Cellular Automata based Decentralized Cooperative Collision Avoidance Control for Multiple Mobile Robots

Erick J. Rodríguez-Seda and Catalina K. Rico

***Abstract*— One of the main challenges in the decentralized implementation of mobile multi-robotic systems is the avoidance of collisions among agents and other obstacles in cluttered environments as well as the convergence of each robot to its desired destination. This paper presents a novel decentralized, cooperative navigation algorithm for a team of mobile robots based on the concept of cellular automata that is proven to guarantee collision avoidance at all times even under the presence of static non-cooperative obstacles. The algorithm does not need to differentiate among agents and obstacles and use a time-varying localization-based priority rule to guarantee safe motion while reducing the occurrence of deadlocks. A Monte Carlo simulation is performed showcasing the performance of the algorithm under different densities of obstacles and agents.**

I. Introduction

In recent years, both scientific and governmental agencies have significantly invested in the development of multi-robotic systems, such as Mobile Wireless Sensor Networks (MWSNs), for area coverage applications including force protection, reconnaissance, multi-target pursuit, disaster relief, border security, and environmental sampling, among others [1], [2]. A team of relatively simple and low-cost cooperative mobile robots can replace a single, complex agent and cover large-scaled areas in shorter time, provide robustness and resiliency to the overall mission, and reduce maintenance and operational costs [3]. However, the success and efficiency of the MWSN requires a high level of coordination among agents, particularly, in navigation tasks.

The implementation and guidance of numerous autonomous agents are typically achieved through a centralized controller. The focus of contemporary research is on removing the dependency on a single decision-making unit and decentralizing the control process [4], [5]. A decentralized system implies that the decision process for an individual agent is not dependent on information of all other agents. For example, in a decentralized MWSN, each agent's information (e.g., position, state, and velocity) is typically unknown to every other agent until they become sufficiently close. Such decentralized approach offers several advantages including robustness to single-point failures, modularity, and scalability [6]. It, however, increases the chances of deadlocks. A deadlock occurs when one or multiple agents cannot reach their desired destination [7]. Ideally, each agent in the multi-robotic system should be able to reach its desired destination while avoiding collisions with other agents and obstacles.

Recognizing the advantages of decentralized control for multi-robotic applications, this paper presents a decentralized cooperative navigation algorithms based on the concept of cellular automata [8]. The algorithm assumes agents to move one cell at a time in any direction according to a set of predefined transition rules. Only information about nearby agents within two cells of distance is required and the agent does not need to differentiate between static obstacles and other agents. Moreover, the algorithm is reactive, i.e., it updates the right course of action when other agents are detected. A Monte Carlo simulation was carried out in order to evaluate and compare the performance of the algorithm under different scenarios. It is shown that the cellular automata based cooperative algorithm allows agents to successfully avoid collisions while reducing the occurrence of deadlocks.

II. Related Work

Collision avoidance for autonomous agents has been a topic of research for several decades (see [9], [10] for surveys). Example of early work includes the occupancy grid [11] and the certainty grid [12] in the late 1980s. In general, collision and obstacle avoidance laws are classified in terms of the agent's dynamics and kinematics [13]–[15], the presence of static or dynamic obstacles, level of decentralization, computational complexity, cooperation among agents, planning methods, robustness to modeling and sensing uncertainties [16], required knowledge, input constraints [17], and scalability, among other features. Some of the most popular approaches include the use of velocity obstacle [18], artificial potential field functions [19], and avoidance control [20]–[22]. All of these methods, however, cannot guarantee collision avoidance for a large number of agents [18], [23] or cannot prevent the occurrence of deadlocks [7], [24].

Herein, collision avoidance for a large group of agents is addressed using the concept of cellular automata [8]. A cellular automaton is a dynamical system in which time and space are discretized, and which evolution depends on local interactions [25]. It is described by the dimension of the space (number of cells), a discrete set of states for each cell (e.g., occupancy), the neighborhood around each cell (e.g., range of Moore or von Neumann neighborhood), and a set of rules governing the evolution of the cellular automaton based on the current state of the cells and neighboring cells. Cellular automata based methods have been widely used in the modelling and simulation of crowds and pedestrian

This work was supported in part by the U.S. Office of Naval Research Grant No. N0001419WX01589.

