A Distributed Simplex Architecture for Multi-Agent Systems*

Usama Mehmood[1], Scott D. Stoller[1], Radu Grosu[2],
Shouvik Roy[1], Amol Damare[1], and Scott A. Smolka[1]

[1] Department of Computer Science, Stony Brook University, USA
[2] Department of Computer Engineering, Technische Universität Wien, Austria

Abstract. We present the *Distributed Simplex Architecture* (DSA), a new runtime assurance technique that provides safety guarantees for multi-agent systems (MASs). DSA is inspired by the Simplex control architecture of Sha et al., but with some significant differences. The traditional Simplex approach is limited to single-agent systems or a MAS with a centralized control scheme. DSA addresses this limitation by extending the scope of Simplex to include MASs under distributed control. In DSA, each agent runs a local instance of traditional Simplex such that the preservation of safety in the local instances implies safety for the entire MAS. Control Barrier Functions play a critical role. They are used to define DSA's core components (the baseline controller and the decision module's switching logic between advanced and baseline controllers) and to verify the safety of a DSA instance in a distributed manner. We provide a general proof of safety for DSA, and present experimental results for several case studies, including flocking with collision avoidance, safe navigation of ground rovers through way-points, and the safe operation of a microgrid.

Keywords: Runtime assurance · Simplex architecture · Control Barrier Functions · Distributed flocking · Reverse switching.

1 Introduction

A multi-agent system (MAS) is a group of autonomous, intelligent agents that work together to solve tasks and carry out missions. MAS applications include the design of power systems and smart-grids [1, 2], autonomous control of robotic swarms for monitoring, disaster management, military battle systems, etc. [3], and sensor networks [4]. Many MAS applications are safety-critical. It is therefore paramount that MAS control strategies ensure safety.

In this paper, we present the *Distributed Simplex Architecture* (DSA), a new runtime assurance technique that provides safety guarantees for MASs under distributed

* This work is supported in part by NSF awards OIA-2040599, CCF-1918225, CCF-1954837, CPS-1446832 and ONR award N000142012751.

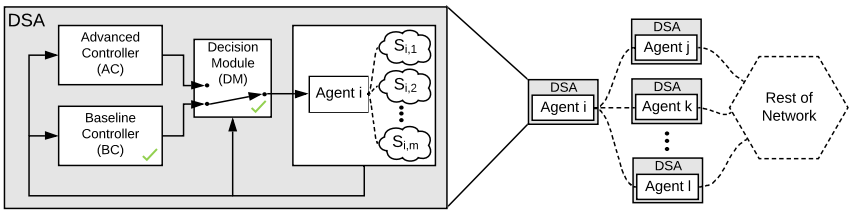


Fig. 1: Architectural overview of DSA. Agents are homogeneous and operate under DSA control; the figure zooms in on DSA components for agent i. Sensed state of agent i's j^{th} neighbor denoted as $S_{i,j}$. AC, BC, and DM take as input the state of the agent and its neighbors.

control. DSA is inspired by Sha et al.'s Simplex Architecture [5, 6], but differs from it in significant ways. The Simplex Architecture provides runtime assurance of safety by switching control from an unverified (hence potentially unsafe) *advanced controller* (AC) to a verified-safe *baseline controller* (BC), if the action produced by the AC could result in a safety violation in the near future. The switching logic is implemented in a verified *decision module* (DM). The applicability of the traditional Simplex Architecture is limited to systems with a centralized control architecture.

DSA, illustrated in Fig. 1, addresses this limitation by re-engineering the traditional Simplex architecture to widen its scope to include MASs. Also, as in [7], it implements *reverse switching* by reverting control back to the AC when it is safe to do so.

In DSA, for each agent, there is a verified-safe BC and a verified switching logic such that if all agents operate under DSA, then safety of the MAS is guaranteed. The BC and DM along with the AC are distributed and depend only on local information. DSA itself is *distributed* in that it involves one local instance of traditional Simplex per agent such that the conjunction of their respective safety properties yields the desired safety property for the entire MAS. For example, consider our flocking case study, where a group of robotic agents is moving cohesively, and we want to establish collision-freedom for the entire MAS. This can be accomplished in a distributed manner by showing that each local instance of Simplex, say for agent i, ensures collision-freedom for agent i and its neighboring agents.

DSA allows agents to switch their mode of operation independently. At any given time, some agents may be operating in AC mode while others are operating in BC mode. Our approach to the design of the BC and DM leverages *Control Barrier Functions* (CBFs), which have been used to synthesize safe controllers [8, 9, 10], and are closely related to Barrier Certificates used for safety verification of closed dynamical systems [11, 12]. A CBF is a mapping from the system's (i.e., plant's) state space to a real number, with its zero level-set partitioning the state space into safe and unsafe regions. If certain inequalities on the derivative of the CBF in the direction of the state trajectories (also known as Lie derivative) are satisfied, then the corresponding control actions are considered safe (admissible).

In DSA, the BC is designed as an optimal controller with the goal of increasing a utility function based on the Lie derivatives of the CBFs. As CBFs are a measure of the safety of a state, optimizing for control actions with a higher Lie derivative values

provides a direct way to make the state safer. The safety of the BC is further guaranteed by constraining the control action to remain in a set of admissible actions that satisfy certain inequalities on the Lie derivatives of the CBFs. CBFs are also used in the design of the switching logic, as they provide an efficient method for checking whether an action could lead to a safety violation during the next time step.

We demonstrate the effectiveness of DSA on several example MASs, including a flock of robots moving coherently while avoiding inter-agent collisions, ground rovers safely navigating through a series of way-points, and safe operation of a microgrid.

2 Background

2.1 Simplex Architecture

The Simplex Control Architecture relies on a verified-safe baseline controller (BC) in conjunction with the verified switching logic of the Decision Module (DM) to guarantee the safety of the plant, while permitting the use of an unverifiable, high-performance advanced controller (AC); see agent i in Fig. 1.

Let the *admissible* states of a system be those which satisfy all safety constraints and operational limits. All other states are *inadmissible*. The goal of the Simplex Architecture is to ensure that the system never enters an inadmissible state. The set $\mathcal{R}$ of *recoverable* states is a subset of the admissible states such that the BC, starting from any state in $\mathcal{R}$, guarantees that all future states are also in $\mathcal{R}$. The recoverable set takes into account the inertia of the physical system, giving the BC enough time to preserve safety.

The DM's *forward switching condition* (FSC) evaluates the control action proposed by the AC and decides whether to switch to the BC. A common technique used to develop an FSC is to shrink the recoverable region by a margin based on the maximum time derivative of the state and the length of a time step, and switch to BC if the current state lies outside this smaller set.

2.2 Control Barrier Functions

Control Barrier Functions (CBFs) [13, 14] are an extension of the Barrier Certificates used for safety verification of hybrid systems [11, 12]. CBFs are a class of Lyapunov-like functions used to guarantee safety for nonlinear control systems by assisting in the design of a class of safe controllers that establish the forward-invariance of safe sets [10, 15]. Our presentation of CBFs is based on [14].

