Diffusion Based Swarm Dynamic for Decentralized Maze Solving

Ms. Arianne Ritz Master Student ETHZ

***Abstract*— The general structure of multi-agent control systems as well as model predictive control is discussed. The goal is to utilize a heuristic based on minimal information of other agents in order to diverge in a graph like maze until the goal node is found, where the heuristic is used to converge all agents on that node. The methods includes simplifying the unknown and obstructed space to a simple series of convex spaces that are easily navigable and solving the graph using to find nodes that are suitable targets. Other than most decentralized systems, the one described here does not have agents who's states are influenced directly by other agent's states. There exists a demonstration of this paper on GitHub.**

I. INTRODUCTION

It is arguably easier and more efficient to control a multi-agent system from one centralized location. An argument is that the centralized controller has access to more data than an individual agent and can hence the globally best known decisions. Centralized control requires agents to be able to communicate with the controller, where sending large amounts of sensor data can be slow. In such applications, decentralized control can be better, where each agent controls itself but can rely on minimal information from other agents to make more informed decisions than it could on it's own. Here, the example of decentralized control will be multiple agents solving a maze with minimal communication between each other.

Existing approaches to decentralized multi-agent control often rely on prior knowledge of the goal, structured communication models, or predefined coordination behaviors. For example, MC-Swarm by Lee and Park [4] assumes goal positions are known in advance and focuses on collision-free trajectory planning without communication after the planning stage. Stigmergy-based systems, such as those proposed by Li et al. [2], coordinate through virtual pheromone fields around fixed communication nodes. Other methods, like the work of Horyna et al. [3], use rule-based flocking and implicit visual signaling to achieve collective search and convergence.

In contrast, the system presented here addresses a setting where neither the goal location nor the maze is known at the outset, agents cannot rely on environmental markers such as communication nodes, and coordination emerges from heuristic decisions based on sparse qualitative information exchanged between agents. The idea is that each agent's decisions are influenced to move away from agents that are performing worse and favor directional choices that move towards agents that are more successful than themselves. Since the goal of the maze is not known initially, this should lead to a diffusion effect such that agents spread out as much as possible and later, upon goal discovery by at least one agent, congregate on the goal. This means that agents collectively are expected to explore the entirety of the maze faster in a coordinated manner without actually coordinating via a central controller.

II. PROBLEM DESCRIPTION

A. Multi-Agent System

The system is a collection of multiple autonomous vehicles, termed agents. As described, these agents do not receive their commands from the same centralized controller but rather use a controller within them to come up with their own, uncoordinated target location and input trajectories. For this, the agents will come up with reference points to approach and use Model Predictive Control MPC to optimize input trajectories according to a cost function and re-plan after each input in Receding Horizon Control (RHC). Agents are simplified to points in two dimensional space that can apply a force with a maximal amplitude as an input. Here agents are allowed to inhabit the same exact point space. In reality this could be replicable by having a set height for each agent.

B. Environment

The environment is that of an unknown space with a relatively high amount of obstruction and an unknown goal location. Specifically, the environment is fully connected; there exists an accessible and bidirectional path between any two free spaces in the environment. The environment will practically have hallway or maze-like properties. This description purposefully does not exclude any topological features like dead ends and loops.

Generally this is represented in a continuous two-dimensional domain. For simplicity's sake, there also exists a simplified two-dimensional graph representation of the environment. In reality, an agent could come up with the graph representation during exploration. This could be achieved by declaring a node the position of each time step and finding the connectivity of nodes based on feature extraction of visual data. This would also require keeping track of unvisited spaces by checking which part of the vision data is both free space and near the maximum range of vision.

C. Agent Sensing

At each time step, agents receive "visual" information about their environment. For simplicity reasons, this will be a given area centered around the agents, giving information if there is a node of the graph representation in vision. With the node, agents also get information about neighboring nodes.