E. J. Rodríguez-Seda and C. K. Rico are with the Department of Weapons, Robotics, and Control Engineering at the United States Naval Academy, Annapolis, MD. `rodrigue@usna.edu`, `catalinrico@gmail.com`

dynamics [26], [27]. It has also been applied to path planning algorithms for robots [28]. Particularly, in [25], a cellular automata based model is proposed that guarantees collision avoidance for a large group of agents while maintaining a given formation. Another recent–yet, centralized–path planning method is the use of a triangular grid, which can achieve multi-agent coordination when the agent density is close to 50% [29]. In contrast to previous methods, the algorithm presented in this paper not only is proven to guarantee collision avoidance with dynamics agents at all times, but is also designed to minimize the probability of deadlocks by introducing a *right-to-move* priority rule.

III. Assumptions and Problem Statement

This paper presents a decentralized, collision-free navigation algorithm for an arbitrarily large group of autonomous agents (as long as the workspace is large enough). The goal of the algorithm is for each agent to converge safely to their desired destination in finite time, according to the following assumptions.

1) The workspace $\mathbb{W} \subset \mathbb{R}^2$ is bounded, convex, and divided into equally sized cells $c(x, y, t)$, similar to Fig. 1. Each cell $c(x, y, t)$ is associated with Cartesian coordinates x and y.

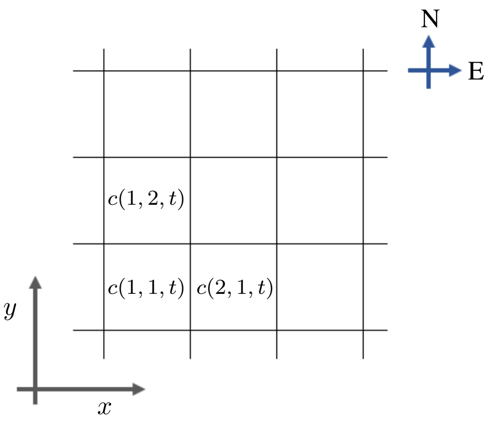


Fig. 1: Workspace Divided in Cells.

2) A cell can take one of two states at any given time $t \in \mathbb{N}_0$, where t represents a time unit and $\mathbb{N}_0$ is the set of natural numbers. The cell can either be occupied by an agent or obstacle ($c(x, y, t) = 1$) or be free ($c(x, y, t) = 0$).
3) A cell can only be occupied by one agent or obstacle at any given time, otherwise, it would imply a collision. Similarly, an agent or obstacle can only occupy one cell at any given time. The positions of the ith agent and jth obstacle are defined as $\mathbf{p}_i(t) = [x_i(t), y_i(t)]^T$ and $\mathbf{q}_j(t) = [x_j(t), y_j(t)]^T$, respectively. Without loss of generality, the dependence on time will be omitted unless deemed necessary.
4) Each agent can sense the location of other agents, including obstacles, within a Moore neighborhood of range $R = 2$ cells (see Fig. 2). This neighborhood is defined as the ith agent's detection region and is given by

$$\mathcal{D}_i(t) = \{c(x_j, y_j, t) : |x_j - x_i| \le R, |y_j - y_i| \le R\}.$$

Furthermore, $\mathcal{D}_i(t)$ will be divided in two parts: the Northwestern part, $\mathcal{D}_i^{NW}(t)$, and the Southeastern part, $\mathcal{D}_i^{SE}(t)$, as shown in the right of Fig. 2.

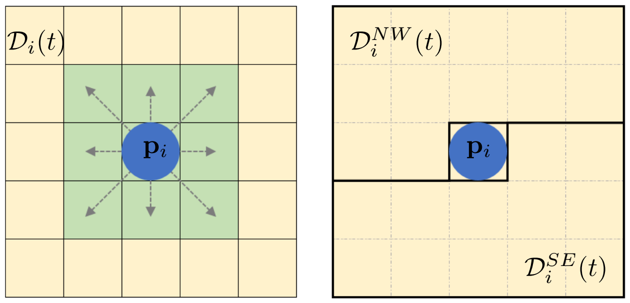


Fig. 2: Agent's Detection Region. The 5×5 cell^2 yellow region represents the ith agent's detection region. The 3×3 cell^2 green region represents the ith agent's range of motion within one unit of time.

