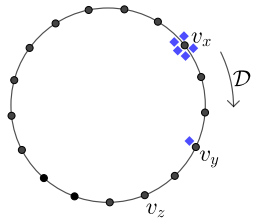



**Figure 1: $v_x$ and $v_y$ are consecutive occupied nodes in the arc $(v_x, v_z)_{\mathcal{D}}$ of length 5 in direction $\mathcal{D}$. Black dots represent the nodes on the ring and blue boxes represent the robots.**

*Definition 4.2 (Clockwise Chain).* Let $\mathcal{R}$ be a ring with vertices $v_0, v_1, \ldots, v_{n-1}$ in clockwise direction $\mathcal{D}$ (say). The arc starting from the node $v_{i-1 \pmod n}$ to $v_j$ i.e. $(v_{i-1 \pmod n}, v_j)_{\mathcal{D}}$, for some $i, j$ is called *clockwise chain (CW-chain)* if

(i) $v_i$ is multiplicity node, $v_{j-1}$ is occupied node and $v_j$ is null node.

(ii) $d_{\mathcal{D}}(v_x, v_y) = k$, for any two consecutive occupied nodes $v_x$, $v_y \in (v_{i-1 \pmod n}, v_j)_{\mathcal{D}}$.

(iii) $d_{\mathcal{D}}(v_{j-1}, v_p) > k$, for the nearest occupied node $v_p$ in clockwise direction from the node $v_{j-1}$.

similarly one can define *counterclockwise chain (CCW-chain)* considering counterclockwise direction.

*Definition 4.3 (Chain Configuration and its range).* A configuration that contains both clockwise and counterclockwise chains, is called *chain configuration* (Fig 2). In a chain configuration, the maximum size arc of the ring $\mathcal{R}$ such that every node on that arc is contained in at least one chain is called the *range of the chain configuration.*

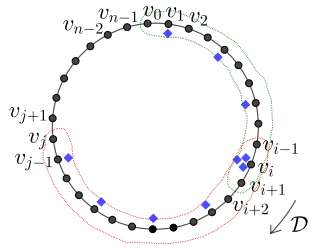
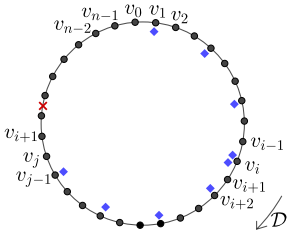
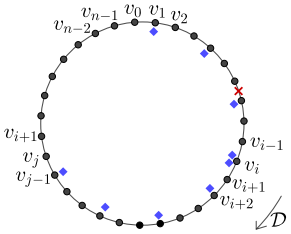



**Figure 2: Chain configuration where $(v_{i-1}, v_j)_{\mathcal{D}}$ is a chain in clockwise direction $\mathcal{D}$ and $(v_{i+1}, v_0)_{\mathcal{D}'}$ is a chain in counter clockwise direction for $k = 3$.**

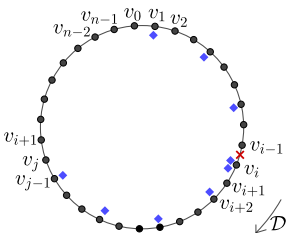
If the configuration is not chain configuration then it is called *non-chain configuration.* In a chain configuration, a chain is called a *good chain* if the chain contains no missing edge. Otherwise, it is called a *bad chain.* There are three types of chain configuration: chain configuration containing two good chains, containing one good chain, and one bad chain, containing two bad chains. A configuration is called *dispersed configuration* if there is no multiplicity node.




(a) Both Chain good chain




(b) one good and one bad chain




(c) Both chain bad chain

**Figure 3: Different types of chain configuration for $k = 3$**

*Definition 4.4 (Head).* In a dispersed configuration, if there exists exactly one pair of occupied nodes $v_x$ and $v_y$ on the ring $\mathcal{R}$ such that $d_{\mathcal{D}}(v_x, v_y) < k$ for clockwise direction $\mathcal{D}$, then $v_x$ is called the *head* and denoted as $H$.