D. Inter Agent Communication

The communication between agents here is the intended limiting factor. For this intent, agents will broadcast, at each time step, their current expected performance and their absolute position, in essence, three floating-point numbers. In reality, this could be achievable via sound. Sound could actually carry additional information about the absolute travel distance to the agent that is broadcasting their information, which could be used to extract the fastest way to reach said agent by following the intensity of the sound, in the right environment.

Especially, every agent should work as intended, no matter how much communication from other agents it gets; such that each agent is capable of reaching the goal on it's own and only gets heuristic aid in decision making based on other agents. This restriction is chosen to make the system as robust to communication issues as possible.

This means that the stability of each agent need to be guaranteed for all numbers of agents and all inputs derived from their states. Here it is chosen to completely decouple the influence of other agents to the input of the system and rather have a decentralized system dynamic where agents influence the targeting of other agents. This means that most classical decentralized control theorems are not applicable to the situation as they focus on system stability where there is a direct influence on each agent.

III. Control Algorithm

A. Discretized System Dynamics

Given the equation of motion (EoM) that describes an agents dynamics as a second order differential equation with a force vector as input u and generalized coordinates q:

$$M\ddot{q} + D\dot{q} + Kq = F$$

Since the problem is abstracted to two dimensions, and agents will be represented as points, agents do not have an orientation. We can derive a linear time-independent (LTI) state description where, since we care about the position of the system, we will extract it from the state x_i as an output y_i, such that:

$$\dot{x} = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \ddot{x_1} \\ \ddot{x_2} \end{bmatrix} = \begin{bmatrix} \mathbb{0}^{2\times2} & \mathbb{I}_2 \\ -M^{-1}K & -M^{-1}D \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dot{x_1} \\ \dot{x_2} \end{bmatrix} + \begin{bmatrix} \mathbb{0}^{2\times2} \\ M^{-1} \end{bmatrix} F \tag{1}$$

$$:= A_c x + B_c u \tag{2}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \mathbb{I}_2 & \mathbb{0}^{2\times2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dot{x_1} \\ \dot{x_2} \end{bmatrix} := C_c x \tag{3}$$

For $M, D, K \in \mathbb{R}^{2\times2}$, $A \in \mathbb{R}^{4\times4}$, $B \in \mathbb{R}^{4\times2}$, $C \in \mathbb{R}^{2\times4}$,d $y, u = F \in \mathbb{R}^2$ and $x \in \mathbb{R}^4$. This system is in continuous time. Since we intend to control agents using a discrete system, the agent dynamics should be discretized exactly[6] such that:

$$\begin{bmatrix} A & B \\ \mathbb{0}^{4\times2} & \mathbb{0}^{2\times2} \end{bmatrix} = exp(\begin{bmatrix} A_c & B_c \\ \mathbb{0}^{4\times2} & \mathbb{0}^{2\times2} \end{bmatrix} \cdot T_s) \tag{4}$$

Where a matrix exponential is guaranteed to converge for any real matrix[7]. Here the sampling time T_s is also used for a zero order hold on the variables x, u, y such that $x_k = x(k \cdot T_s)$ and analogous for u, y. Since the output y is still the same as $\begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$ it follows that the continuous and discrete matrix C are the same. This gives the discretized agent dynamics:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned} \tag{5}$$

The assumption of agents as points also imply that the mass matrix M is a scaled identity matrix and K is a zero matrix.

B. Reference Tracking

Each agent will find points to approach. These reference points are such that the output of the agent dynamics is supposed asymptotically approach them. This is called a steady state x_S. Hence we require that for reference $r = y_S = Cx_S$ as $k \to \infty$ that the agent does not deviate from the steady state $x_S = Ax_S + Bu_S$ which gives a system of equations such that the steady state is governed by:

$$\begin{bmatrix} \mathbb{I}_4 - A & -B \\ C & \mathbb{0}^{2\times2} \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} \mathbb{0}^{2\times1} \\ r \end{bmatrix} \tag{6}$$

Which has a stable solution given by $x_S = \begin{bmatrix} r_1 & r_2 & 0 & 0 \end{bmatrix}^T$ and $u_S = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$ given that $r = \begin{bmatrix} r_1 & r_2 \end{bmatrix}^T$.