Consider a nonlinear affine control system

$$\dot{x} = f(x) + g(x)u \tag{1}$$

with state $x \in D \subset \mathbb{R}^n$, control input $u \in U$, and functions f and g that are locally Lipschitz. The set $\mathcal{R}$ of recoverable states is defined as the super-level set of a continuously differentiable function $h : D \subset \mathbb{R}^n \to \mathbb{R}$. The recoverable set $\mathcal{R}$ and its boundary $\partial\mathcal{R}$ are given by:

$$\mathcal{R} = \{x \in D \subset \mathbb{R}^n | h(x) \geq 0\} \tag{2}$$

$$\partial\mathcal{R} = \{x \in D \subset \mathbb{R}^n | h(x) = 0\} \tag{3}$$

The time derivative of $h(x)$ along the direction of the state evolution is

$$\frac{dh(x)}{dt} = \frac{\partial h(x)}{\partial x}\dot{x} = \frac{\partial h(x)}{\partial x}(f(x) + g(x)u) \tag{4}$$

which can be restated in the Lie derivative formulation as

$$\frac{dh(x)}{dt} = L_f h(x) + L_g h(x)u. \tag{5}$$

For all $x \in \mathcal{D}$, if there exists an extended class $\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}$ (strictly increasing and $\alpha(0) = 0$) such that the following condition on the Lie derivative of h is satisfied:

$$\sup_{u \in U} [L_f h(x) + L_g h(x)u + \alpha(h(x))] \geq 0 \tag{6}$$

then h is a valid CBF. Condition (6) implies the existence of a control action for all $x \in D$, such that the Lie derivative of h is bounded from below by $-\alpha(h(x))$. Furthermore, for $x \in \partial\mathcal{R}$, condition (6) reduces to a result for set invariance known as Nagumo's theorem [16, 17]. Condition (6) is used to define the set $K(x)$ of control actions that establish the forward invariance of set $\mathcal{R}$; i.e., starting from $x \in \mathcal{R}$, the state will always remain inside the set $\mathcal{R}$:

$$K(x) = \{u \in U : L_f h(x) + L_g h(x)u + \alpha(h(x)) \geq 0\} \tag{7}$$

The following theorem is from [14].

Theorem II.1 *For the control system given in Eq. (1) and recoverable set $\mathcal{R}$ defined in (2) as the super-level set of some continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$, if h is a CBF for all $x \in D$ and $\frac{\partial h}{\partial x} \neq 0$ for all $x \in \partial\mathcal{R}$, then any controller u such that $\forall x \in D : u(x) \in K(x)$ ensures forward-invariance of the set $\mathcal{R}$.*

Proof. See [14]. Condition (6) on the Lie derivative of h reduces, on the boundary of $\mathcal{R}$, to the set invariance condition of Nagumo's theorem: for $x \in \partial\mathcal{R}$, $\dot{h} \geq -\alpha(h(x)) = 0$. Hence, according to Nagumo's theorem [16, 17], the set $\mathcal{R}$ is forward-invariant.

A widely used technique for the synthesis of CBFs is SOS-optimization [18] based search, which can be applied to a polynomial approximation of the systems dynamics. Other methods of synthesizing CBFs are surveyed in [14].

3 Distributed Simplex Architecture

This section describes the Distributed Simplex Architecture (DSA). We formally introduce the MAS safety problem and then discuss the main components of DSA, namely, the distributed baseline controller (BC) and the distributed decision module (DM).

Let an instance of DSA be *symmetric* if every agent uses the same switching condition and baseline controller. Moreover, DSA, or more precisely the MAS it is controlling, is *homogeneous* if every constituent agent is an instance of the same plant model.

Consider a MAS consisting of k homogeneous agents, denoted as $\mathcal{M} = \{1, ..., k\}$, where the nonlinear control affine dynamics for the i^{th} agent are:

$$\dot{x}_i = f(x_i) + g(x_i)u_i \tag{8}$$

where $x_i \in D \in \mathbb{R}^n$ is the state of agent i and $u_i \in U \subset \mathbb{R}^m$ is its control input. For an agent i, we define the set of its *neighbors* $\mathcal{N}_i \subseteq \mathcal{M}$ as the agents whose state is accessible to i either through sensing or communication. Depending on the application, the set of neighbors could be fixed or vary dynamically. For example, in our flocking case study (Section 4), agent i's neighbors (in a given state) are the agents within a fixed distance r of agent i; we assume agent i can accurately sense the positions and velocities of these agents.

We denote a combined state of all of the agents in the MAS as the vector $x = \{x_1^T, x_2^T, \ldots x_k^T\}^T$ and denote a state of the neighbors of agent i (including i itself) as $x_{\mathcal{N}_i}$. DSA uses discrete-time control: the DMs and controllers execute every η seconds. We assume that all agents execute their DM and controllers simultaneously; this assumption simplifies the analysis.

Admissible States The set of admissible states $\mathcal{A} \subset \mathbb{R}^{kn}$ consists of all states that satisfy the safety constraints. A constraint $C : D^k \to \mathbb{R}$ is a function from k-agent MAS states to the reals. In this paper, we are primarily concerned with *binary constraints* (between neighboring agents) of the form $C_{ij} : D \times D \to \mathbb{R}$, and *unary constraints* of the form $C_i : D \to \mathbb{R}$. Hence, the set of admissible states, $\mathcal{A} \subset \mathbb{R}^{kn}$ are the MAS states of $\mathbf{x} \in \mathbb{R}^{kn}$ such that all of the unary and binary constraints are satisfied.

Formally, a symmetric instance of DSA is tasked with solving the following problem. Given a MAS defined as in Eq. (8) and $\mathbf{x}(0) \in \mathcal{A}$, design a BC and DM to be used by all agents such that the MAS remains safe; i.e. $\mathbf{x}(t) \in \mathcal{A}, \forall\, t > 0$.

Recoverable States For each agent i, the local admissible set $\mathcal{A}_i \subset \mathbb{R}^n$ is the set of states $x_i \in \mathbb{R}^n$ that satisfy all unary constraints. The set $\mathcal{S}_i \subset \mathcal{A}_i$ is defined as the super-level set of the CBF $h_i : \mathbb{R}^n \to \mathbb{R}$, which is designed to ensure forward-invariance of $\mathcal{A}_i$. Similarly, for a pair of neighboring agents i, j where $i \in \mathcal{M}, j \in \mathcal{N}_i$, the pairwise-admissible set $\mathcal{A}_{ij} \subset \mathbb{R}^{2n}$ is the set of pairs of states that satisfy all binary constraints. The set $\mathcal{S}_{ij} \subset \mathcal{A}_{ij}$ is defined as the super-level set of the CBF $h_{ij} : \mathbb{R}^{2n} \to \mathbb{R}$ designed to ensure forward-invariance of $\mathcal{A}_{ij}$. The recoverable set $\mathcal{R}_{ij} \subset \mathbb{R}^{2n}$, for a pair of neighboring agents i, j where $i \in \mathcal{M}, j \in \mathcal{N}_i$, is defined in terms of $\mathcal{S}_i$, $\mathcal{S}_j$ and $\mathcal{S}_{ij}$.

