A Distributed Framework for Real Time Path Planning in Practical Multi-agent Systems

Mohamed Abdelkader * **Hassan Jaleel** ** **Jeff S. Shamma** **

* *Physical Science & Engineering (PSE), King Abdullah University of Science and Technology, Thuwal, KSA (e-mail: mohamed.abdkader@kaust.edu.sa).*
** *Computer, Electrical, and Mathematical Sciences & Engineering (CEMSE) Division, King Abdullah University of Science and Technology, Thuwal, KSA (e-mail: hassan.jaleel@kaust.edu.sa, jeff.shamma@kaust.edu.sa).*

Abstract: We present a framework for distributed, energy efficient, and real time implementable algorithms for path planning in multi-agent systems. The proposed framework is presented in the context of a motivating example of capture the flag which is an adversarial game played between two teams of autonomous agents called defenders and attackers. We start with the centralized formulation of the problem as a linear program because of its computational efficiency. Then we present an approximation framework in which each agent solves a local version of the centralized linear program by communicating with its neighbors only. The premise in this work is that for practical multi-agent systems, real time implementability of distributed algorithms is more crucial then global optimality. Thus, instead of verifying the proposed framework by performing offline simulations in MATLAB, we run extensive simulations in a robotic simulator V-REP, which includes a detailed dynamic model of quadrotors. Moreover, to create a realistic scenario, we allow a human operator to control the attacker quadrotor through a joystick in a single attacker setup. These simulations authenticate that the proposed framework is real time implementable and results in a performance that is comparable with the global optimal solution under the considered scenarios.

Keywords: Cooperative systems, Coordination of multiple vehicle systems, Control under communication constraints, Real-time control, Real-time algorithms, scheduling, and programming.

1. INTRODUCTION

The rapid advancement in the field of robotics has resulted in the development of low cost and reliable robotic platforms like quadrotors. These platforms are typically equipped with basic sensing devices for perception, and have sufficient on-board storage and processing capabilities to perform online computations. As a result, the interest in developing efficient algorithms for autonomous multi-agent systems has increased exponentially. However, one of the challenging tasks that still requires attention in developing such a system is distributed path planning under uncertain and dynamic environments in real time.

In multi-agent systems consisting of mobile robots that are battery powered, have limited processing capabilities, and communicate over noisy and shared channels, real time implementability of an algorithm has higher priority than optimal performance. In particular, for systems like quadrotors that have higher-order complex dynamics, it is extremely important to compute a feasible control action in real time. Optimality of control actions is always desirable but time complexity for computing these actions can make them undesirable for such complex systems under practical situations.

In this work, we present a distributed framework for real time path planning under adversarial environments. The distributed algorithm is presented in the context of a motivating setup of capture the flag game, which was initially presented in (D'Andrea and Babish, 2003) for teams of robots controlled by humans. Although, a variety of setups for this game can be found in the literature, we will consider a basic setup in which there are two teams of autonomous agents, called attackers and defenders, with conflicting interests. As the name suggests, there is a flag located somewhere in the playing arena. The objective of the attackers is to capture the flag by entering into a defense zone around it, whereas the objective of the defenders is to stop the attackers from capturing the flag by defending the defense zone.

Capture the flag is popular because it is a complex game with a variety of challenges that must be addressed in a practical multi-agent system (see for example D'Andrea and Babish (2003); Earl and D'Andrea (2007); Chasparis and Shamma (2008); Huang et al. (2015)). The game

⋆ Research supported by funding from KAUST.

is played under adversarial setting between two teams of autonomous robots with conflicting objectives. The players within each team collaborate with each other to achieve their objective. They compute their update actions based on the information they observe through their own sensors and the information they receive from their neighboring team members over shared communication channels. It is imperative that the update actions are computed efficiently in real time because a small delay in the computation or execution of a feasible action may result in the defeat of the entire team.

In the literature, numerous approaches have been proposed for path planning problems. One approach is to formulate the problem as a dynamic program as in Flint et al. (2002), which can guarantee optimal solution. However, except for special cases like LQR problems that have analytical solutions, dynamic programs suffer from the curse of dimensionality and cannot be solved even for a moderate size system. A computationally efficient approach is to formulate the problem as a mixed integer linear program or a linear program. However, only centralized solutions have been proposed based on these approaches (Earl and D'Andrea, 2007; Chasparis and Shamma, 2008). Another popular approach is based on model predictive control (MPC) in which each agent assumes a model for all the other dynamic agents in its environment over a finite prediction horizon and solves an optimization problem (see for example Dunbar and Murray (2002); Ferrari-Trecate et al. (2009); Müller et al. (2012)). However, these solutions typically require communication of entire trajectories among neighboring agents, which can result in excessive communication cost and latency in decision making.