C. Model Predictive Control

Agents will use classic open-loop Model Predictive Control (MPC) to optimize over the best input trajectory to reach a desired end state. MPC is especially useful in this application because MPC allows for constrained optimization, which in this case is necessary to avoid obstacles and stay within force actuation limits. Each agent will solve it's own MPC Problem since the problem at hand is decentralized control and then apply the first of the optimal inputs and re-plan for the next time step. This process is called Receding Horizon Control (RHC). MPC with a finite horizon N will optimize its trajectories over the next N time steps according to a cost function $J(x_0, U)$, where state x and input u are constrained to their respective convex set $\mathcal{X}$,$\mathcal{U}$. The cost function is dependent on $x_0 = x(k \cdot T_s)$ and $U = \begin{bmatrix} u_0 & ... & u_{N-1} \end{bmatrix}^T$ since $x_1, ..., x_N$ are deterministic given the agent dynamics (5). Here, optimal variables are denoted with a top script asterisk " * ".

For this implementation, the cost function is a weighted square norm with a final cost I_f and stage cost I. These are suitable candidates as they are all convex, which allow for minimization. Additionally $\mathcal{X}$,$\mathcal{U}$ are also required to be convex, which is true for $\mathcal{U}$ as it is a norm ball where the norm is restricted. The restrictions to set $\mathcal{X}$ are that it is a

$$
\begin{aligned}
U_k^* &= \arg\min_{U} \quad J(x_0, U) \\
\text{s.t.} \quad J(x_0, U) &= I_f(x_N) + \sum_{i=0}^{N-1} I(x_i, u_i) \\
I_f(x_N) &= ||x_N - x_S||_P^2 \\
I(x_i, u_i) &= ||x_i - x_S||_Q^2 + ||u_i - u_S||_R^2 \\
P, Q, R &\succ 0 \\
x_{i+1} &= Ax_i + Bu_i \\
x_0 &= x(k \cdot T_s) \\
x_i, x_N &\in \mathcal{X} \\
u_i &\in \mathcal{U} \\
u_S &= \mathbb{0}^2 \\
\forall\ i &\in \begin{bmatrix} 0 & \dots & N-1 \end{bmatrix}
\end{aligned}
$$

Fig. 1: Open Loop model predictive control to find input sequence U that minimizes a weighted square norm of the error form the next N states to a reference steady state, while constraining states and inputs to a convex set. The weights of the cost fiction are similar to that of a finite horizon linear quadratic regulator

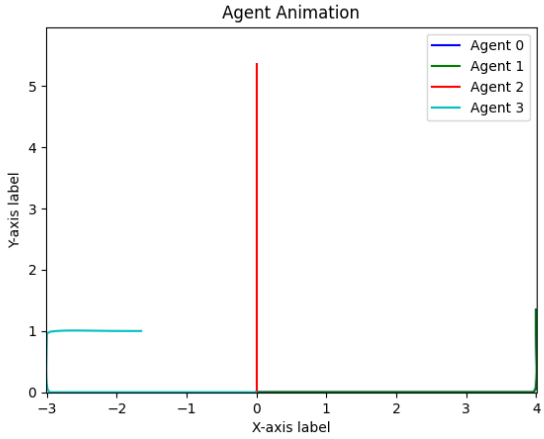
maximal absolute distance from a line. The line in question is the direct path between two given nodes in the graph, representing the hallway, where the distance is the hallway width. This creates a convex normed vector space, visually equivalent to a "capsule" or "pill" shape.

Usually MPC specifies that the last state x_N be within some terminal set $\mathcal{X}_T$. Usually this is done such that the terminal set has a guaranteed stable solution for any state in the set. Restricting the final state in that set therefore guarantees a feasible solution. Here this is not necessary as discussed in section VI. Instead here the reference or steady state is chosen to be in some set called the target set or target domain $\mathcal{X}_f$.