$$\mathcal{S}_i = \{x_i \in \mathbb{R}^n | h_i(x_i) \geq 0\} \tag{9}$$

$$\mathcal{S}_{ij} = \{(x_i, x_j) \in \mathbb{R}^{2n} | h_{ij}(x_i, x_j) \geq 0\} \tag{10}$$

$$\mathcal{R}_{ij} = (\mathcal{S}_i \times \mathcal{S}_j) \cap \mathcal{S}_{ij} \tag{11}$$

The recoverable set $\mathcal{R} \subset \mathcal{A}$ for the entire MAS is defined as the set of system states for which $(x_i, x_j) \in \mathcal{R}_{ij}$ for every pair of neighboring agents i, j. Note that if agent i and j's controllers satisfy the following constraints based on the Lie derivatives of h_i, h_j

and h_{ij}, similar to the constraints in (7), the pairwise state of agents i and j will remain in $\mathcal{R}_{ij}$ according to Theorem II.1.

$$L_f h_i(x_i) + L_g h_i(x_i) u_i + \alpha(h_i(x_i)) \geq 0 \tag{12a}$$

$$L_f h_j(x_j) + L_g h_j(x_j) u_j + \alpha(h_j(x_j)) \geq 0 \tag{12b}$$

$$L_f h_{ij}(x_i, x_j) + L_g h_{ij}(x_i, x_j) \begin{bmatrix} u_i \\ u_j \end{bmatrix} + \alpha(h_{ij}(x_i, x_j)) \geq 0 \tag{12c}$$

Constraint Partitioning Note that the constraints in (12) are linear in the control variable. For ease of notation, we write the unary constraints as $A_i u_i \leq b_i$ and the binary constraints as $\left[\begin{smallmatrix} P_{ij} & Q_{ij} \end{smallmatrix}\right] \left[\begin{smallmatrix} u_i \\ u_j \end{smallmatrix}\right] \leq b_{ij}$.

The binary constraint in (12c) is a condition on the control actions of a pair of agents. For a centralized MAS, the global controller can pick coordinated actions for agents i and j to ensure the binary constraint (12c) is satisfied. For a decentralized MAS, however, the distributed control of the two agents cannot independently satisfy the binary constraint without running an agreement protocol.

As DSA is a distributed control framework, we solve the problem of the satisfaction of binary constraints by partitioning a binary constraint into two unary constraints such that the satisfaction of the unary constraints by agents i and j implies the satisfaction of the binary constraint (but not necessarily vice versa) [10].

$$\left.\begin{matrix} P_{ij} u_i \leq b_{ij}/2 \\ Q_{ij} u_j \leq b_{ij}/2 \end{matrix}\right\} \Rightarrow \begin{bmatrix} P_{ij} & Q_{ij} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \end{bmatrix} \leq b_{ij} \tag{13}$$

Moreover, the equal partitioning of the binary constraint ensures that the agents share an equal responsibility in maintaining it. The admissible control space for agent i, denoted by $\mathcal{L}_i$, is the intersection of half-spaces of the hyper-planes defined by the linear constraints.

$$\mathcal{L}_i = \{u_i \in U \mid \forall j \in \mathcal{N}_i : A_i u_i \leq b_i \wedge P_{ij} u_i \leq b_{ij}/2\} \tag{14}$$

Theorem III.1 *Given a MAS indexed by $\mathcal{M}$ and with dynamics as in (8), if the controller for each agent $i \in \mathcal{M}$ chooses an action $u_i \in \mathcal{L}_i$, thereby satisfying the Lie derivative constraints on the respective CBFs, and $\mathbf{x}(0) \in \mathcal{R}$, then the MAS is guaranteed to remain safe.*

Proof. If all agents choose an action from their respective admissible control spaces $\mathcal{L}_i$, then the forward-invariance of the set $\mathcal{S}_i$ for all $i \in \mathcal{M}$ and $\mathcal{S}_{ij}$ for all $i \in \mathcal{M}, j \in \mathcal{N}_i$ is established by Theorem II.1. Therefore, $\mathcal{R}_{ij}$ is forward-invariant for all $i \in \mathcal{M}, j \in \mathcal{N}_i$ and consequently $\mathcal{R}$ is forward-invariant.

3.1 Baseline Controller

The BC is a distributed controller tasked with keeping the state of an agent in the safe region. For an agent i, the BC's control law depends on i's state x_i and the states of

its neighbors $x_j, \forall j \in \mathcal{N}_i$. In our design, the BC is strictly focused on safety, leaving mission-critical objectives to the AC. Specifically, the BC is designed to move the system away from unsafe states and toward safer states as quickly as possible.

We design the BC as the solution to the following constrained multi-objective optimization (MOO) problem where the utility function is the weighted sum of objective functions based on the Lie derivatives of the CBFs h_i and h_{ij} introduced above:

$$\begin{aligned} u_i^* = \underset{u_i}{argmax} \quad & \frac{1}{h_i}(L_f h_i + L_g h_i u_i) + \sum_{j \in \mathcal{N}_i} \frac{1}{h_{ij}}(L_f h_{ij} + L_g h_{ij} \begin{bmatrix} u_i \\ 0 \end{bmatrix}) \\ \text{s.t.} \quad & u_i \in \mathcal{L}_i \end{aligned} \tag{15}$$

The bottom component of the column vector in the last term is agent i's prediction for agent j's next control action u_j. Since we consider MASs in which agents are unable to communicate their planned control actions, agent i simply predicts that $u_j = 0$. This approach has been shown to work well in prior work on distributed model-predictive control for flocking [19] , where the control actions are accelerations. Despite its complex form, at any given time, the utility function in Eq. (15) is linear in u_i, as the values of all other quantities are fixed. Since the constraints are also linear, the optimization problem in Eq. (15) is a linear program and hence can be efficiently solved in real-time.

Recall that, by definition, the CBFs quantify the degree of safety of a state with respect to the given safety constraints, with larger (positive) values indicating safer states. A positive value of the Lie derivative indicates that the proposed action will lead to a state that has a higher CBF value and therefore is safer.

The solution to the optimization problem (15) is a control action that maximizes the weighted sum of the Lie derivatives of the CBFs. We note that in a weighted-sum formulation of a MOO problem, it is possible that some objective functions are negative in the optimal solution. We ensure the selected action u_i is safe by constraining u_i to be in the admissible control space $\mathcal{L}_i$, defined in Eq. (14).

The weights in the utility function in Eq. (15) prioritize certain safety constraints over others. We use state-dependent weights in the form of inverses of the CBFs, thereby giving more weight to maximizing the Lie derivatives of CBFs corresponding to safety constraints that are closer to being violated.

3.2 Decision Module

Each agent's DM implements the switching logic for both forward and reverse switching. Control is switched from the AC to the BC if the *forward switching condition* (FSC) is true. Similarly, control is reverted back to the AC (from the BC) if the *reverse switching condition* (RSC) is true. For an agent i, the state of the DM is denoted as $DM_i \in \{AC, BC\}$, with $DM_i = AC$ ($DM_i = BC$) indicating that the advanced (baseline) controller is in control. DSA starts with all agents in the AC mode; i.e., $DM_i(t) = AC$ for all $t \leq 0$ and $i \in \mathcal{M}$; this is justified by the assumption that $\mathbf{x}(0) \in \mathcal{R}$.