In this work, we present a distributed, energy efficient, and real time implementable algorithm for path planning for capture the flag game. We start with the linear programming formulation of the problem presented in Chasparis and Shamma (2008) and propose an approximate algorithm for its distributed implementation. For the distributed implementation, each defender solves a local version of the linear program based on the current locations of its neighboring agents, and of the attackers. Since the optimization problem has to be solved over a fixed prediction horizon, each defender requires a model for the evolution of the attackers. We assume that the attackers are updating their locations based on a feedback law that moves them towards the flag.

The main contribution of this work is that we evaluate the performance of the proposed distributed framework through realistic simulations on V-REP. Typically, the algorithms presented in the existing literature are simulated in an offline environment in MATLAB in which agents are modeled as point masses with single or double integrator dynamics. These simulations are useful for analyzing the stability and convergence properties of the proposed algorithms. However, they do not provide insight about the behavior of the algorithm under actual agent dynamics and hardware constraints. Moreover, for multi-agent systems, communication among the agents over band-limited shared channels also impose serious challenges because of packet loss, latency issues, and interference. To bridge this gap between theory and practice, we simulate

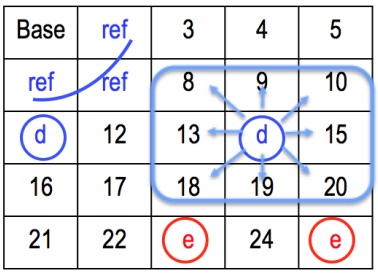

Fig. 1. Representation of the battlefield. The defense zone is $\mathcal{S}_{\text{base}} = \{s_1\}$ labeled 'Base' and $\mathcal{S}_{\text{ref}} = \{s_2, s_6, s_7\}$ labeled 'ref'. There are two defenders that are located at sectors s_{11} and s_{14}. Neighborhood of the defender at sector s_{14} is represented by the arrows pointing to sectors $\{s_8, s_9, s_{10}, s_{13}, s_{15}, s_{18}, s_{19}, s_{20}\}$. Attackers are labeled as 'e' and are located at sectors s_{23} and s_{25}.

a multi-agent system of quadrotors in V-REP. Although our problem formulation is based on a simple linear model for defenders and a linear feedback law for attackers, V-REP uses a detailed quadrotor model as a result of which the simulations are more accurate and closer to reality then simple MATLAB simulations. Moreover, to have a realistic model for an attacker that can actively update its strategies based on the actions of the defenders, we introduce human in the loop where a human controls the attacker quadrotor through a joystick. Based on these realistic simulations, we verify that the proposed framework is real time implementable.

We start by providing a detailed description of the system in Section 2.1. Then, we formulate the problem as a centralized linear program in Section 2.3. The proposed distributed algorithm, which is the main result of this work, is presented in Section 3.2. The results of real time simulations of the system using V-REP are presented in Section 4.2.

2. PROBLEM FORMULATION

2.1 System Description

The proposed framework for distributed real time path planning is presented in the context of capture the flag game. In this section we present a detailed description of the system.

- *Battlefield*: The battlefield for the game is a grid, which is defined as a collection of sectors $\mathcal{S} = \{s_1, \cdots, s_{n_s}\}$, where n_s is the total number of sectors. The defense zone is a collection of base sectors, $\mathcal{S}_{\text{base}} \subset \mathcal{S}$, that should be defended, and reference sectors, $\mathcal{S}_{\text{ref}} \subset \mathcal{S}$, that should be occupied by defenders to provide a protective perimeter around base sectors.
- *Teams*: The set of players $\mathcal{A} = \{a_1, a_2, \ldots, a_{n_p}\}$ is divided into two teams, defender team (d) and attacker team (e). Each team is represented by a set
$$\mathcal{A}^r = \{a_1^r, \cdots, a_{n_r}^r\},$$
where $r \in \{d, e\}$ is the type of the team, a_i^r is the i^{th} player of type r, and n_r is the total number of agents in the respective team r such that $n_d + n_e = n_p$.