5) The dynamics of the agents can be ignored.
6) Agents act cooperatively, i.e., all agents follow the same set of rules.
7) Agents cannot distinguish between obstacles and other agents.
8) Agents can recognize or are aware of the workspace's boundary and the cells at the boundary are therefore assumed as occupied $c(x, y, t) = 1, \; \forall [x, y]^T \notin \mathbb{W}$.
9) Agents can only move up to one cell in any direction within a single unit of time (see green squared area in Fig. 2).
10) Agents and obstacles start all at different cells.
11) No two agents share the same desired destination, $\mathbf{p}_i^d = [x_i^d, y_i^d]^T$, and all destinations are at least $d_\star$ cells apart in any direction, i.e.,

$$\max\{|x_i^d - x_j^d|, |y_i^d - y_j^d|\} \ge d_\star \qquad \forall \; i \neq j.$$

where $d_\star \ge 2$ cells.
12) Obstacles are static and at least $d_\star \ge 2$ cells apart from each other in any direction. Moreover, they are also at least $d_\star$ cells apart from any desired destination and $d_\star$ from the agents' initial positions.
13) The number of obstacles, N_q, plus the number of agents, N_p, is sufficiently small compared to the number of cells, N_c, in the workspace. Specifically, $N_p + N_q << \frac{1}{4} N_c$.

Remark 1: Assumptions (10) through (13) are in place to guarantee that first, the agents do not start from a collision state; and second, that there is at least one collision-free path for all agents to their destination. Increasing $d_\star > 2$ cells guarantees that no agent or obstacle lies in each other's detection region when all agents reach their destination. This is a very common assumption among collision avoidance algorithms [21].

Remark 2: Herein, the distance between any two agents, obstacles, or cells is defined using $\mathcal{L}_\infty$ norm, also known as the Chebyshev distance, i.e., $d_{Ch}(\mathbf{p}_i, \mathbf{p}_j) = \max\{|x_i - x_j|, |y_i - y_j|\}$.

The problem is to define a set of cooperative rules that can guarantee collision avoidance at all times and reduce the likelihood of deadlocks. In cellular automata, these rules are known as transition rules and, together with the dimensions of the workspace, the states of the cells, and the neighborhood of each cell, they define a finite cellular automaton [25]. In what follows, a collision is assumed to happen when 1) two agents or obstacles occupy the same cell, i.e., $\mathbf{p}_i(t) = \mathbf{p}_j(t)$, 2) one agent replaces another agent, i.e., $\mathbf{p}_i(t) = \mathbf{p}_j(t-1)$, or 3) when the paths of two agents $\mathbf{p}_i(t-1) \to \mathbf{p}_i(t), \mathbf{p}_j(t-1) \to \mathbf{p}_j(t)$ intersect in a straight line for some $i \neq j$ and some $t \in \mathbb{N}_0$.

IV. Collision-Free Navigation Algorithm

The navigation algorithm assumes that each agent can only move within its Moore neighborhood of range 1, i.e., a 3×3 cells^2 area centered at the agent's position and illustrated to the left of Fig. 2. The ith agent's detection region is also illustrated in the same figure. The algorithm is governed by the following set of transition rules.

Rule 1: Do not move to an occupied cell, i.e., $\mathbf{p}_i(t+1) \neq [x, y]^T$ if $c(x, y, t) = 1$.

Rule 2: Do not move if intended motion is also a possible move for an agent in a higher priority region. Higher priority region will switch between Northwestern ($\mathcal{D}^{NW}(t)$) and Southeastern ($\mathcal{D}^{SE}(t)$) every T_{switch} units of time, starting with $\mathcal{D}^{NW}(0)$.

Rule 3: If it does not contradict Rules 1 and 2, move to a cell closer to destination:

- if $|x_i^d - x_i| < |y_i^d - y_i|$, then $\mathbf{p}_i(t+1) = \mathbf{p}_i(t) + [0, \operatorname{sgn}(y_i^d - y_i)]^T$;
- if $|x_i^d - x_i| > |y_i^d - y_i|$, then $\mathbf{p}_i(t+1) = \mathbf{p}_i(t) + [\operatorname{sgn}(x_i^d - x_i), 0]^T$;
- if $|x_i^d - x_i| = |y_i^d - y_i|$, then $\mathbf{p}_i(t+1) = \mathbf{p}_i(t) + [\operatorname{sgn}(x_i^d - x_i), \operatorname{sgn}(y_i^d - y_i)]^T$

where $\operatorname{sgn}(x) = 1$ if $x > 0$, $\operatorname{sgn}(x) = -1$ if $x < 0$, and $\operatorname{sgn}(x) = 0$ if $x = 0$

Rule 4: If Rule 3 is not allowed, sequence through motions that shorten the Euclidean distance to destination while satisfying all other rules. Try clockwise first, then counter-clockwise. If no motion that shorten the distance is available, then stay in same cell, i.e., $\mathbf{p}_i(t+1) = \mathbf{p}_i(t)$.