We derive the switching conditions from the CBFs as follows. To ensure safety, the FSC must be true in a state $x_{\mathcal{N}_i}(t)$ if an unrecoverable state is reachable from $x_{\mathcal{N}_i}(t)$

in one time step η. The check for one-step reachability of an unrecoverable state is based on computing the Taylor series approximation of the CBF at the current time t, and evaluating it one time step in the future, i.e., at time $t + \eta$. The Taylor series approximation of the CBF depends on its time-derivatives which can be regarded as a function of time based on the dynamics of the system for a given value of the control input; we take that control input of agent i to be the command proposed by the AC at time t and use the worst-case commands as the control inputs for the neighboring agents $j \in \mathcal{N}_i$. The worst-case commands are defined as the control inputs that minimize the value of the Taylor approximation of the CBF. If the Taylor series approximation of any of the CBFs is negative during the next time step η, we switch control to the BC. We denote the Taylor series approximation of the CBF h as $\hat{h}$. This results in an FSC of the following form:

$$FSC(x_{\mathcal{N}_i}, t) = \exists t_c \in (t,\, t+\eta] \mid (\hat{h}_i(t_c) < 0) \vee (\exists j \in \mathcal{N}_i \mid \hat{h}_{ij}(t_c) < 0) \qquad (16)$$

We derive the RSC using a similar approach, except the inequalities are reversed, the worst-case commands are used as the control inputs for all the agents, and an m-time-step reachability check with $m > 1$ is used; the latter is to prevent frequent back-and-forth switching between the AC and BC. The RSC holds if the Taylor series approximations of all the CBFs remain positive during the next $m \cdot \eta$ seconds.

$$RSC(x_{\mathcal{N}_i}, t) = \forall t_c \in (t,\, t + m \cdot \eta] \mid (\hat{h}_i(t_c) > 0) \wedge (\forall j \in \mathcal{N}_i \mid \hat{h}_{ij}(t_c) > 0) \quad (17)$$

We experimented with various orders of Taylor series approximations in our case studies. Since the time step η is typically small, even low-order Taylor series approximation gives very good results, i.e., $\hat{h}(t + \eta)$ is very close to the exact value $h(t + \eta)$. The switching condition can be made more rigorous by taking into account the remainder error in the Taylor series approximation; Taylor's theorem provides a bound on the remainder error. We will explore this idea in future work.

3.3 Safety Theorem

Our main result is the following safety theorem for DSA.

Theorem III.2 *Given a MAS indexed by $\mathcal{M}$ with dynamics specified as in Eq. (8), if each agent operates under DSA with the BC as in Eq. (15), the switching logic as in Eqs. (16) and (17), and $\mathbf{x}(0) \in \mathcal{R} \subset \mathbb{R}^{kn}$, then the MAS will remain safe.*

Proof. The proof proceeds by considering both possible DM states for an arbitrary agent i, and establishing that i's next state is safe. First, consider agent i at time t with $DM_i(t) = AC$. As the FSC is false, the one-step reachability check associated with the FSC ensures that the CBFs for unary and binary safety constraints are strictly positive in the next state $x_i(t+\eta)$; i.e. $h_i(x_i(t+\eta)) > 0$ and $\forall j \in \mathcal{N}_i : h_{ij}(x_i(t+\eta), x_j(t+\eta)) > 0$. Hence the next state is recoverable.

Subsequently, consider agent i at time t with $DM_i(t) = BC$. The unary safety constraint is satisfied for agent i as the BC's action is constrained within the admissible control space. Next, we show that the binary safety constraints with all neighboring

agents are also satisfied. We divide the neighbors of i into two sets based on their DM states: the set of neighbors in AC mode and the set of neighbors in BC mode are denoted as $\mathcal{N}_i^{AC}$ and $\mathcal{N}_i^{BC}$, respectively. The neighbors in BC mode choose their control actions from their corresponding admissible control spaces as in Eq. (14). As agent i also chooses its control action from its admissible control space, according to *Theorem* III.1, the neighbors in BC mode will satisfy the binary safety constraints with agent i. As for neighbors in AC mode, due to the one-step reachability check in their FSC, in state $x_i(t+\eta)$, the pairwise CBFs satisfy $h_{ij}(x_i(t+\eta), x_j(t+\eta)) \geq 0$ for all $j \in \mathcal{N}_i^{AC}$. Hence, $x_i(t+\eta)$ is recoverable for $DM_i(t) = BC$. We have proven that for any agent i and time step t, if $x_i(t)$ is recoverable, then $x_i(t+\eta)$ is recoverable. By assumption, $\mathbf{x}(0) \in \mathcal{R}$. Therefore, by induction, $\mathbf{x}(t) \in \mathcal{R}$ for $t > 0$.

4 Flocking Case Study

We evaluate DSA on the distributed flocking problem with the goal of preventing inter-agent collisions. Consider a MAS consisting of k robotic agents with double integrator dynamics, indexed by $\mathcal{M} = \{1, \ldots, k\}$:

$$\begin{bmatrix} \dot{p}_i \\ \dot{v}_i \end{bmatrix} = \begin{bmatrix} 0 & I_{2\times 2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p_i \\ v_i \end{bmatrix} + \begin{bmatrix} 0 \\ I_{2\times 2} \end{bmatrix} a_i \tag{18}$$

where p_i, v_i, $a_i \in \mathbb{R}^2$ are the position, velocity and acceleration of agent $i \in \mathcal{M}$, respectively. The magnitudes of velocities and accelerations are bounded by $\bar{v}$ and $\bar{a}$, respectively. Acceleration a_i is the control input for agent i. As DSA is a discrete-time protocol, the state of the DM and the a_i's are updated every η seconds. The *state* of an agent i is denoted by the vector $s_i = [p_i^T v_i^T]^T$. The *state* of the entire flock at time t is denoted by the vector $\mathbf{s}(t) = [\mathbf{p}(t)^T\ \mathbf{v}(t)^T]^T \in \mathbb{R}^{4n}$, where $\mathbf{p}(t) = [p_1^T(t) \cdots p_n^T(t)]^T$ and $\mathbf{v}(t) = [v_1^T(t) \cdots v_n^T(t)]^T$ are the vectors respectively denoting the positions and velocities of the flock at time t.

We assume that an agent can accurately sense the positions and velocities of nearby agents within a fixed distance r. The set of the *spatial neighbors* of agent i is defined as $\mathcal{N}_i(\mathbf{p}) = \{j \in \mathcal{M} \mid j \neq i \wedge \|p_i - p_j\| < r\}$, where $\|\cdot\|$ denotes the Euclidean norm. For ease of notation, we sometimes use $\mathbf{s}$ and $\mathbf{s}_i$ to refer to the state variables $\mathbf{s}(t)$ and $\mathbf{s}_i(t)$, respectively, without the time index.

The MAS is characterized by a set of operational constraints which include physical limits and safety properties. States that satisfy the operational constraints are called *admissible*, and are denoted by the set $\mathcal{A} \in \mathbb{R}^{4k}$. The desired safety property is that no pair of agents is in a "state of collision". A pair of agents is considered to be in a *state of collision* if the Euclidean distance between them is less than a threshold distance $d_{min} \in \mathbb{R}^+$, resulting in a binary safety constraint of the form: $\|p_i - p_j\| - d_{min} \geq 0\ \forall\ i \in \mathcal{M}, j \in \mathcal{N}_i$. Similarly, a state $\mathbf{s}$ is $recoverable$ if all pairs of agents can brake (de-accelerate) relative to each other without colliding. Otherwise, the state $\mathbf{s}$ is considered *unrecoverable*.