- *States*: The state of each sector s_i is $x_{s_i} \in \mathbb{Z}_+$, which is the number of its occupants, where $\mathbb{Z}_+$ is the set of non-negative integers. The state of an agent $a_i \in \mathcal{A}$ is
$$x_{a_i} = [x_{s_1,a_i} \dots x_{s_{n_s},a_i}]^T \in \mathcal{B}^{n_s},$$
where $\mathcal{B} \triangleq \{0,1\}$ and
$$x_{s_j,a_i} = \begin{cases} 1 & a_i \text{ occupies } s_j \\ 0 & \text{otherwise.} \end{cases}$$
The state of the grid with respect to all the agents is
$$\mathbf{x} = \sum_{a_i \in \mathcal{A}} x_{a_i},$$
and the state of the grid with respect to a team $r \in \{d, e\}$ is
$$\mathbf{x}^r = \sum_{a_i^r \in \mathcal{A}_r} x_{a_i^r}.$$
- *Actions*: Actions are described by transfer of agents between sectors in $\mathcal{S}$. For an agent a_j, we define an input with respect to sector s_i as
$$u_{s_i \to s_k, a_j} = \begin{cases} 1 & x_{s_i,a_j} = 1 \text{ and } a_j \text{ transfers to } s_k \\ 0 & \text{otherwise.} \end{cases}$$
An augmented input vector for agent a_j is
$$\mathbf{u}_{s_i,a_j} = [u_{s_i \to s_1,a_j} \dots u_{s_i \to s_{i-1},a_j} u_{s_i \to s_{i+1},a_j} \dots u_{s_i \to s_{n_s},a_j}]^T.$$
If agent a_j occupies sector s_i, then $\mathbf{u}_{s_i,a_j} \in \mathcal{B}^{(n_s-1)}$ represents its transfer to some other sector in the grid. The complete input vector of agent a_j with respect to all the sectors in the grid is
$$\mathbf{u}_{a_j} = [\mathbf{u}_{s_1,a_j}^T \ \mathbf{u}_{s_2,a_j}^T \dots \mathbf{u}_{s_{n_s},a_j}^T]^T \in \mathcal{B}^{n_u}.$$
where $n_u = n_s(n_s - 1)$. Similar to states, the input vectors for defenders and attackers are
$$\mathbf{u}^d = \sum_{a_i^d \in \mathcal{A}^d} \mathbf{u}_{a_i^d} \qquad \mathbf{u}^e = \sum_{a_i^e \in \mathcal{A}^e} \mathbf{u}_{a_i^e}$$

2.2 Dynamical Model and Constraints

The evolution of the state of s_i with respect to agent a_j is modeled by the following linear model.
$$x^+_{s_i,a_j} = x_{s_i,a_j} + \sum_{s_k \in \mathcal{N}(s_i)} u_{s_k \to s_i,a_j} - \sum_{s_k \in \mathcal{N}(s_i)} u_{s_i \to s_k,a_j} \quad (1)$$
where $x^+_{s_i,a_j}$ is the updated state of sector s_i with respect to agent a_j. The first and the second terms in (1) represent the flow of a_j into and out of sector s_i, respectively, with respect to the neighborhood set of s_i defined by $\mathcal{N}(s_i)$. The neighborhood set of a sector s_i is the set of all sectors that surrounds it (see Fig. 1). The state of a sector is constrained to be non-negative (only positive resources).
$$x_{s_i,a_j} \in \mathcal{B} \triangleq \{0,1\},$$
$$0 \le \sum_{s_k \in \mathcal{N}(s_i)} u_{s_i \to s_k,a_j} \le x_{s_i,a_j} \quad (2)$$
for all $a_j \in \mathcal{A}$.

The dynamical model (1) and the constraints (2) for the defender team can be written in a compact form as
$$\begin{cases} (\mathbf{x}^d)^+ = \mathbf{x}^d + \mathbf{B}_{\text{in}}\mathbf{u}^d - \mathbf{B}_{\text{out}}\mathbf{u}^d = \mathbf{x}^d + \mathbf{B}\mathbf{u}^d \\ \mathbf{0} \le \mathbf{B}_{\text{out}}\mathbf{u}^d \le \mathbf{x}^d \\ \mathbf{x}^d \in \mathcal{B}^{n_s},\ \mathbf{u}^d \in \mathcal{B}^{n_u} \end{cases} \quad (3)$$
for appropriate $\mathbf{B}_{\text{in}}$ and $\mathbf{B}_{\text{out}}$. We refer the reader to Chasparis and Shamma (2008) for further details on the problem setup.