Northwestern priority implies that if an agent $\mathbf{p}_j(t)$ is in ith agent's $\mathcal{D}_i^{NW}(t)$, then, the jth agent has the right to move while the ith agent remains stationary. Otherwise, the agent ith has the right to move and the jth agent remains stationary. Similarly, Southeastern priority implies that if an agent $\mathbf{p}_j(t)$ is in $\mathcal{D}_i^{SE}(t)$, then, the jth agent has the right to move. The rule is meant to avoid collisions and deadlocks by allowing at least one agent to move, but not both. The switching between both priorities aims to further reduce the probabilities of deadlocks, particularly, when a higher priority agent that has reached its destination is in the path of a lower priority agent. The priority regions have been set arbitrarily and the switching frequency can be seen as a control tuning parameter. The complete control strategy is implemented in Algorithm 1.

Algorithm 1: Collision-Free Navigation

```
Data: $\mathbf{p}_i(t), \mathbf{p}_i^d, c(x,y,t) \in \mathcal{D}_i(t)$, $T_{switch}$
Result: $\mathbf{p}_i(t+1)$
if $\mathbf{p}_i(t) \neq \mathbf{p}_i^d$ then
    Determine next position $\mathbf{p}_i^+$ based on Rule 3;
    Verify Rules 1 and 2;
    if $\mathbf{p}_i^+$ violates Rules 1 or 2 then
        for $k \in \{Set\ of\ possible\ motions\}$ do
            Update $\mathbf{p}_i^+$ based on Rule 4;
            Verify Euclidean distance from $\mathbf{p}_i^+$ to $\mathbf{p}_i^d$;
            Verify Rules 1 and 2;
            if $\mathbf{p}_i^+$ violates Rules 1 or 2 or Euclidean
             distance is not decreased then
                Do not move, $\mathbf{p}_i(t+1) = \mathbf{p}_i(t)$;
            else
                Move to $\mathbf{p}_i(t+1) = \mathbf{p}_i^+$;
                break for;
            end
        end
    else
        Move to $\mathbf{p}_i(t+1) = \mathbf{p}_i^+$;
    end
else
    Do not move, $\mathbf{p}_i(t+1) = \mathbf{p}_i(t)$;
end
```

Remark 3: The switching of priority requires time synchronization among agents in order to avoid collisions. To soften this requirement, the switching can be modified with an additional small time window between switching where no agent has a higher priority. Within this time window, an agent will assume a lower priority. This allows some asynchronous behavior among agents.

Remark 4: The evaluation of the Euclidean norm in Rule 4 rather than the Chebyshev norm allows a greater range of motion for some particular cases without increasing the Chebyshev distance. This is due to the fact that the Euclidean distance is in general larger than the Chebyshev distance. For example, consider an agent $\mathbf{p}_i = [x_i, y_i]^T$ with desired position $\mathbf{p}_i^d = [x_i + 2, y_i + 1]^T$ and surrounding cells empty except for $c(x_i + 2, y_i, t) = c(x_i + 1, y_i + 1, t) = 1$. Under Chebyshev metric and Southeastern priority, no other neighboring cells results in a shorter distance. However, using the Euclidean norm, moving to $c(x_i, y_i + 1, t)$ reduces the Euclidean distance while maintaining the same Chebyshev distance. The move eventually can decrease the chances of a deadlock.