4.1 Synthesis of Control Barrier Function

Let $\mathcal{R}_{ij} \subset \mathbb{R}^8$ be the set of recoverable states for a pair of agents $i, j \in \mathcal{M}$. The flock-wide set of recoverable states, denoted by $\mathcal{R} \subset \mathbb{R}^{4k}$, is defined in terms of $\mathcal{R}_{ij}$. As in [15], the set $\mathcal{R}_{ij}$ is defined as the super-level set of a pairwise CBF $h_{ij} : \mathbb{R}^8 \to \mathbb{R}$: $\mathcal{R}_{ij} = \{s_i, s_j \mid h_{ij}(s_i, s_j) \geq 0\}$. The flock-wide set of recoverable states $\mathcal{R} \subset \mathcal{A}$ is defined as the set of system states in which $(s_i, s_j) \in \mathcal{R}_{ij}$, for every pair of neighboring agents i, j.

In accordance with [15], the function $h_{ij}(s_i, s_j)$ is based on a safety constraint over a pair of agents $i, j \in \mathcal{M}$. The safety constraint ensures that for any pair of agents, the maximum braking force can always keep the agents at a distance greater than d_{min} from each other. As introduced earlier, d_{min} is the threshold distance that defines a collision. Considering that the tangential component of the relative velocity, denoted by Δv, causes a collision, the constraint regulates Δv by application of maximum acceleration to reduce Δv to zero. Hence, the safety constraint can be represented as the following condition on the inter-agent distance $\|\Delta \mathbf{p}_{ij}\| = \|p_i - p_j\|$, the braking distance $(\Delta v)^2/4\bar{a}$, and the safety threshold distance d_{min}:

$$\left\|\Delta \mathbf{p}_{ij}\right\| - \frac{(\Delta v)^2}{4\bar{a}} \geq d_{min} \tag{19}$$

$$h_{ij}(s_i, s_j) = \sqrt{4\bar{a}(\left\|\Delta \mathbf{p}_{ij}\right\| - d_{min})} - \Delta v \geq 0 \tag{20}$$

The braking distance is the distance covered while the relative speed reduces from Δv to zero under a deceleration of $2\bar{a}$. The constraint in Eq. (19) is re-arranged to get the CBF h_{ij} given in Eq. (20).

Combining Eqs. (20) and (12c), we arrive at the linear constraint on the accelerations for agents i and j, which constrains the Lie derivative of the CBF in (20) to be greater than $-\alpha(h_{ij})$. We set $\alpha(h_{ij}) = \gamma h_{ij}^3$, as in [15], where $\gamma \in \mathbb{R}^+$, resulting in the following constraint on the accelerations of agents i, j:

$$\begin{aligned} \frac{\Delta \mathbf{p}_{ij}^T(\Delta \mathbf{a}_{ij})}{\|\Delta \mathbf{p}_{ij}\|} - \frac{(\Delta \mathbf{v}_{ij}^T \Delta \mathbf{p}_{ij})^2}{\|\Delta \mathbf{p}_{ij}\|^3} + \frac{\|\Delta \mathbf{v}_{ij}\|^2}{\|\Delta \mathbf{p}_{ij}\|} \\ + \frac{2\bar{a}\Delta \mathbf{v}_{ij}^T \Delta \mathbf{p}_{ij}}{\|\Delta \mathbf{p}_{ij}\| \sqrt{4\bar{a}(\|\Delta \mathbf{p}_{ij}\| - d_{min})}} \geq -\gamma h_{ij}^3 \end{aligned} \tag{21}$$

where the left-hand side is the Lie derivative of the CBF h_{ij} and $\Delta \mathbf{p}_{ij} = p_i - p_j$, $\Delta \mathbf{v}_{ij} = v_i - v_j$, and $\Delta \mathbf{a}_{ij} = a_i - a_j$ are the vectors representing the relative position, the relative velocity, and the relative acceleration of agents i and j, respectively. We further note that the binary constraint (21) can be reformulated as $\left[\begin{smallmatrix} P_{ij} & Q_{ij} \end{smallmatrix}\right] \left[\begin{smallmatrix} a_i \\ a_j \end{smallmatrix}\right] \leq b_{ij}$, and hence can be split into two unary constraints $P_{ij}u_i \leq b_{ij}/2$ and $Q_{ij}u_j \leq b_{ij}/2$, following the convention in Eq. (13). The set of safe accelerations for an agent i, denoted by $\mathcal{K}_i(\mathbf{s}_i) \subset \mathbb{R}^2$, is defined as the intersection of the half-planes defined by the Lie-derivative-based constraints, where each neighboring agent contributes a single constraint:

$$\mathcal{K}_i(\mathbf{s}_i) = \left\{a_i \in \mathbb{R}^2 \mid P_{ij}u_i \leq b_{ij}/2,\ \forall j \in \mathcal{N}_i\right\} \tag{22}$$

With the CBFs for collision-free flocking defined in (20) and the admissible control space defined in (22), the BC, FSC, and RSC follow from (15), (16), and (17), respectively. We use Taylor approximation of order one to compute FSC and RSC.

4.2 Advanced Controller

We use the Reynolds flocking model [20] as the AC. In the Reynolds model, the accelerations a_i for each agent is a weighted sum of three acceleration terms based on simple rules of interaction with neighboring agents: *separation* (move away from your close-by neighbors), *cohesion* (move towards the centroid of your neighbors), and *alignment* (match your velocity with the average velocity of your neighbors). The acceleration for agent i is $a_i = w_s a_i^s + w_c a_i^c + w_{al} a_i^{al}$, where $w_s, w_c, w_{al} \in \mathbb{R}^+$ are scalar weights and $a_i^s, a_i^c, a_i^{al} \in \mathbb{R}^2$ are the acceleration terms corresponding to separation, cohesion, and alignment, respectively. We note that the Reynolds model does not guarantee collision avoidance. Nevertheless, when the flock stabilizes, the average distance to the closest neighbors is determined by the weights of the interaction terms.