2.3 Problem Setup

In this game, the objective is to find optimal control inputs for defenders, $(\mathbf{u}^*)^d[t]$ for all $t \in \{0, 1, \dots, T_{\text{game}}\}$, such that interventions of the attackers into the base, $\mathcal{S}_{\text{base}}$, are minimized. To achieve this objective, we formulate the cost function at time instant t as a linear function of the state $\mathbf{x}[t]$
$$J_t(\mathbf{x}[t]) = (\alpha\mathbf{x}^e[t] + \beta\mathbf{x}_{\text{ref}}[t])^T \mathbf{x}^d[t] \quad (4)$$
This objective function captures two main performance aspects at a particular time step t. The term $\mathbf{x}^e[t]^T\mathbf{x}^d[t]$ is maximized when there is maximum overlap between the locations of defenders and attackers. Therefore, for $\alpha < 0$, the first term captures pursuit behavior by the defenders. The higher the magnitude of α, the more aggressive the defenders will be in pursuing the attackers. Similarly, $\mathbf{x}_{\text{ref}}[t]^T\mathbf{x}^d[t]$ increases as the defenders remain in the reference sectors. Thus, the second term captures surveillance behavior when $\beta < 0$, and higher magnitude of β forces the defenders to remain in $\mathcal{S}_{\text{ref}}$ and protect $\mathcal{S}_{\text{base}}$ from there. The coefficients α and β are selected to assign the desired weights to each of the behavior and they satisfy $|\alpha|, |\beta| \in [0,1]$ and $|\alpha| + |\beta| = 1$.

Thus, the optimization problem is
$$\begin{aligned} \min_{\substack{\mathbf{x}^d[t] \\ \forall t \in \{0,1,\dots,T_{\text{game}}\}}} & \sum_{t=0}^{T_{\text{game}}-1} (\alpha\mathbf{x}^e[t] + \beta\mathbf{x}_{\text{ref}})^T \mathbf{x}^d[t] \\ \text{s.t.} \quad & \mathbf{x}^d[t+1] = \mathbf{x}^d[t] + \mathbf{B}\mathbf{u}^d[t] \\ & \mathbf{0} \le \mathbf{B}_{\text{out}}\mathbf{u}^d[t] \le \mathbf{x}^d[t] \\ & \mathbf{x}^d[t] \in \mathcal{B}^{n_s},\ \mathbf{u}^d[t] \in \mathcal{B}^{n_u} \end{aligned} \quad (5)$$

3. PROBLEM SOLUTION

We aim to obtain a tractable and real-time solution of problem (5). However, to solve this optimization problem, attacker's trajectory $\mathbf{x}^e[t]$ needs to be estimated and the performance will depend on the accuracy of the attacker trajectory estimation. Following the assumption in Chasparis and Shamma (2008), we assume that the dynamical model of the attacker is similar to that of defenders as in equations (3). Also, the attacker inputs $\mathbf{u}^e$ is assumed to be a linear feedback of the form
$$\mathbf{u}^e[t] = \mathbf{G}^e \cdot \mathbf{x}^e[t], \quad (6)$$
where $\mathbf{G}^e$ is a stochastic representation on how likely the attacker will move to a particular sector that makes it closer to the base-zone defined by $\mathcal{S}_{\text{base}}$.

In the problem formulation presented in (5), there are two main sources of complexity. The mixed-integer nature of the problem and the arbitrary game time, T_{game}. To address these issues, we introducing the following simplifications.

- *State relaxation*: The state constraints in (5) are relaxed, i.e,
$$0 \leq x_{s_i,a_j} \leq 1$$
This relaxation allows us to formulate the problem as a linear program instead of mixed integer linear program, which improves the computation speed.
- *Prediction horizon*: We solve the problem in (5) in a receding horizon setup with a finite and small prediction horizon $T_p \leq T_{\text{game}}$