Theorem 1: Consider a group of N_p agents and N_q obstacles and suppose that Assumptions (1) through (13) are satisfied. Let the group of agents evolve according to the Collision-Free Navigation Algorithm. Then, the agents are

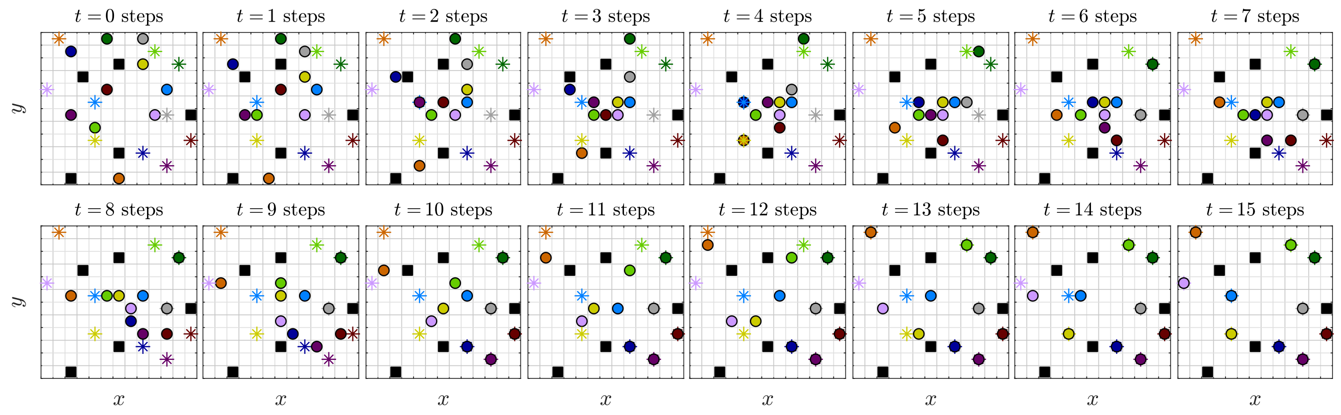


Fig. 3: Sequential Motion of Agents for $N_p = 10$ and $N_q = 5$ in a 13×12 cell^2 Workspace. Agents, obstacles, and desired destinations are denoted by the colored circles, black squares, and colored asterisks, respectively.

guaranteed to avoid collision at all times.

Proof: A collision takes place if for some time $t > 0$ one have that: 1) two or more agents (including obstacles) occupy the same cell; 2) an agent moves to a previously occupied cell, i.e., $c(x, y, t-1) = 1$; or 3) the paths of two agents intersect. The second scenario is a direct contradiction of Rule 1 and, therefore, it is guaranteed not to take place. The third scenario is ruled out by Rule 2 as only one agent is allowed to move to a potential collision cell. One is left to prove that the first scenario cannot take place.

The first scenario assumes that at least two or more agents move to the same pre-empty cell, i.e., $\mathbf{p}_i(t+1) = \mathbf{p}_j(t+1) = [x, y]^T$ for $c(x, y, t) = 0$. Without loss of generality, consider a collision between two agents. Given that all agents can only move one cell at a time, both agents should have been able to detect each other at least one time step before the collision. Since for any arrangement between the two agents, one will always have priority (Rule 2), the second should have not been able to move to the same cell. Therefore, such a collision move is not allowed by the algorithm and the agents should not collide. ■

Theorem 1 guarantees collision avoidance for a group agents and static obstacles that evolve according to the defined transition rules. Note that it does not guarantee anything about avoiding deadlocks which is a hard problem to overcome, particularly, with non-cooperative obstacles [16]. A simulation example of the evolution of ten agents with five static obstacles in a 13×12 cells^2 workspace is given in Fig. 3 for $d_\star = 2$ cells and $T_{switch} = 10$ time steps. The figure shows the motion for different instances of time and demonstrate how, at least in this case, the agents are able to converge to their desired destination without colliding.

V. Simulations

To evaluate the performance of the proposed algorithm, a series of simulations using MATLAB were performed by varying the agents' initial conditions, desired destinations, and obstacles' locations for different number of agents and obstacles. A total of 200 cases per scenario were evaluated. A scenario was defined by the number of agents, static obstacles, and the choice of minimum distance $d_\star$ among obstacles and desired positions. The number of agents varied from 5 to 20 agents, whereas the number of obstacles ranged from 0 to 20. In total, 48 different scenarios were evaluated under 200 different initial conditions and desired destinations (i.e., cases). The initial conditions and agents' destinations were chosen uniformly at random in a 30×30 cell^2 workspace subject to the following constraints:

- $d_{CH}(\mathbf{p}_i(0), \mathbf{p}_i^d) \geq 10$ cells $\forall\ i$,
- $d_{CH}(\mathbf{p}_i(0), \mathbf{p}_j(0)) \geq 1$ cells $\forall\ i \neq j$,
- $d_{CH}(\mathbf{p}_i(0), \mathbf{q}_j) \geq 2$ cells $\forall\ i, j$,
- $d_{CH}(\mathbf{p}_i^d, \mathbf{p}_j^d) \geq d_\star\ \forall\ i \neq j$,
- $d_{CH}(\mathbf{q}_i, \mathbf{q}_j) \geq d_\star\ \forall\ i \neq j$, and
- $d_{CH}(\mathbf{p}_i^d, \mathbf{q}_j) \geq d_\star\ \forall\ i, j$

where $d_\star$ varied between 2 and 4 cells. The first constraint was chosen in order to increase the chances of interactions among agents (i.e., each agent needs to travel at least 10 cells to reach its destination). The other constraints were chosen according to Assumptions (11) through (13) of Section III when $d_\star = 2$ cells. Simulations for $d_\star > 2$ cells were carried to illustrate how the algorithm performs under more restricted conditions. The algorithm switched every $T_{switch} = 10$ time steps between both priority regions.

A. Performance Criteria

The metrics used to evaluate the performance of the algorithm were number of collisions, percentage of deadlocks, and the time to completion. Ideally, there should be no collisions, the percentage of deadlocks per scenario should be zero, and the time to completion should be small. The percentage of deadlocks is counted as the total number of cases that led to at least one deadlock divided by the total number of cases per scenario (i.e., 200 cases). The time to completion was taken as the minimum time t for which all agents reached their destinations.

In addition to the above metrics, all agents should ideally travel the shortest distance from their initial positions to their destinations. To quantify this behavior, define the *Normalized Collision-Free Traveled Distance* (NCFTD) as the total

Chebyshev distance travelled by the ith agent divided by the Chebyshev distance between the initial position and the agent's destination. Mathematically,

$$\mathrm{NCFTD}_i = \frac{\sum_{t=1}^{\infty} d_{Ch}(\mathbf{p}_i(t-1), \mathbf{p}_i(t))}{d_{Ch}(\mathbf{p}_i(0), \mathbf{p}_i^d)}.$$

Note that, ideally, NCFTD_i should be 1 and that larger values indicate a poor performance (i.e., that the agents took a route other than the shortest path).

Similarly, define the average NCFTD among all agents in a group as

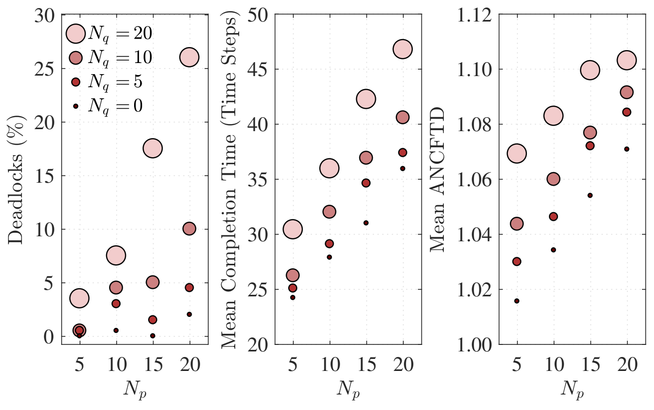
$$\mathrm{ANCFTD} = \frac{\sum_{i=1}^{N_p} \mathrm{NCFTD}_i}{N_p}.$$

Once again, this value should ideally be 1.

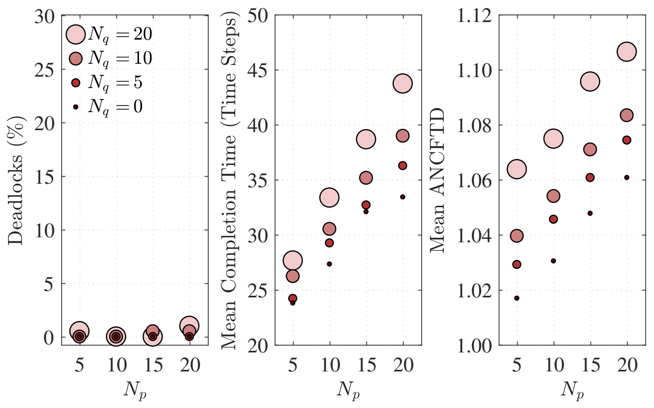
B. Results

Fig. 4a illustrates the results for different scenarios when the distance among obstacles and the distance between any other obstacle or desired destination was at least $d_\star = 2$ cells. It can be observed that the number of deadlocks, the time to completion, and the average distance traveled by the agents, ANCFTD, increased with an increase in the number of agents or an increase in the number of obstacles. No collisions took place on any of the scenarios. It should be noticed, however, that even under the absence of static obstacles, some scenarios led to deadlocks (e.g., 3.5% for $N_p = 20$ agents).