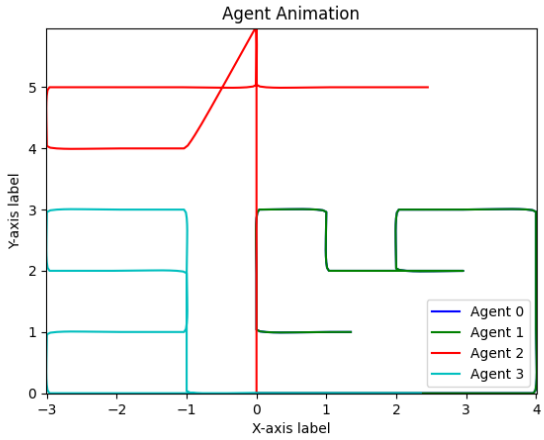
The score or performance of an agent at each time step is the expected cost in N steps and hence the result of the MPC problem. Since all matrices in the cost function are at least positive semi definite, this means that the cost function $J(x_o, U) >= 0$ and hence the performance score is non-negative.

D. Target Domain Selection

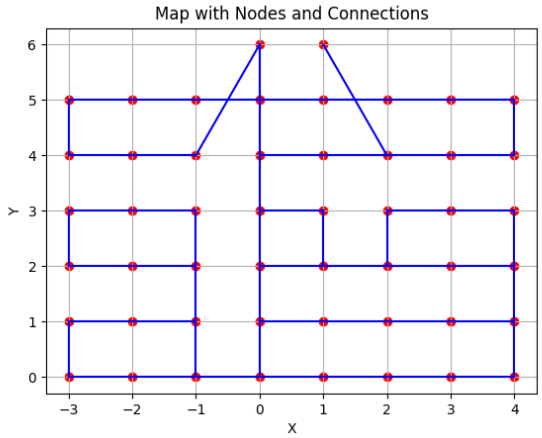
Choosing appropriate reference points to approach for the agents in this abstracted graph environment has now become a graph search problem. Here, the agent physically travels to the best choice of nodes such that it discovers more and more nodes until the goal node is found. The synopsis of graph search appropriate for this section is that a common and optimal choice is the so called A star or A* algorithm. Where nodes are explored first if the sum of distance to the starting node and the heuristic aid to determine approximate distance to the starting node is smaller than all other nodes. This is essentially informed graph search.



(a) Initial divergence of agents. Approximately 60 time steps or 6 seconds into the simulation.



(b) Graph mostly searched. Approximately 235 time steps or 23.5 seconds into the simulation.



(c) Full graph used for demonstration containing 50 Nodes

Fig. 2: Agent trajectories in a simulated maze environment using the proposed targeting heuristic. In (a), agents initially spread out such that they reach the border of the graph instead of targeting closer nodes. In (b), agents have searched most of the map with a clear differentiation between searched areas of each agent. In both (a) and (b) only 3 of 4 agent paths are visible due to agent 0 and agent 1 sticking together. In (c) the whole graph is shown for reference.

To determine reference points r or reference steady states x_S for each agent, A star is used to find the optimal path in optimal time from that agents current node to a target node n_t. The reference points are the sequentially next nodes to approach for the agent on the path to the target node n_t.

The choice of target nodes n_t is where the communication of other agents factor in. Essentially we choose from all seen, but not yet visited nodes, based on their respective distance to the current node, like in A star, and based on how well other agents are doing that are close to that node. Since it is the goal to stay at the goal node, this is only done if an agent is not yet at the goal location. We choose:

$$\text{Targeting heuristic } h_t(\text{candidate node}) = \tag{7}$$

$$\alpha * \text{graph distance to candidate node} + \tag{8}$$

$$\beta * \sum_{i \neq 0}^{n} \frac{score(i)}{score(0) * (1 + L)} \tag{9}$$

Where score(i) are the scores of all agents not performing the heuristic and score(0) is the score of that agent. L denotes the distance found with the 2 norm of agent 0's current node and the candidate node. The co factors $\alpha, \beta \in \mathbb{R}$ are there to regulate how much each term influences the heuristic. Candidates are chosen according to which candidate has the smallest heuristic.