3.1 Centralized Solution: Linear Program Approach

We can now formulate the problem in (5) as a linear program and solve it in a centralized manner as presented in Chasparis and Shamma (2008). Define a state trajectory
$$\mathbf{X}^d = [\mathbf{x}^d[1]^T, \cdots, \mathbf{x}^d[T_p]^T]^T$$
and a control input trajectory
$$\mathbf{U}^d = [\mathbf{u}^d[0]^T, \cdots, \mathbf{u}^d[T_p-1]^T]^T$$
over the prediction horizon T_p. The objective function in (5) and the constraints in (3) can be reformulated as,

$$\begin{aligned} \min\ & [\alpha\mathbf{X}^e + \beta\mathbf{X}_{\text{ref}}]^T\ \mathbf{X}^d \\ \text{s.t.}\quad & \mathbf{X}^d - \mathbf{T}_u\ \mathbf{U}^d = \mathbf{T}_{x_0}\ \mathbf{x}^d[0] \\ & \mathbf{T}_{u,c}\ \mathbf{U}^d \leq \mathbf{T}_{x_0,c}\ \mathbf{x}^d[0] \\ & \mathbf{0} \leq \mathbf{X}^d \leq \mathbf{1} \\ & \mathbf{0} \leq \mathbf{U}^d \leq \mathbf{1} \end{aligned} \tag{7}$$

for appropriate $\mathbf{T}_u, \mathbf{T}_{x_0}, \mathbf{T}_{x_0,c}, \mathbf{T}_{u,c}$. Here
$$\mathbf{X}^e = [\mathbf{x}^e[1]^T, \cdots, \mathbf{x}^e[T_p]^T]^T$$
is the attacker trajectory. Note that the state and input constraints $\mathbf{X}^d$ and $\mathbf{U}^d$ are relaxed to formulate the linear program.

The relaxed formulation of the problem as a linear program is solved based on the principles of model predictive control (MPC). In MPC, an optimization problem is solved to compute the optimal control over a time horizon of length T_p. The control input for the first interval is applied to update the state and then the problem is solved again. The consequence of the relaxation of the binary constraints on $\mathbf{X}^d$ and $\mathbf{U}^d$ is that for an agent a_j occupying s_i, the solution to the linear program will assign weights to the neighboring sectors of s_i. Using these weights, a_i will move to that sector that will have the maximum weight.

3.2 LP-based Distributed Algorithm

In this section, we propose a distributed framework for solving the problem in (7) that eliminates the use of central authority and allows agents to achieve a suboptimal solution by using only neighborhood information and solving local optimization sub-problems. The performance of the proposed framework will be evaluated in the following sections via realistic real time simulations in V-REP and hardware implementation on a multi-agent system.

For formulating the local optimization problem for the defender a_i^d located at s_j, we define

$$\begin{aligned} \mathbf{x}^{d_i} &= \sum_{a_k^d \in \mathcal{N}(a_i^d)} x_{a_k^d} \\ \mathbf{u}^{d_i} &= \sum_{a_k^d \in \mathcal{N}(a_i^d)} u_{a_k^d} \end{aligned}$$

where $\mathcal{N}(a_i^d)$ is the neighborhood set of the defender a_i^d located at sector s_j and is defined as
$$\mathcal{N}(a_i^d) = a_i^d \cup \{a_k^d \in \mathcal{A}^d \ \mid\ x_{s_l,a_k^d} = 1 \text{ and } s_l \in \mathcal{N}(s_j)\}$$
The state and input trajectories of the neighborhood of a_i^d are
$$\begin{aligned} \mathbf{X}^{d_i} &= [\mathbf{x}^{d_i}[1]^T\ \mathbf{x}^{d_i}[2]^T \dots \mathbf{x}^{d_i}[T_p]^T]^T \\ \mathbf{U}^{d_i} &= [\mathbf{u}^{d_i}[0]^T\ \mathbf{u}^{d_i}[2]^T \dots \mathbf{u}^{d_i}[T_p-1]^T]^T \end{aligned}$$
Each defender a_i^d solves the following local problem,

$$\begin{aligned} \min\ & [\alpha\mathbf{X}_e + \beta\mathbf{X}_{\text{ref}}]^T\ \mathbf{X}^{d_i} \\ \text{s.t.}\quad & \mathbf{X}^{d_i} - \mathbf{T}_u\ \mathbf{U}^{d_i} = \mathbf{T}_{x_0}\ \mathbf{x}^{d_i}[0] \\ & \mathbf{T}_{u,c}\ \mathbf{U}^{d_i} \leq \mathbf{T}_{x_0,c}\ \mathbf{x}^{d_i}[0] \\ & \mathbf{0} \leq \mathbf{X}^{d_i} \leq \mathbf{1} \\ & \mathbf{0} \leq \mathbf{U}^{d_i} \leq \mathbf{1} \end{aligned} \tag{8}$$