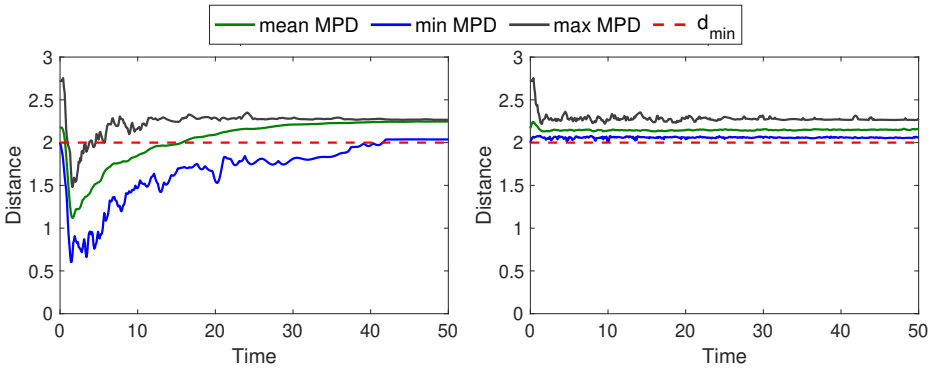
4.3 Experimental Results

The number of agents in the MAS is $k = 15$. The other parameters used in the experiments are $r = 4$, $\bar{a} = 5$, $\bar{v} = 2.5$, $d_{min} = 2$, and $\eta = 0.1$s. The length of the simulations is 50 seconds. The initial positions and the initial velocities are uniformly sampled from $[-10, 10]^2$ and $[-1, 1]^2$, respectively, and we ensure that the initial state is recoverable. The weights of the Reynolds model terms are chosen experimentally to ensure that no pair of agents are in a state of collision in the steady state. They are set to $w_s = 3$, $w_c = 1.5$, and $w_{al} = 0.5$. .

To demonstrate the effectiveness of DSA in preventing inter-agent collisions, we generated 100 simulation runs using two different control strategies, starting from the same set of random initial configurations. In the first set of 100 simulations, Reynolds model is used to control all agents for the duration of the simulations. In the second set of 100 simulations, Reynolds model is wrapped with a verified safe BC and DM designed using DSA.

We define the *minimum pairwise distance* (MPD), as the minimum Euclidean distance between any pairs of agents in the flock, i.e., $\min_{i,j \in \mathcal{M}} \|p_i - p_j\|$. Fig. 2 shows the spread of MPD at each time step, by plotting its mean, minimum, and maximum values, calculated over 100 simulation runs.

As evident from Fig. 2(b), the minimum MPD is greater than d_{min} for the entire duration of the simulation runs, indicating that DSA is able to prevent inter-agents collisions for the 100 simulations. In contrast, as shown in Fig. 2(a), Reynolds model results in safety violations during the first 42 seconds (Only the last 8 seconds are collision-free in all 100 simulations) and the mean MPD crosses the safety threshold at around 16 seconds. Moreover, operating under DSA, the distribution of MPD is relatively uniform over the duration of the simulations. We further note that the average time the agents spend in BC mode is only 3.44 percent of the total duration of the simulation, indicating



(a) Reynolds Model (b) Reynolds Model with DSA

Fig. 2: The minimum pairwise distance (MPD) for a flock of size 15, calculated over 100 simulation runs, with and without DSA.

that DSA is largely non-invasive. Videos of flocking under both control strategies are available online.[3]

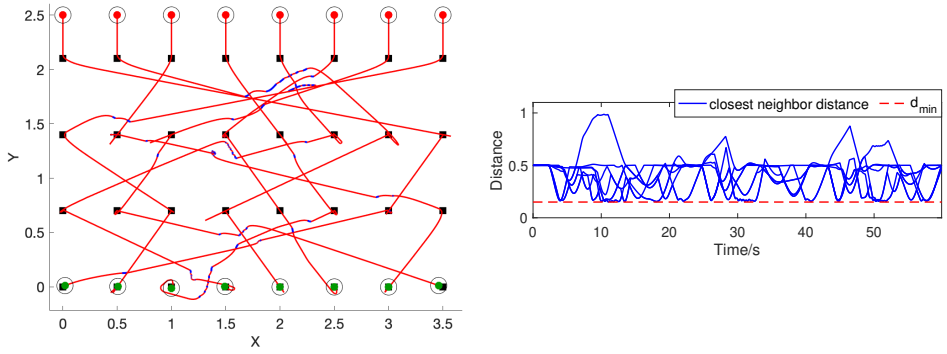
The simulation results clearly demonstrate the effectiveness of DSA in guaranteeing inter-agent collision avoidance. We also ran simulations where all agents are solely under the control of the BC. As the BC is strictly focused on safety, we observed that the flock fragments as agents safely move out of the sensing ranges of other agents.

The flocking case study clearly illustrate the guiding principles and benefits of DSA. In particular, it shows that: (a) the AC is not always safe, but (b) the combination of the AC and BC in DSA is safe, and (c) DSA outperforms the BC.

5 Way-point Case Study

This section describes the problem setup and experimental results for the way-point (WP) control case study. The agent model is the same as the one used for the flocking case study, given in Eq. (18). The experimental setup is shown in Fig. 3, where the agents, initially positioned along the top of the figure, are to navigate through a series of WPs while maintaining a safe distance from one another. The WPs are represented by the black squares. The CBFs, BC and DM are same as those defined for the flocking problem; see Section 4

The AC is a rule-based controller where each agent accelerates towards its next WP (ignoring the other agents) until the final WP is reached. Agents are assigned one WP from each row such that they are on a collision course if they follow the AC's commands.



(a) Trajectories of agents passing through WPs. Red/blue segments indicate AC/BC mode.

(b) Distance to closest neighbor for all agents.

Fig. 3: Experimental results for way-point case study.

5.1 Experimental Results

The number of agents used in the experiment is eight and the the number of WPs an agent is required to visit is four (one in each row). Initially, the agents are at rest with their positions represented by the red dots in Fig. 3(a). The final positions are shown as green dots. The duration of the simulation is 60 seconds. The other parameters used in the experiments are $r = 1.0$, $\bar{a} = 0.8$, $\bar{v} = 0.2$, $d_{min} = 0.15$, and $\eta = 0.05$s. The trajectories of the agents are given in Fig. 3(a), where the segments in blue indicate when the BC is in control. Fig. 3(b) plots the smallest inter-agent distances, indicating that the agents maintain a safe distance from one another. A video of the simulation is available online.[4]

6 Microgrid Case Study

With an increasing prevalence of distributed energy resources (DERs) such as wind and solar power, electrification using microgrids (MGs) has witnessed unprecedented growth. Unlike traditional power systems, MG DERs do not have rotating components such as turbines. The lack of rotating components can lead to low inertia, making MGs susceptible to oscillations resulting from transient disturbances [21]. Ensuring the safe operation of an MG is thus a challenging problem. In this case study, we demonstrate the effectiveness of DSA in maintaining MG voltage levels within safe limits.

The MG we consider is a network of k droop-controlled inverters, indexed by $\mathcal{M} = \{1, \ldots, k\}$. The dynamics of each inverter is modeled as [21, 22, 23, 24]:

$$\dot{\theta}_i = \omega_i \tag{23a}$$

[3] https://youtu.be/E_ufaJRnfvo, https://youtu.be/PZz6nUA5fD8

[4] https://youtu.be/AcC8iUI0TjU

$$\tau_i \dot{\omega_i} = \omega_i^0 - \omega_i + \lambda_i^p (P_i^{set} - P_i) \tag{23b}$$

$$\tau_i \dot{v_i} = v_i^0 - v_i + \lambda_i^q (Q_i^{set} - Q_i) \tag{23c}$$

where θ_i, ω_i, and v_i are respectively the phase angle, frequency, and voltage of inverter i, $i \in \mathcal{M}$. The state vector for the MG is denoted by $\mathbf{s} = [\theta^T \ \omega^T \ v^T]^T \in \mathbb{R}^{3k}$, where θ, ω, and v are respectively vectors representing the voltage phase angle, frequency, and voltage at each node of the MG. A pair of inverters are considered neighbors if they are connected by a transmission line. Also, λ_i^p and λ_i^q are droop coefficients of "active power vs frequency" and "reactive power vs voltage" droop controllers, respectively. $\tau_i \in \mathbb{R}^+$ is the time constant used for the low-pass filters that are processing the active and reactive power measurements. Finally, ω_i^0 and v_i^0 are the nominal frequency and voltage values.