*Definition 4.5 (Block).* Let $\mathcal{R}$ be a ring with vertices $v_0, v_1, \ldots, v_{n-1}$ in a direction $\mathcal{D}$. The arc starting from the node $v_i$ to $v_j$ i.e. $(v_i, v_j)_{\mathcal{D}}$, for some $i, j$ with maximum arc length is called a *block* in direction $\mathcal{D}$ if

(i) $v_i$ is a multiplicity node or head.

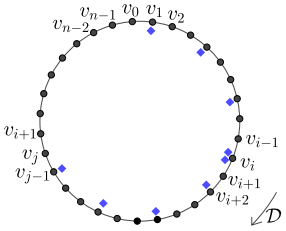
(ii) $v_j$ is a null node and $v_{j-1}$ is occupied node, where $v_{j-1}$ is the node adjacent to $v_j$ in the arc $(v_i, v_j)_{\mathcal{D}}$.

(iii)(a) $d_{\mathcal{D}}(v_x, v_y) = k$, for all consecutive occupied nodes $v_x$, $v_y \in (v_i, v_j)_{\mathcal{D}}$

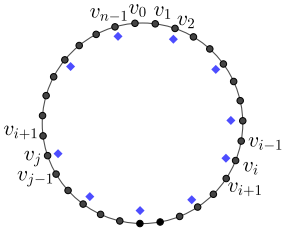
or, (b) $d_{\mathcal{D}}(v_i, v_z) < k$, where $v_i$ and $v_z$ are consecutive occupied nodes $\in (v_i, v_j)_{\mathcal{D}}$ and $d_{\mathcal{D}}(v_x, v_y) = k$, for all consecutive occupied nodes $v_x, v_y \in (v_z, v_j)_{\mathcal{D}}$

(iv) $d_{\mathcal{D}}(v_{j-1}, v_p) > k$, for the nearest occupied node $v_p$ in direction $\mathcal{D}$ from the node $v_{j-1}$.

The node $v_i$ is called *block head.* For a block conditions (iii)(a) and (iii)(b) can not satisfy simultaneously. In a block if condition (iii)(a) is satisfied it is called a *chain block* otherwise if condition (iii)(b) is satisfied it is called a *non-chain block.*




(a) Block configuration.




(b) Dispersed configuration.

**Figure 4: Different types of non-chain configuration for $k = 3$**

*Definition 4.6 (Target Configuration).* A dispersed configuration is said to be target configuration ($C_T$) if, $d_{\mathcal{D}}(v_x, v_y) \geq k$, for all consecutive occupied nodes $v_x, v_y$ on the ring $\mathcal{R}$, where $\mathcal{D}$ be either CW direction and CCW direction.

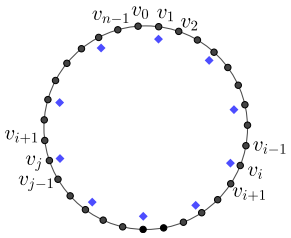
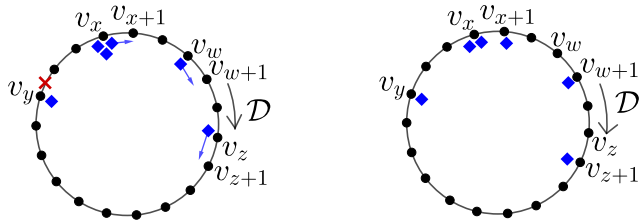
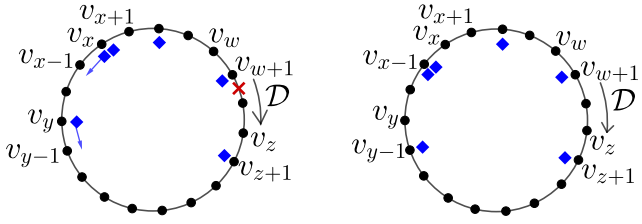



**Figure 5: Target Configuration of D-$k$-D where $k = 3$**




(a) Chain configuration. (b) Non-chain configuration.