The solution of the optimization problem in (8) includes the optimal trajectories $\mathbf{X}^{d_i}$ and inputs $\mathbf{U}^{d_i}$ for all the defenders a_k^d that belong to $\mathcal{N}(a_i^d)$ at time $t = 0$, from the perspective of agent a_i^d. Agent a_i^d can extract its own input at time $t = 0$ from $\mathbf{u}^{d_i}[0]$ and use it to update its state $x_{a_i^d}$ based on the model in (3).

Thus, each agent solves a local version of the centralized problem in (7) by considering only neighborhood information, i.e., current locations of the defenders in the neighborhood and the locations of all the attackers. As a special case, if all agents happen to be in the neighborhood of d, then $\mathcal{N}(a_i^d) = \mathcal{A}$ for all defenders, and the exact solution of Problem (7) is obtained.
The distributed implementation is summarized in Algorithm 1. It is important to highlight that in Algorithm 1, the agents are either sensing the locations of their neighbors or are allowed to communicate once to get the location information. This limited communication reduces the latency in computing a feasible action and improves the real time properties of the system, which is the main goal in this work. The price of reduced communication is that the constraint that two defenders should not occupy the same sector is not guaranteed. However, this issue can easily be resolved by implementing a basic collision avoidance algorithm which is an essential requirement for any practical system under all circumstances.

4. SIMULATION

In this section, we present two types of simulation results. Firstly, we provide a comparison, using MATLAB, between the performance of the proposed distributed algorithm against the centralized algorithm. Secondly, we evaluate the performance of the proposed algorithm in a game simulation setup. For that, we use MATLAB and a well know robot simulator called V-REP, to test the algorithm in multi-quadrotor setup.

Algorithm 1 LP-based Distributed Cooperative Control

1: **Data:** Discretized environment $S = \{s_1, \cdots, s_{n_s}\}$ with defense zone $\mathcal{S}_{\text{ref}}$ and $\mathcal{S}_{\text{base}}$
2: Set α and β such that $|\alpha| + |\beta| = 1$.
3: Compute optimization matrices in (8)
4: Each defender can sense the locations of the attackers, $\mathbf{x}_e[0]$.
5: Each defender $a_i^d \in \mathcal{A}^d$ performs the following steps.
6: **repeat**
7: Transmits its location to all $a_j^d \in \mathcal{N}(a_i^d)$ and receive their location.
8: From location information, forms $\mathbf{x}^{d_i}[0]$.
9: From attacker location vector $\mathbf{x}_e[0]$ and (6), forms $\mathbf{X}_e$.
10: Solves the linear program (8).
11: Extracts $\mathbf{u}_{a_i^d}[0]$.
12: Converts the relaxed control vector to binary vector. If the maximum value of the vector $\mathbf{u}_{a_i^d}[0]$ is located at index l, then set that entry equal to 1 and all the remaining entries equal to 0.
13: Applies the binary vector $\mathbf{u}_{a_i^d}[0]$ to compute next state $(\mathbf{x}_d)^+$ as in (3)
14: **until** game ends or all the attackers are captured

4.1 Offline MATLAB Simulation

We start by numerically comparing our proposed Algorithm 1, against the centralized setup. The comparison will be in terms of optimality, execution time, and required communication.

Setup: The game has 4 defenders, and 4 attackers, in 10×10 grid, with one base sector $\mathcal{S} = \{s_{45}\}$ and reference sectors $\mathcal{S}_{\text{ref}} = \{s_{34}, s_{35}, s_{36}, s_{44}, s_{46}, s_{54}, s_{55}, s_{56}\}$. The optimization prediction horizon is $T_p = 3$ time steps. We simulated 100 random configuration of defenders and attackers $(\mathbf{x^d}[\mathbf{0}], \mathbf{x^e}[\mathbf{0}])$ in the grid using Algorithm 1 and centralized algorithm in (7), and the optimal next position of defenders were obtained for both.