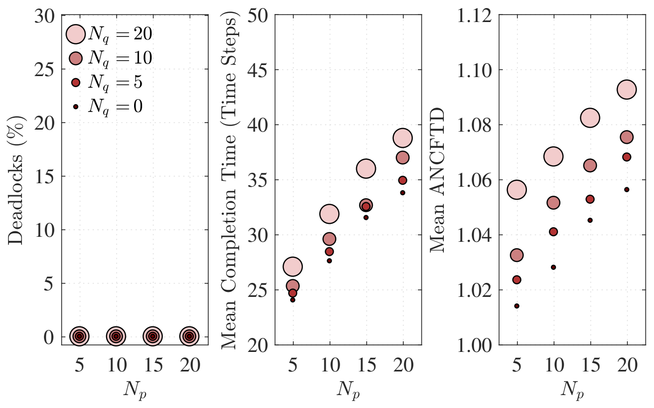
A similar series of simulations were then performed by restricting the Chebyshev distance between any two obstacles or desired destinations to be at least 3 cells, in Fig. 4b, and 4 cells, in Fig. 4c. The increase in the distance among obstacles as well as desired destinations allows more mobility among agents and guarantees that no other agent or obstacle lies within their detection region once all agents have reached their destination (a common assumption among collision avoidance algorithms [21]). It can be seen that in all scenarios, the time to completion as well as the average distance traveled by agents increased with the number of agents or obstacles. The number of deadlocks was reduced to zero when $d_\star = 4$ cells and near zero when $d_\star = 3$ cells. Note, however, that all agents converged to their destination when no static obstacles where included. Comparing the algorithm under different $d_\star$ distances, it can be seen that all measures of performance improved with an increase in $d_\star$.



(a) $d_\star = 2$ cells



(b) $d_\star = 3$ cells



(c) $d_\star = 4$ cells

Fig. 4: Results for $d_{Ch}(\mathbf{q}_i, \mathbf{q}_j) \geq d_\star$, $d_{Ch}(\mathbf{p}_i^d, \mathbf{p}_j^d) \geq d_\star$, and $d_{Ch}(\mathbf{p}_k^d, \mathbf{q}_j) \geq d_\star$, $\forall\ i, j, k, i \neq j$, when (a) $d_\star = 2$ cells, (b) $d_\star = 3$ cells, and (c) $d_\star = 4$ cells, respectively.

VI. Conclusions

This paper introduced a cellular automata based, decentralized cooperative collision avoidance algorithm for a team of agents with static obstacles. The algorithm assumes that agents can detect the position of other agents and obstacles within a Moore neighborhood of range 2, while their range of motion is the Moore neighborhood of range 1. The algorithm is theoretically proven to guarantee collision avoidance and is designed to reduce the occurrence of deadlocks among agents. Although the cooperative algorithm cannot guarantee deadlock avoidance in all scenarios, it is shown via an extensive series of simulation examples that the agents are able to converge to their destination most of the time, particularly, when the density of agents and obstacles is small.

Future research directions include increasing the velocity (cells per one time step) of agents and increasing the range of motion when a potential deadlock has been recognized.

References

[1] D. P. Agrawal, "Applications of sensor networks," in *Embedded Sensor Systems*. Singapore: Springer, 2017, pp. 35–63.

[2] U.S. Department of the Navy, "U.S. Navy program guide 2017," Washington, DC, Tech. Rep., 2017.

[3] E. J. Rodríguez-Seda, J. J. Troy, C. A. Erignac, P. Murray, D. M. Stipanović, and M. W. Spong, "Bilateral teleoperation of multiple mobile agents: Coordinated motion and collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 4, pp. 984–992, July 2010.

[4] L. Bakule, "Decentralized control: An overview," *Annual Reviews in Control*, vol. 32, no. 1, pp. 87–98, 2008.

[5] L. Sabattini, N. Chopra, and C. Secchi, "Decentralized connectivity maintenance for cooperative control of mobile robotic systems," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1411–1423, 2013.