**Figure 6: Chain to non-chain configuration**




(a) Non-chain configuration. (b) Chain configuration.

**Figure 7: Non-chain to chain configuration**

## 5 Discussion of algorithm and correctness

In this section, the main idea of D-$k$-D has been discussed. All the robots are initially placed on a particular node i.e., the configuration is rooted initially. The robots have chirality i.e., they agree on a particular direction (without loss of generality, let the direction be clockwise). The main task is divided into two subtasks. If the configuration is a chain configuration (Fig. 2), then a robot, say $r$ on chain executes the algorithm Spread($r$) (Algorithm 2). If the configuration is not a chain configuration (Fig. 4(a)) then the robots reconstruct the chain configuration by executing ReconstructChain($r$) (Algorithm 3). The main aim is to form the target configuration ($C_T$) (Fig. 2) where the distance between two consecutive occupied nodes is at least $k$. Note that initially, the rooted configuration is a chain configuration. The idea is to spread the range of the chain configuration by relocating one more agent from the multiplicity node to a null node (thus number of robots on multiplicity decreases). Now since the ring is 1-interval connected, to achieve a spread maintaining the chain configuration might not be possible. This ends up resulting in a non-chain configuration (Fig 6). In this scenario, the aim becomes reconstructing a chain configuration from the non-chain configuration while maintaining the number of robots on the multiplicity (Fig 7). So a combination of Spread($r$) followed by finite consecutive execution (at most $k-1$) of ReconstructChain($r$), the configuration spreads the range of the chains while reducing the number of robots on the multiplicity point by one. We call an execution of Spread($r$) followed by a finite consecutive (at most $k-1$) execution of ReconstructChain($r$) until a chain configuration is reached, an *iteration*. Note that an iteration can be at most of $k$ rounds. So for $l$ robots within $l$ iteration, the configuration will be dispersed. Then from the disperse configuration within at most $k-1$ execution of ReconstructChain($r$), the configuration becomes $C_T$ (Fig. 5).

**Algorithm 1:** D-$k$-D DynamicRing($r$)

```
if chain configuration then
    Execute Spread(r);
else
    Execute ReconstructChain(r);
```

**Subroutine** Spread($r$)**:** During the execution of the algorithm at any round, the configuration can be chain configuration. In chain configuration two chains exist, one is CW-chain another is CCW-chain. Then three cases can arise: both are good chains, one chain is good another is bad, and both are bad chains (Fig 3). Initially, all the robots are at the same node which is called the root i.e. the initial configuration is rooted configuration. Note that this is a chain configuration by Definition 4.3. If the configuration contains two good chains, then either there is no missing edge or the missing edge is not in any chain. In this case, the robots on singleton nodes of the CCW chain move in the counterclockwise direction. The robots on the multiplicity can find the least ID robot by exploiting local communication. The robot with the least ID on the multiplicity moves in the counter clockwise direction. If the configuration contains one good chain and one bad chain, then the missing edge is on the bad chain but not adjacent to the multiplicity node. Let $\mathcal{D}$ be the direction (either CW or CCW) from multiplicity in which the good chain exists. In this case, all singleton robots on the good chain and the least ID robot on multiplicity move in direction $\mathcal{D}$. If the configuration contains two bad chains, then the missing edge is adjacent to multiplicity which is contained in both chains. This missing edge must be next to the multiplicity node either in the clockwise direction or in the counterclockwise direction. Let $\mathcal{D}$ be the direction in which multiplicity has no missing edge. In this case, the least ID robot on multiplicity moves in direction $\mathcal{D}$. The