Figure 2 shows that, using Algorithm 1, 60% of the random scenarios had the exact centralized solution, and 34% of the random scenarios had a mismatch with the centralized solution in exactly one defender action. Thus, for 94% of the simulated scenarios, there was at most one mismatch between the centralized and the distributed solution. It is important to highlight that the distributed setup only requires one time communication between the neighboring defenders, which is before the execution of the algorithm for exchanging the initial locations, line 7 in Algorithm 1. An interesting observation is that, for this specific problem, if the initial locations of the neighbors $\mathbf{x}^{d_i}[0]$ can be sensed, then, the communication burden is eliminated, while providing near optimal solution.

The following table summarizes the numerical findings in terms of optimality, execution time, and communication requirements. These comparisons are with respect to the mentioned game setup and the 100 random simulations.

From Table 1, we can see Algorithm 1 requires low execution time and communication for each decision step, which makes it appropriate for real-time implementation.

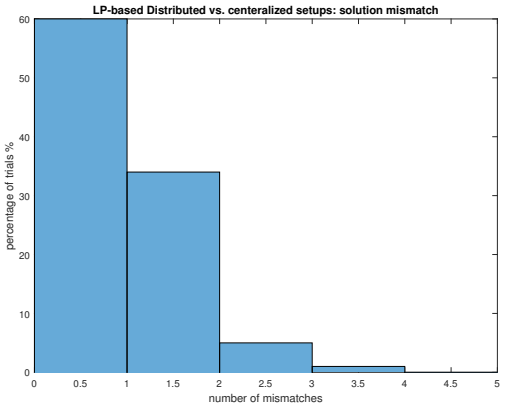

Fig. 2. LP-based Distributed vs. centralized setups: solution mismatch

Table 1. Algorithm 1 Performance

	No mismatch	One mismatch	Exec. time/scenario	Comm. steps
Algorithm 1	60%	94%	0.1 seconds	1

4.2 Online game simulation

In this section, we present the results of real time system simulation in the robotic simulator V-REP. Supplementary videos of the experiments mentioned in this section and the next are available online [1]

The analysis in section 4.1 was preformed in an offline setting. In this work, however, we specifically would like to implement Algorithm 1 in more realistic scenarios, which requires real-time assessment. For that reason, we simulated an online setting of a particular game setup. In this setup, there are two main software components

(1) MATLAB which is used to execute Algorithm 1
(2) V-REP which is a robot simulator used to simulate robots, environment dynamics, and visualization (Rohmer et al., 2013).

V-REP sends the position of the robots to MATLAB, which is used to execute Algorithm 1. The commanded position of the robots are then sent back from MATLAB to V-REP. In this setup, we use quadrotors as the agents.

Setup 1: In this setup, there are three defenders against one attacker. The attacker is controlled by a human via a joystick input.

Figure 3 shows a simulation result of Setup 2, where defenders were distributing themselves in order to attack the attacker, until it was captured by defender 2. Defender 1 had the furthest start from the attacker, and had an attacking strategy induced by $(\alpha = -0.99, \beta = -0.01)$. It tries to chase the attacker from the start to the end although it was not able to capture it. Defender 2 and 3, however, had less attacking strategy, $(\alpha = -0.6, \beta = -0.4)$ and $(\alpha = -0.5, \beta = -0.5)$, respectively. That led them to keep a closer distance to the base, and encapsulate the attacker, which was finally captured by defender 2.

[1] `https://www.dropbox.com/sh/zb2c5pemw0bh7yx/AAAj0JhSWlffpimaQfhzJOpWa?dl=0`

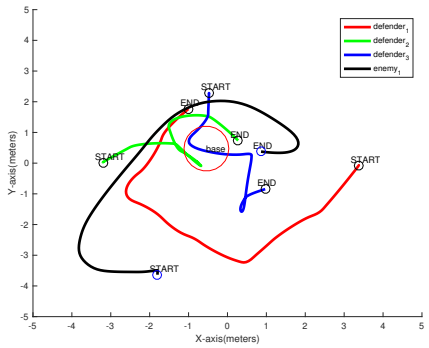

Fig. 3. Simulation with three defenders and human controlled attacker. Defender 1 has an attacking strategy induced by $\alpha = -0.99, \beta = -0.1$. Defender 2 has a less attacking strategy induced by $\alpha = -0.6, \beta = -0.4$. Defender 3 has a balanced strategy induced by ($\alpha = -0.5, \beta = -0.5$)