P_i and Q_i are the active and reactive powers injected by inverter i into the system:

$$\begin{aligned} P_i &= v_i \sum_{j \in \mathcal{N}_i} v_j (G_{i,j} \cos\theta_{i,j} + B_{i,j} \sin\theta_{i,j}) \\ Q_i &= v_i \sum_{j \in \mathcal{N}_i} v_j (G_{i,j} \sin\theta_{i,j} - B_{i,j} \cos\theta_{i,j}) \end{aligned} \tag{24}$$

where $\theta_{i,j} = \theta_i - \theta_j$, and $\mathcal{N}_i \subseteq \mathcal{M}$ is the set of neighbors. $G_{i,j}, B_{i,j}$ are respectively conductance and susceptance values of the transmission line connecting inverters i and j.

P_i^{set} and Q_i^{set} are the active power and reactive power setpoints. The inverters have the ability to change their respective power setpoints according to the MG's operating conditions. This is modeled as:

$$P_i^{set} = P_i^0 + u_i^p, \ Q_i^{set} = Q_i^0 + u_i^q \tag{25}$$

where P_i^0 and Q_i^0 are the setpoints for the nominal operating condition, and u_i^p and u_i^q are control inputs.

6.1 Synthesis of Control Barrier Function

The safety property for the MG network is a set of unary constraints restricting the voltages at each node to remain within safe limits. The recoverable set $\mathcal{R}_i \subset \mathbb{R}^3$ for inverter i is defined as the super-level set of a CBF $h_i : \mathbb{R}^3 \to \mathbb{R}$. We follow the SOS-optimization technique given in [24] to synthesize the CBFs. Since the power flow equations (23) are nonlinear, we apply a third-order Taylor series expansion to approximate the dynamics in polynomial form. We then follow the three-step process given in [24] to obtain the CBF for each MG node. We then calculate the admissible control space according to Eq. (14), and the BC, FSC, and RSC follow from Eqs. (15), (16), and (17), respectively. We have experimented with various orders of Taylor approximations for the computation of FSC and RSC.

6.2 Advanced Controller

The AC sets the active/reactive power setpoints to their nominal values. Thus, the AC does not limit voltage and frequency magnitudes but is only concerned with stabilizing frequency and voltage magnitudes to their nominal values.

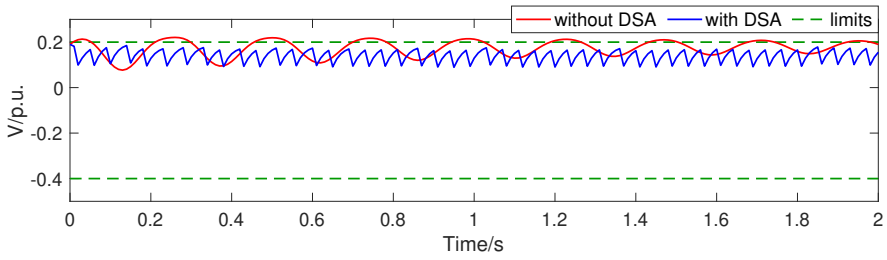


Fig. 4: Voltage graph at node 4 of the MG network.

6.3 Experimental Results

We consider a 6-bus MG [24]. Disconnecting the MG from the main utility, we replace bus 0 with a droop-controlled inverter (Eq. (23)), with inverters also placed on buses 1, 4 and 5. Bus 0 is the reference bus for the phase angle. Nominal values of voltage and frequency, as well as the active/reactive power set-points, were obtained by solving the steady-state power-flow equations given in Eq. (24); these were then used to shift the equilibrium point to the origin. Droop coefficients λ_i^p and λ_i^q were set to 2.43 rad/s/p.u. and 0.20 p.u./p.u., τ_i was set to 0.5s, and the control period η was set to 0.01s. Loads are modeled as constant power loads, and a Kron-reduced network [25] with only the inverter nodes was used for analysis. The safe set is defined in terms of the shifted (around the 0 p.u.) nodal voltage magnitudes as follows: $v_i \geq -0.4\,\text{p.u.} \wedge v_i \leq 0.2\,\text{p.u.}$

The duration of the simulation is two seconds. Our results show that with DSA, the voltage at each node remains within safe limits; without DSA, safety limits are exceeded. Fig. 4 gives the voltage plot at node 4. When the MG is operating under the control of DSA and the voltage approaches the upper limit, a switch from AC to BC occurs. Subsequently, the BC reduces the voltage inducing a reverse switch. The voltage profiles at the other nodes are similar.

7 Related Work

The original Simplex architecture [5, 26] was developed for systems comprising a single controller and a single (non-distributed) plant. With DSA, we extend the scope of Simplex to MASs under distributed control. RTA [27, 28] is a runtime assurance technique that can be applied to component-based systems. In this case, however, each RTA wrapper (i.e., each Simplex-like instance) independently ensures a local safety property of a component. For example, in [27], RTA instances for an inner-loop controller and a guidance system are uncoordinated and operate independently. In contrast, in DSA, each agent takes the states of neighboring agents into account when making control decisions, in order to ensure that pairwise safety constraints are satisfied.

A runtime verification framework for dynamically adaptive multi-agent systems (DAMS-RV) is proposed in [29]. DAMS-RV is activated every time the system adapts to a change in the system itself or its environment. This method relies on a *monitoring* phase to observe and identify changes that occur in agent collaboration so that verification can be carried out on the system operating in new contexts. In contrast, DSA does

not require such intermediary supervision. In [30], a dynamic policy model that can be used to express constraints on agent behavior is presented. These constraints limit agent autonomy to lie within well-defined boundaries. Constraint specifications are kept simple by allowing the policy designer to decompose a specification into components and define the overall policy as a composition of these smaller units. In contrast, DSA uses CBFs to compute the requisite safety regions.

In [10, 14, 15, 31], CBF-based methodologies have been used for runtime safety assurance of MASs. For example, in [10, 15], a formal framework for collision avoidance in multi-robot systems is presented. A CBF-based wrapper around an advanced controller guarantees forward invariance of a safe set. The wrapper solves an optimization problem involving the Lie derivative of the CBF to compute minimal changes to the AC's output needed to ensure safety. In contrast, in DSA, no attempt is made to minimally perturb the AC's output. Instead we rely on CBF-based switching logic in the DM to forward switch to the BC if the AC's output is not recoverable.

In [32], a shield-based technique for runtime verification of multi-agent systems is presented. In this approach, which does not require global information, every agent has a shield consisting of two components: a pathfinder that corrects the behavior of the agent, and an ordering mechanism that dynamically modifies the priority of the agent. An upper bound is derived on the maximum deviation for any agent from its original behavior. In contrast, DSA relies on forward and reverse switching between an agent's advanced and baseline controllers to safely allow completion of mission goals.