**Algorithm 2:** Spread($r$)

```
if r is on multiplicity node then
    if r has the least ID among all nodes on the same node then
        if both chains are good chains then
            D ← counter-clockwise direction;
        else if exactly one chain is a bad chain then
            D ← direction of the good chain;
        else if both chains are bad chains then
            D ← direction in which multiplicity node has no adjacent
              missing edge;
        r moves in direction D;
else
    if both chains are good chains then
        if r is on counter-clockwise chain then
            r moves in counter-clockwise direction;
    else if exactly one chain is a bad chain then
        D ← direction of the good chain;
        if r is on good chain then
            r moves in direction D;
    else if both chains are bad chains then
        D ← direction in which multiplicity node has no adjacent missing
          edge;
        if r is on the chain in direction D then
            r moves in direction D;
```

singleton robots of the chain in $\mathcal{D}$ direction move in the direction $\mathcal{D}$.

As a consequence of the subroutine Spread($r$), one can observe that, if the chain configuration contains one good chain and one bad chain or two bad chains then, after executing the subroutine Spread($r$) the chain configuration may get hampered and the configuration will become a non-chain configuration.

**Subroutine** ReconstructChain($r$) **:** If the configuration is not chain configuration (Fig 4) then the configuration must contain one chain block and one non-chain block . Let $\mathcal{D}$ be the direction in which the chain block occurs. If there is no missing edge in both blocks (chain block or non-chain block) or if the missing edge is in the non-chain block, then all singleton robots on the chain block move away from the block head in the direction $\mathcal{D}$ and all robots on block head moves towards the direction $\mathcal{D}$. If the missing edge is in the chain block, then the singleton robots in the non-chain block move away from the block head in direction $\mathcal{D}'$, opposite direction of $\mathcal{D}$ Now we discuss the correctness of the algorithm

**Algorithm 3:** ReconstructChain($r$)

```
if r is on block head then
    if no edges are missing ∨ missing edge is in non-chain block then
        r moves in the direction of the chain block
else
    if no edges are missing ∨ missing edge is in non-chain block then
        if r is on chain block then
            r moves away from block head;
    else if missing edge is in chain block then
        if r is on non-chain block then
            r moves away from the block head;
```

D-$k$-D DynamicRing. One can prove that, at any round $t$, if a robot $r$ executes the subroutine Spread($r$) then at the end of the round $t$ the number of singleton nodes will increase. Moreover, While executing Algorithm 1, if $C(t)$ be a non-chain configuration but not target configuration at round $t$, then the configuration $C(t)$ must have two blocks (chain block and non-chain block).

If $C(t)$ is a chain configuration while $C(t+1)$ is a non-chain configuration during the execution of Algorithm 1. Then the following statements are true.

(1) In $C(t+1)$, there exists a direction $\mathcal{D}$ such that $d_{\mathcal{D}}(H, H+1) = 1$, where $H$ and $H+1$ are the block head and the adjacent occupied node of block head in direction $\mathcal{D}$
(2) $d_{\mathcal{D}}(v_x, v_y) = k$, for all consecutive occupied nodes on the block in direction $\mathcal{D}$ where $v_x \neq H$.
(3) $d_{\mathcal{D}'}(v_x, v_y) = k$ for all consecutive occupied nodes $v_x$ and $v_y$ on the block in the direction $\mathcal{D}'$, where $\mathcal{D}'$ be the opposite direction of $\mathcal{D}$.

Again if $C(t)$ is a non-chain configuration for some $t > 0$, which is not the target configuration. Then the following are true.