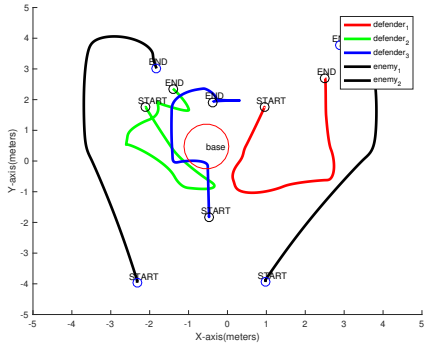

Fig. 4. Simulation with three defenders and two attackers. Defender 1 has an attacking strategy induced by $\alpha = -0.99, \beta = -0.1$. Defender 2 has a less attacking strategy induced by $\alpha = -0.6, \beta = -0.4$. Defender 3 has a balanced strategy induced by ($\alpha = -0.5, \beta = -0.5$)

Setup 2: In this setup, there are three defenders and two attackers. Defenders have mixed strategy as in Setup 1. Attacker were assigned a predefined paths as shown in Figure 4.

Figure 4 shows that defender 1, which had an attacking strategy, was chasing attacker 1. Defender 2 was chasing attacker 2 with less attacking strategy, which led it to be closer to the base than defender 1. Defender 3 had a balanced strategy, which led it stay in the middle, closer to the base.

We can notice that, more interesting strategies can be induced by adjusting the objective function's parameters, α, β. Moreover, they can be used to produce dynamic strategies in the same game, as they can be adjusted online without introducing any extra overhead on the optimization process that is carried out in Algorithm 1.

5. CONCLUSION

We presented a distributed framework for real time path planning in multi-agent systems and verified its performance by implementing it on an actual multi-agent system. The framework was proposed in the context of capture the flag game which was selected because it offered a rich variety of challenges in multi-agent systems. We started with a centralized and relaxed linear program formulation of the problem over receding horizon. Based on this relaxed LP formulation, a distributed framework was presented in which at each decision time, the defenders communicated their locations with their neighbors. Then each defender solved a local version of the original LP over the prediction horizon using its local information only. The solution of the local LP contained the trajectories of that defender and of its neighbors but from the perspective of that particular defender. Each defender only used the first control input from the local solution to update its state. The performance of this distributed LP formulation was evaluated via realistic simulations on V-REP. Thorough these realistic simulations, it was verified that the proposed framework is real time implementable. Moreover, the performance of the proposed framework was compared with the centralized problem based on offline MATLAB simulation and the performance was shown to be near optimal. An ongoing work includes more formal analytical analysis on the optimality, and actual hardware test-bench of the experiment.

REFERENCES

Chasparis, G.C. and Shamma, J.S. (2008). Lp-based multi-vehicle path planning with adversaries. *Cooperative Control of Distributed Multi-Agent Systems*, 261–279.

D'Andrea, R. and Babish, M. (2003). The roboflag testbed. In *American Control Conference, 2003. Proceedings of the 2003*, volume 1, 656–660. IEEE.

Dunbar, W.B. and Murray, R.M. (2002). Model predictive control of coordinated multi-vehicle formations. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 4, 4631–4636. IEEE.

Earl, M.G. and D'Andrea, R. (2007). *Multi-vehicle cooperative control using mixed integer linear programming*, 231–259. John Wiley & Sons, Ltd. doi:10.1002/9780470724200.ch10. URL `http://dx.doi.org/10.1002/9780470724200.ch10`.

Ferrari-Trecate, G., Galbusera, L., Marciandi, M.P.E., and Scattolini, R. (2009). Model predictive control schemes for consensus in multi-agent systems with single-and double-integrator dynamics. *IEEE Transactions on Automatic Control*, 54(11), 2560–2572.

Flint, M., Polycarpou, M., and Fernandez-Gaucherand, E. (2002). Cooperative path-planning for autonomous vehicles using dynamic programming. In *Proceedings of the IFAC 15th Triennial World Congress*, 1694–1699.

Huang, H., Ding, J., Zhang, W., and Tomlin, C.J. (2015). Automation-assisted capture-the-flag: A differential game approach. *IEEE Transactions on Control Systems Technology*, 23(3), 1014–1028.

Müller, M.A., Reble, M., and Allgöwer, F. (2012). Cooperative control of dynamically decoupled systems via distributed model predictive control. *International Journal of Robust and Nonlinear Control*, 22(12), 1376–1397.

Rohmer, E., Singh, S.P., and Freese, M. (2013). V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1321–1326. IEEE.