8 Conclusion

We have presented the Distributed Simplex Architecture, a runtime assurance technique for the safety of multi-agent systems. DSA is distributed in the sense that it involves one local instance of traditional Simplex per agent such that the conjunction of their respective safety properties yields the desired safety property for the entire MAS. Moreover, an agent's switching logic depends only on its own state and that of neighboring agents. We demonstrated the effectiveness of DSA by successfully applying it to flocking, way-point visiting, and microgrid control. As future work, we plan to apply DSA to non-homogenous MASs and implement it on a physical platform.

In some situations, the BC's optimization problem might become infeasible. For example, for the flocking case study, infeasibility of the BC's optimization problem is possible if the agents are crowded in a small region. One possible solution for infeasibility is to design aggressive control barrier functions that guarantee feasibility at the cost of performance. For the flocking case study, one such solution is a CBF that enforces the braking manoeuvre [10].

Bibliography

[1] M. Nasir, Z. Jin, H. A. Khan, N. A. Zaffar, J. C. Vasquez, and J. M. Guerrero, "A decentralized control architecture applied to DC nanogrid clusters for rural electrification in developing regions," *IEEE Transactions on Power Electronics*, vol. 34, no. 2, pp. 1773–1785, 2019.

[2] Z. Boussaada, O. Curea, H. Camblong, N. Bellaaj Mrabet, and A. Hacala, "Multi-agent systems for the dependability and safety of microgrids," *International Journal on Interactive Design and Manufacturing*, 2016.

[3] A. Tahir, J. Böling, M.-H. Haghbayan, H. T. Toivonen, and J. Plosila, "Swarms of unmanned aerial vehicles — a survey," *Journal of Industrial Information Integration*, vol. 16, p. 100106, 2019.

[4] R. Tynan, G. M. P. O'Hare, D. Marsh, and D. O'Kane, "Multi-agent system architectures for wireless sensor networks," in *Computational Science – ICCS 2005*. Springer, 2005, pp. 687–694.

[5] D. Seto and L. Sha, "A case study on analytical analysis of the inverted pendulum real-time control system," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU/SEI-99-TR-023, 1999.

[6] L. Sha, "Using simplicity to control complexity," *IEEE Software*, vol. 18, no. 4, pp. 20–28, 2001.

[7] D. Phan, R. Grosu, N. Jansen, N. Paoletti, S. A. Smolka, and S. D. Stoller, "Neural simplex architecture," in *Proceedings of NASA Formal Methods Symposium (NFM 2020)*, 2020.

[8] T. Gurriet, A. Singletary, J. Reher, L. Ciarletta, E. Feron, and A. Ames, "Towards a framework for realizable safety critical control through active set invariance," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, 2018, pp. 98–106.

[9] M. Egerstedt, J. N. Pauli, G. Notomista, and S. Hutchinson, "Robot ecology: Constraint-based control design for long duration autonomy," *Annual Reviews in Control*, vol. 46, pp. 1 – 7, 2018.

[10] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for heterogeneous multi-robot systems," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 5213–5218.

[11] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Hybrid Systems: Computation and Control, 7th International Workshop*, ser. Lecture Notes in Computer Science, R. Alur and G. J. Pappas, Eds., vol. 2993. Springer, 2004, pp. 477–492.

[12] S. Prajna, "Barrier certificates for nonlinear model validation," *Autom.*, vol. 42, no. 1, pp. 117–126, 2006.

[13] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 462 – 467, 2007, 7th IFAC Symposium on Nonlinear Control Systems.

[14] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *18th European Control Conference, ECC 2019, Naples, Italy*. IEEE, 2019, pp. 3420–3431.
[15] U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, "Control barrier certificates for safe swarm behavior," in *ADHS*, ser. IFAC-PapersOnLine, M. Egerstedt and Y. Wardi, Eds., vol. 48, no. 27. Elsevier, 2015, pp. 68–73.
[16] F. Blanchini and S. Miani, *Set-Theoretic Methods in Control*, 1st ed. Birkhäuser Basel, 2007.
[17] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747 – 1767, 1999.
[18] L. Wang, D. Han, and M. Egerstedt, "Permissive barrier certificates for safe stabilization using sum-of-squares," in *2018 Annual American Control Conference, ACC 2018*. IEEE, 2018, pp. 585–590.
[19] U. Mehmood, N. Paoletti, D. Phan, R. Grosu, S. Lin, S. D. Stoller, A. Tiwari, J. Yang, and S. A. Smolka, "Declarative vs rule-based control for flocking dynamics," in *Proceedings of 33rd Annual ACM Symposium on Applied Computing*, 2018.
[20] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 25–34, Aug. 1987.
[21] N. Pogaku, M. Prodanovic, and T. C. Green, "Modeling, analysis and testing of autonomous operation of an inverter-based microgrid," *IEEE Transactions on Power Electronics*, vol. 22, no. 2, pp. 613–625, 2007.
[22] J. Schiffer, R. Ortega, A. Astolfi, J. Raisch, and T. Sezi, "Conditions for stability of droop-controlled inverter-based microgrids," *Automatica*, vol. 50, no. 10, pp. 2457 – 2469, 2014.
[23] E. A. A. Coelho, P. C. Cortizo, and P. F. D. Garcia, "Small-signal stability for parallel-connected inverters in stand-alone AC supply systems," *IEEE Transactions on Industry Applications*, vol. 38, no. 2, pp. 533–542, 2002.
[24] S. Kundu, S. Geng, S. P. Nandanoori, I. A. Hiskens, and K. Kalsi, "Distributed barrier certificates for safe operation of inverter-based microgrids," in *2019 American Control Conference (ACC)*, 2019, pp. 1042–1047.
[25] P. Kundur, N. Balu, and M. Lauby, *Power System Stability and Control*, ser. EPRI power system engineering series. McGraw-Hill Education, 1994.
[26] D. Seto, B. Krogh, L. Sha, and A. Chutinan, "The Simplex architecture for safe online control system upgrades," in *Proceedings of the 1998 American Control Conference*, vol. 6, 1998, pp. 3504–3508.
[27] M. Aiello, J. Berryman, J. Grohs, and J. Schierman, *Run-Time Assurance for Advanced Flight-Critical Control Systems*, 2010.
[28] J. Schierman, D. Ward, B. Dutoi, A. Aiello, J. Berryman, M. DeVore, W. Storm, and J. Wadley, *Run-Time Verification and Validation for Safety-Critical Flight Control Systems*, 2012.
[29] Y. J. Lim, G. Hong, D. Shin, E. Jee, and D.-H. Bae, "A runtime verification framework for dynamically adaptive multi-agent systems," in *2016 International Conference on Big Data and Smart Computing (BigComp)*, 2016, pp. 509–512.
[30] H. Alotaibi and H. Zedan, "Runtime verification of safety properties in multi-agents systems," in *2010 10th International Conference on Intelligent Systems Design and Applications*, 2010, pp. 356–362.

[31] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[32] D. Raju, S. Bharadwaj, and U. Topcu, "Online synthesis for runtime enforcement of safety in multi-agent systems," *preprint ArXiv:1910.10380*, 2019.