(1) Let $d_{\mathcal{D}}(H, H+1) = d < k$ where $H$ is block head and $H+1$ is the adjacent occupied node of $H$ in direction $\mathcal{D}$, where $\mathcal{D}$ be the direction of the non-chain block in $C(t)$. Then $d_{\mathcal{D}}(H', H'+1) = d+1$ in $C(t+1)$. Here $H'$ is either block head or multiplicity in $C(t+1)$ and $H'+1$ is the adjacent occupied node of $H'$ in direction $\mathcal{D}$ in $C(t+1)$.
(2) Let $C(t')$ be the first chain configuration (if exists) or the target configuration after $C(t)$, where $t' > t$. Then $t' - t \leq k-1$.

Theorem 5.1. *The algorithm D-k-D DynamicRing will terminate in $O(n)$ rounds.*

From Theorem 5.1 and Corollary 3.4 we have the following result:

Theorem 5.2. *Algorithm D-k-D DynamicRing terminates in $\Theta(n)$ rounds for any 1-interval connected ring with n nodes.*

If robots have no chirality agreement, then by algorithm *No-Chir-Preprocess* proposed in [1] by Agarwalla et al. can be used by $l$ co-located robots to agree on a particular direction before starting the execution of D-$k$-D DynamicRing. This way we can solve the D-$k$-D on a dynamic ring even when the robots do not agree on chirality.

## 6 Conclusion

The primary aim of this paper is to introduce the D-$k$-D problem, a generalization of all previously studied variants of the dispersion problem for mobile robots. The paper begins by discussing the necessity of a fully synchronous scheduler to solve this problem, providing a lower bound on the time required, which is $\Omega(n)$ for a dynamic ring with $n$ nodes. Then the paper presents an algorithm, D-$k$-D DynamicRing, which solves the problem in optimal time for a rooted initial configuration, assuming chirality within the context of a 1-interval connected ring under a fully synchronous scheduler. Additionally, for robots without chirality agreement, the problem can still be solved using a known technique ([1]). Several open problems related to this variant of dispersion for further research are to consider in arbitrary networks. Interestingly, when extending the technique used to solve D-2-D in [4] by Kaur et al., the time and memory complexity becomes exponential even when $k = 3$, making it an intriguing subject for study in arbitrary networks. Another research direction involves examining this variant under a limited visibility model, where a robot can only see up to a distance $\phi < \frac{n}{2}$ from its position on a dynamic ring.

**Acknowledgement:** The first author is supported by the University Grants Commissions (UGC), Government of India. The third author is supported by the Science and Engineering Research Board (SERB), Government of India.

## References

[1] Ankush Agarwalla, John Augustine, William K. Moses Jr., Sankar Madhav K., and Arvind Krishna Sridhar. 2018. Deterministic Dispersion of Mobile Robots in Dynamic Rings. In *Proceedings of the 19th International Conference on Distributed Computing and Networking, ICDCN 2018, Varanasi, India, January 4-7, 2018*, Paolo Bellavista and Vijay K. Garg (Eds.). ACM, 19:1–19:4. https://doi.org/10.1145/3154273.3154294

[2] John Augustine and William K. Moses Jr. 2018. Dispersion of Mobile Robots: A Study of Memory-Time Trade-offs. In *Proceedings of the 19th International Conference on Distributed Computing and Networking, ICDCN 2018, Varanasi, India, January 4-7, 2018*, Paolo Bellavista and Vijay K. Garg (Eds.). ACM, 1:1–1:10. https://doi.org/10.1145/3154273.3154293

[3] G.A. Di Luna, S. Dobrev, P. Flocchini, and N. Santoro. 2016. Live Exploration of Dynamic Rings. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. 570–579. https://doi.org/10.1109/ICDCS.2016.59

[4] Tanvir Kaur and Kaushik Mondal. 2023. Distance-2-Dispersion: Dispersion with Further Constraints. In *Networked Systems - 11th International Conference, NETYS 2023, Benguerir, Morocco, May 22-24, 2023, Proceedings (Lecture Notes in Computer Science, Vol. 14067)*, David Mohaisen and Thomas Wies (Eds.). Springer, 157–173. https://doi.org/10.1007/978-3-031-37765-5_12