The first term (8) favors nodes that are easy to reach. The second term (9) is weighted with the performance of each agent i relative to agent 0. Here, a distinction has to be made for each agent i with respect to agent 0:

- If the scores have the same sign, the second term favors nodes that are further away from agents than others,
- Else, if scores have opposite signs, the second term favors nodes that are close to the agent with the negative score, which is the agent that does run this evaluation.

This means that this second component of the heuristic (9) can control the choice of target nodes n_t such that agents either favor nodes further from other agents, leading to a diffusion, or favor nodes that are closer to the goal node.

The attraction to nodes closer to agents with negative scores is due to the subtraction from the heuristic and smaller heuristic meaning better expectation.

Here, it is important to repeat that scores derived from the MPC cost function are nonnegative, meaning that all agents generally have divergent effects from each other. This is unless the score is artificially altered if an agent reaches the goal node, which must be done for the described effect. The score of agents at the goal node should be sufficiently small or any negative number if the target node selection function returns evaluates the heuristic only for agents that have negative scores, given that there are any, with $\alpha = 0$ if there are negative scores for optimality. Noteworthy is that if the score is $-\infty$ then the heuristic doesn't work because the distance to the candidate node is now irrelevant.

IV. Main Result

By simplifying the unknown and nonconvex environment to a graph, we can abstract the environment to a union of convex subspaces. These convex subspaces, representing hallways of a certain width between two nodes, are easily navigable on their own by use of model predictive control. MPC is used to navigate from the current node to the next node on a path to the eventual target node n_t. The target node is chosen to be the in the set of unvisited nodes, guaranteeing that each agent individually visits all nodes of the graph in finite time. Guarantee that the goal node is found by all agents in finite time. If the current node is the goal node, the target node is instead chosen to be the current node, which means that agents stay at the goal node. This implies guaranteed convergence of all agents at the goal node, no matter the state of other agents.

By broadcasting each agent performance and position, the multi-agent system is able to make informed decisions despite not actively coordinating with other agents. This is achieved by choosing target nodes to be further away from agents, who's performance has the same sign, leading to a diffusion effect that makes the system discover more nodes overall, leading to the goal node being found faster than without that communication. By altering the score of agents at the goal node to be negative, the same terms act in order to attract agents in the maze to be attracted to the goal node after it's discovery, leading to a faster convergence of control algorithm than without inter agent communication. Additionally, a term favoring closer nodes is added to the evaluation of optimal target nodes in order to reduce the redundant travel of agents. The best trade-off between favoring nodes close to the current node of an agent and favoring nodes far agents that are not at the goal is not found.

This result is also found in a demonstration made for this paper [5]. Key takeaways of the demonstration are found in section V. Implementation details are found in the code. Data was generated using `python main.py -n 4 -m c_maze`, which is to say, with the default values specified in the repository.

V. Discussion

One key takeaway from the demonstration, visible in figure 2, is that agents with the same initial target stay together throughout the whole simulation. This is explained by the fact that the heuristic (7) is determinant and equivalent for agents that are at the same node and are evaluated at the same time. By introducing some disturbance on the state, agents should eventually decouple from each other, as some agents will reach nodes earlier than others, and hence run the heuristic evaluation of target nodes at different times. For the demonstration, this limits an upper bound of agents used equal to the number of initial choices to be the maximum of efficiency.

The demonstration shows a diffusion effect for the selected scenario in figure 2. Agents consistently choose nodes that spread the agents apart, until the border is reached. Then they

continue to search their own subset of the graph. This gives the impression of a coordinated search, even though that is not the case in the classical centralized control sense. This is not a sufficient proof of the method, but a good indicator that it is working as hypothesized.

Convergence of agents at goal node was also as expected in demonstration, but not visualized.