[6] E. J. Rodríguez-Seda, "Decentralized trajectory tracking with collision avoidance control for teams of unmanned vehicles with constant speed," in *Proc. Am. Control Conf.*, Portland, OR, June 2014, pp. 1216–1223.

[7] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proc. IEEE International Conference on Robotics and Automation*, 1991, pp. 1398–1404.

[8] B. Chopard, *Cellular automata modeling of physical systems*. Springer, 2012.

[9] J. K. Kuchar and L. C. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 179–189, 2000.

[10] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey," *Robotica*, vol. 33, no. 3, pp. 463–497, 2014.

[11] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *IEEE Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[12] H. P. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI Mag.*, vol. 9, no. 2, pp. 61–74, 1988.

[13] E. J. Rodríguez-Seda, C. Tang, D. M. Stipanović, and M. W. Spong, "Trajectory tracking with collision avoidance for nonholonomic vehicles with acceleration constraints and limited sensing," *Int. J. Robot. Res.*, vol. 33, no. 12, pp. 1569–1592, Oct. 2014.

[14] S. Mastellone, D. M. Stipanović, C. R. Graunke, K. A. Intlekofer, and M. W. Spong, "Formation control and collision avoidance for multi-agent non-holonomic systems: Theory and experiments," *Int. J. Robot. Res.*, vol. 27, no. 1, pp. 107–126, Dec. 2008.

[15] E. J. Rodríguez-Seda and J. J. Dawkins, "Decentralized cooperative collision avoidance control for unmanned rotorcraft with bounded acceleration," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 11, pp. 2445–2454, 2018.

[16] E. J. Rodríguez-Seda, D. M. Stipanović, and M. W. Spong, "Guaranteed collision avoidance for autonomous systems with acceleration constraints and sensing uncertainties," *J. Optim. Theory Appl.*, vol. 168, no. 3, pp. 1014–1038, Mar. 2016.

[17] E. J. Rodríguez-Seda and M. W. Spong, "Guaranteed safe motion of multiple Lagrangian systems with limited actuation," in *Proc. IEEE Conf. Decision Control*, Maui, HI, Dec. 2012, pp. 2773–2780.

[18] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, 1998.

[19] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986.

[20] G. Leitmann and J. Skowronski, "Avoidance control," *J. Optim. Theory Appl.*, vol. 23, no. 4, pp. 581–591, Dec. 1977.

[21] D. M. Stipanović, P. F. Hokayem, M. W. Spong, and D. Šiljak, "Cooperative avoidance control for multiagent systems," *J. Dyn. Syst. Meas. Control*, vol. 129, pp. 699–707, Sept. 2007.

[22] E. J. Rodríguez-Seda, "Self-triggered collision avoidance control for multi-vehicle systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seattle, WA, May 2015, pp. 461–466.

[23] J. van den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, Apr. 2011, pp. 3475–3482.

[24] E. J. Rodríguez-Seda and D. M. Stipanović, "Guaranteed collision avoidance with discrete observations and limited actuation," in *Advances in Intelligent Vehicles*, ser. Intelligent Systems, Y. Chen and L. Li, Eds. Academic Press, 2013, pp. 89–110.

[25] K. Ioannidis, G. C. Sirakoulis, and I. Andreadis, "Cellular ants: A method to create collision free trajectories for a cooperative robot team," *Robotics and Autonomous Systems*, vol. 59, no. 2, pp. 113–127, 2011.

[26] C. Burstedde, K. Klauck, A. Schadschneider, and J. Zittartz, "Simulation of pedestrian dynamics using a two-dimensional cellular automaton," *Physica A: Statistical Mechanics and its Applications*, vol. 295, no. 3-4, pp. 507–525, 2001.

[27] C. Feliciani and K. Nishinari, "An improved cellular automata model to simulate the behavior of high density crowd and validation by experimental data," *Physica A: Statistical Mechanics and its Applications*, vol. 451, pp. 135–148, 2016.

[28] P. G. Tzionas, A. Thanailakis, and P. G. Tsalides, "Collision-free path planning for a diamond-shaped robot using two-dimensional cellular automata," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 2, pp. 237–250, 1997.

[29] R. Chinta, S. D. Han, and J. Yu, "Coordinating the motion of labeled discs with optimality guarantees under extreme density," in *Proc. WAFR*, Mérica, Mexico, Dec. 2018, pp. 1–16.