It is unfortunate that the optimal tradeoff for the heuristic (7) is not found, but at $\alpha = 1.0, \beta = 1.0$ the effects of both picking close targets and favoring nodes further from other agents are both noticeable in comparison to either coefficient being zero.

VI. Stability Guarantees

In multi-agent problems, it is never sufficient to prove the stability of a singular agent to make conclusions about system stability. This is because since the stability of each agent depends not just on their own states but also on other agents', there might exist scenarios in which agents align to make some agents' controller unstable. For this reason, we will find sufficient and necessary conditions that guarantee the stability of the system for the given scenario.

A. Problem Termination

First, conditions such that the problem can terminate must be explored. In a sense, it is necessary to know whether or not there can exist at least one solution. A solution in this case is a state trajectory for each agent that at no time violates the constraints $x_k \in \mathcal{X}, u_k \in \mathcal{U}$ and progresses according to the agent dynamics (5) until all agents successfully reach the goal position $\mathcal{X}_{Goal}$ for given initial states $x(0)$. Given the assumptions,

- Each agent starts at rest $\dot{x}(0) = \mathbb{0}^4 \in \mathcal{X}$. This is necessary because otherwise an agent might start in a state that is guaranteed to violate constraints.
- The constraints $\mathcal{X}$,$\mathcal{U}$ allow agents to move in all 4 cardinal directions.
- The environment is fully connected in the sense that there exists an unobstructed bidirectional path between any two free spaces
- There exists a bidirectional fully connected graph that correctly represents the environment
- There exists exactly one node in the environment graph that represents the destination $\mathcal{X}_{Goal}$
- All agents can inhabit the same node at the same time

Then for each agent n we can define the points in time $T_j^n \in \{t_k | t_k = k \cdot T_s \quad \forall k \in \begin{bmatrix}0 & 1 & ... & N_{total}\end{bmatrix}\}$, such that $T_{j+1}^n > T_j^n$. These points in time represent the time that agent n has visited a total of $j \in \begin{bmatrix}1 & 2 & ... & \#\text{graph nodes}\end{bmatrix}$ distinct nodes.

Then we find a measure $V(t)$ of how many nodes each agent still can discover given by $V(t) = (\#\text{graph nodes} - \#\text{nodes discovered at time } t) \geq 0 \quad \forall t \geq 0$ such that we can define an "energy" decrease condition where:

$$V(T_{j+1}^n) - V(T_j^n) = -1 \tag{10}$$

This equation is an abstracted version of the classic Lyapunov stability, where a continuous decrease in "energy" means that the system will reach a stable equilibrium. For times t between different discovery times $t \in \{t_k | T_{j+1}^n \geq t \geq T_j^n\}$ there is no decrease in the measure $V(t)$, which also implies that there is no Lyapunov stability guarantee for the description of the problem unless a new node is discovered. Equation (10) also implies that $V(t) = 0$ can only be reached if $0 \leq V(0) < \infty$ and hence the graph contains a finite number of nodes that agents can reach. Note that it is not necessary to know the number of total nodes in the graph, as long as it is any finite natural number.

This means that if we can guarantee that for all agents n, there is a finite amount of time $T_{j+1}^n - T_j^n$ until a not previously visited node is visited, all agents will inevitably congregate at the goal node $\mathcal{X}_{Goal}$, which is a necessary condition for the existence of a valid solution. Since there exists exactly one node that represents $\mathcal{X}_{Goal}$, all agents will be at the same node. Also, we assume that agents do not move away from the goal node.

While this necessary condition might seem trivial initially, meeting these conditions will allow the system to successfully navigate despite non-convex search spaces and any configuration of agents.

B. Solution Feasibility

Since each agent will calculate a new trajectory of N steps at each timestep in the MPC, it is necessary that each agent's state is in a set of states that has a possible input u such that all future states and inputs satisfy constraints. This set is called the control invariant set [1] and can be calculated before the problem starts.

However, the problem statement requires that prior knowledge of the environment is not given, and it would be necessary in order to calculate the actual control invariant set; this will not work. If none of the obstructions in the environment are part of the constraints that go into the calculation of the control-invariant set, and states are simply restricted to that control-invariant set, then agents will behave as if there exist no obstructions, which can lead to crashes.

If the set of all constraint abiding states is not knowable before the start of the problem, then agents will just have to understand the constraints for each situation in which they are in. For this reason, the graph made from the environment may only contain edges if there is a straight path connecting that edge's two nodes. This is easily achievable by interpolating more complex paths in the environment into segments of straight lines on the graph. Additionally, if we had chosen $x_N \in \mathcal{X}_T$, that path must be no longer than the maximal distance an agent can move in N steps, starting at rest and ending at rest.

This way, if the MPC of an agent finds a possible trajectory it is guaranteed to not run into a currently unseen obstacle later on that trajectory, which makes the currently chosen trajectory infeasible in the first place. Additionally this makes the constrained space $\mathcal{X}$ convex between two nodes, such that if we restrict the rate of change in the state, essentially

the velocity, to a maximum, we have a stabilizable system. Otherwise we would have to restrict $x_N \in \mathcal{X}_T$ such that: If at time t_k there is a feasible trajectory to $\mathcal{X}_T$ for an agent, that means that there must exist a feasible trajectory at t_{k+1} to $\mathcal{X}_T$ as well given recursive feasibility. And for the case that an agent cannot reach its target $\mathcal{X}_T$ in N steps, then there must exist other nodes that the target is reachable from. The agent can then choose the node nearest to $\mathcal{X}_T$ and have a guaranteed path from there. This can be repeated recursively until a reachable node is found, guaranteeing that an agent reaches its current target $\mathcal{X}_T$. The same goes for the target set $\mathcal{X}_f$ where, since there exists a recursively findable path from any node to the target set and that path is made up of convex subsets, we can reach each node leading up to the target node.

C. Agent Stability

Given that the feasible solution exists, we still need to prove that the controller will find the feasible trajectory. This can be achieved by proving that the cost function of the MPC problem is asymptotically stable using Lyapunov.

$$\begin{aligned}
&\text{Cost Function is Lyapunov } \leftrightarrow J^*(x(k+1)) < J^*(x(k))\\
J(x^*(k+1)) \leq & I_f(Ax_N^* + B\kappa(x_N^*)) + ...\\
& I(x_N^*, \kappa(x_N^*)) + \sum_{k=1}^{N-1} I(x_k^*, u_k^*)\\
= & I_f(Ax_N^* + B\kappa(x_N^*)) + I(x_N^*, \kappa(x_N^*)) + ...\\
& \sum_{k=0}^{N-1} I(x_k^*, u_k^*) - I(x_0^*, u_0^*)\\
= & J(x^*(k)) - I(x_0^*, u_0^*) + ...\\
& I_f(Ax_N^* + B\kappa(x_N^*)) + I(x_N^*, \kappa(x_N^*))\\
\leq & J(x^*(k)) - I(x_0^*, u_0^*)\\
\text{Assuming: } & I_f(x(k+1)) - I_f(x(k)) \leq ...\\
& - I(x(k), \kappa(x(k))) \ \forall x(k) \in \mathcal{X}_f
\end{aligned}$$

For a quadratic cost function with reference tracking, such that:

$$I_f(x_N) = ||x_N - x_s||_P^2 \tag{11}$$

$$I(x_i, u_i) = ||x_i - x_s||_Q^2 + ||u_i - \mathbb{0}^2||_R^2 \tag{12}$$

$$i = 0, ..., N-1$$

$$P, Q, R \succ 0 \tag{13}$$

the Lyapunov Stability is obviously only guaranteed while the reference r and hence x_s, u_s is constant. Since the reference changes if current reference is reached this makes the cost function only piecewise stable for intervals where the reference is constant. In this case this is allowed and necessary to approach an unknown bounded number of reference points.

To satisfy the assumption that $I_f(x(k+1)) - I_f(x(k)) \leq -I(x(k), \kappa(x(k)))\ \forall x(k) \in \mathcal{X}_f$ such that $x^T(k+1)Px^T(k+1) - x^T(k)Px^T(k) \leq -(x(k)^TQx(k) + u(k)^TRu(k))$, we can choose the infinite horizon steady-state solution P_∞, F_∞ such that:

$$\kappa(x(k)) = u(k) = Fx(k) \tag{14}$$

$$F = F_\infty = -(B^TP_\infty B + R)^{-1}B^TP_\infty A \tag{15}$$

$$\begin{aligned} P = P_\infty = & A^TP_\infty A + Q - ... \\ & A^TP_\infty B(B^TP_\infty B + R)^{-1}B^TP_\infty A \end{aligned} \tag{16}$$

Such that the assumption simplifies to:

$$x^T(k+1)Px^T(k+1) - x^T(k)Px^T(k) = \tag{17}$$

$$x(k)^T\Big(A^TPA + A^TPBF + F^TB^TPA + ... \tag{18}$$

$$F^TB^TPBF - P\Big)x(k) = \tag{19}$$

$$x(k)^T\Big(P - Q + F^TB^TPA + F^TB^TPBF - P\Big)x(k) = \tag{20}$$

$$x(k)^T\Big(F^TB^TPBF - F^TRF - F^TB^TPBF - Q\Big)x(k) = \tag{21}$$

$$x(k)^T\Big(-Q - F^TRF\Big)x(k) = \tag{22}$$

$$-x(k)^T\Big(Q + F^TRF\Big)x(k) \tag{23}$$

VII. Future Work

The greatest improvement point is finding a trade-off between the two terms of the node targeting heuristic (7). For now, both terms add to the efficiency of the system, but how to find an optimal trade-off for a given scenario is needed. Otherwise the method can be expanded by generalizing the scenario where agents may not inhabit the same space and expanding the agent systems to where they actually derive their information instead of the information just being handed to them.

Especially interesting seems to be that the heuristic may work on "hallway distance" instead of an absolute distance. This hallway distance would encrypt the best path to agents at the goal node as an example. Also it might make the trade-off in the heuristic clearer. In reality, the hallway distance might be derived from signal strength, where an agent can follow the gradient of a signal to go to it's source.

ACKNOWLEDGMENT

The Author would like to extend their thanks to her significant other, Franziska Carla Gundersen, for her emotional support and general existence. It is a pleasure.

References

[1] Mirko Fiacchini and Mazen Alamir. *Computing control invariant sets is easy*. 2017. arXiv: 1708.04797 [cs.SY]. URL: https://arxiv.org/abs/1708.04797.

[2] Li G et al. *A Pheromone-Inspired Monitoring Strategy Using a Swarm of Underwater Robots*. 2019. DOI: 10.3390/s19194089.

[3] Horyna J. et al. *Decentralized swarms of unmanned aerial vehicles for search and rescue operations without explicit communication*. 2023. DOI: https://doi.org/10.1007/s10514-022-10066-5.

[4] Yunwoo Lee and Jungwon Park. *MC-Swarm: Minimal-Communication Multi-Agent Trajectory Planning and Deadlock Resolution for Quadrotor Swarm*. 2025. arXiv: 2505.08593 [cs.RO]. URL: https://arxiv.org/abs/2505.08593.

[5] Arianne Ritz. *ATC-FS25: Diffusion Based Swarm Dynamic for Decentralized Maze Solving*. https://github.com/RitzArianne/ATC-FS25. GitHub repository. 2025. URL: https://github.com/RitzArianne/ATC-FS25.

[6] Wikipedia contributors. *Discretization*. Accessed: 22-07-2025. 2025. URL: https://en.wikipedia.org/wiki/Discretization.

[7] Wikipedia contributors. *Matrix exponential*. Accessed: 22-07-2025. 2025. URL: https://en.wikipedia.org/wiki/Matrix_exponential.