This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2019.2933993, IEEE Transactions on Control of Network Systems

Distributed Submodular Minimization And Motion Planning Over Discrete State Space

Hassan Jaleel and Jeff S. Shamma

***Abstract*—We develop a framework for the distributed minimization of submodular functions. Submodular functions are a discrete analog of convex functions and are extensively used in large-scale combinatorial optimization problems. While there has been a significant interest in the distributed formulations of convex optimization problems, distributed minimization of submodular functions has received relatively little research attention. Our framework relies on an equivalent convex reformulation of a submodular minimization problem, which is efficiently computable. We then use this relaxation to exploit methods for the distributed optimization of convex functions. The proposed framework is applicable to submodular set functions as well as to a wider class of submodular functions defined over certain lattices. We also propose an approach for solving distributed motion planning problems in discrete state space based on submodular function minimization. We establish through a challenging setup of the capture the flag game that submodular functions over lattices can be used to design artificial potential fields for multiagent systems with discrete inputs. These potential fields are designed such that their minima correspond to desired behaviors, i.e, agents are attracted towards their goals and are repulsed from obstacles and from each other for collision avoidance. Finally, we demonstrate that the proposed distributed framework can be employed effectively for generating feasible trajectories in such motion coordination problems.**

I. Introduction

Submodular functions play a similar role in combinatorial optimization as convex functions play in continuous optimization. These functions can be minimized efficiently in polynomial time using combinatorial or subgradient methods (see e.g. [1] and the references therein). Therefore, submodular functions have numerous applications in matroid theory, facility location, min-cut problems, economies of scale, and coalition formation (see e.g., [2], [3], [4], and [5]). Submodular functions can also be maximized approximately, which has applications in resource allocation and welfare problem [6] and [7], large scale machine learning problems [8] and [9], controllability of complex networks [10] and [11], influence maximization [12] and [13], and utility design for multiagent systems [14].

Unlike convex optimization and submodular maximization for which efficient distributed algorithms exist in the literature (see e.g., [15], [16], [17], [18], and [19]), distributed minimization of submodular functions has received relatively little research attention. Moreover, most of the existing literature on submodular optimization focuses on submodular set functions, which are defined over all the subsets of a base set. However, our focus in this work is on a wider class of submodular functions defined over ordered lattices, which are products of a finite number of totally ordered sets. In particular, we establish that submodular functions over ordered lattices can play a significant role in motion planning for multiagent systems with discrete inputs.

Our first contribution is a framework for the distributed minimization of submodular functions defined over ordered lattices. The enabler in the proposed framework is a particular continuous extension that extends any function defined over an ordered lattice to the set of probability measures. This extension, which was presented in [20], is a generalization of the Lovász extension for set functions [21], and can be computed in polynomial time through a simple greedy algorithm. The key feature of this extension is that the extended function is convex on the set of probability measures, which is a convex set, if and only if the original function is submodular. Furthermore, minimizing the extended function over the set of probability measures and minimizing the original function over an ordered lattice are equivalent if and only if the original function is submodular.

In the proposed framework, we first formulate an equivalent convex optimization problem for a given submodular minimization problem by employing the continuous extension in [20]. After formulating an equivalent convex optimization problem, we propose to implement any efficient distributed optimization algorithm for non-smooth convex functions. This combination of a convex reformulation of a submodular minimization problem and distributed convex optimization enables us to minimize a submodular function in polynomial time in a distributed manner. Although submodular optimization and distributed convex optimization have been well established areas of research for a long time, the link between these two areas has not received much research attention. Thus, our contribution is to exploit this link for the distributed minimization of submodular functions.

Distributed convex optimization has been an active area of research and numerous approaches exist in the literature for the distributed minimization of convex functions (see e.g. [22], [23], [24], and [25]). In the proposed framework, we employ the projected subgradient based algorithm presented in [25]. This algorithm is well suited for the proposed framework because a subgradient of the continuous extension of a submodular function is a byproduct of the greedy algorithm from [20], which computes the continuous extension of the submodular function. In the projected subgradient based algorithm, each

H. Jaleel is with the Department of Electrical Engineering, Syed Babar Ali School of Science & Engineering at Lahore Univeristy of Management Sciences (LUMS), Lahore, Pakistan. J.S. Shamma is with the Robotics, Intelligent Systems & Control (RISC) Lab, Computer, Electrical and Mathematical Sciences and Engineering Division (CEMSE) at King Abdullah University of Science and Technology (KAUST), Thuwal 23955–6900, Saudi Arabia. Email: hassan.jaleel@lums.edu.pk, jeff.shamma@kaust.edu.sa. Research supported by funding from KAUST.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2019.2933993, IEEE Transactions on Control of Network Systems

agent maintains a local estimate of the global optimal solution. An agent is only required to communicate with a subset of the other nodes in the network for information mixing. However, through this local communication and an update in the descent direction of a local subgradient, the algorithm asymptotically drives the estimates of all the agents to the global optimal solution.

Our second contribution is to identify a novel application domain of submodular optimization. We establish that submodular functions over ordered lattices can be used effectively for distributed motion planning in multiagent systems in which each agent has discrete inputs. Typically, motion planning problems under uncertainties are computationally complex. In multiagent systems, the size of the problem increases exponentially with the number of agents, which further increases the complexity of the problem. One approach for handling computational complexity and uncertainties in motion planning is the use of potential functions (see [26] and the references therein for details).

In the potential function based approach for motion planning, the task is to design a function whose minima correspond to a desired behavior, and whose gradient or a subgradient is easily computable. Then, by moving an agent along a descent direction of the potential function from any given initial condition, we can drive the agent to a minimum point of the potential function where the desired behavior will be achieved. In [27], it was shown that if we can simulate attractive forces for go-to-goal behavior and cohesion among agents and repulsive forces for obstacle and collision avoidance, we can design multiagent systems with complex behaviors by intelligently combining these attractive and repulsive forces.

Traditionally, potential function based approach is used when the decision variables belong to a continuous set. Our contribution is to extend this approach to the scenario when decision variables belong to discrete sets. In particular, we establish that for a motion planning problem in multiagent systems with discrete inputs, we can design such potential functions to simulate attractive and repulsive forces using submodular functions over an ordered lattice. Thus, we can find a feasible motion plan in a distributed manner by using submodular potential functions and our proposed framework for distributed submodular minimization.

To validate our claim, we consider a version of the capture the flag game from [28] and [29], which is played between two teams: offense and defense. This game is selected because it has a complex setup with both collaborative and adversarial components. Moreover, it offers a variety of challenges involved in multiagent motion coordination. We formulate the problem from the perspective of the defense team under the framework of receding horizon control with one step prediction horizon.

For this game, we design potential functions that generate attractive forces between defenders for cohesion and go-to-goal behaviors. We also design potential functions that generate repulsive forces for obstacle avoidance and collision avoidance among the members of the defense team. We prove that these potential functions are submodular over a discrete set of decision variables, and hence the overall problem is a submodular minimization problem. Thus, at each decision time, we can compute a motion plan for the defense team using our proposed framework for distributed submodular minimization in polynomial time. Finally, we show through extensive simulations that the defenders can effectively defend the defense zone while avoiding collisions and obstacles by using the motion plan computed from our proposed framework

II. Preliminaries

A. Notations

Let $S = \{s_0, s_1, \ldots, s_{m-1}\}$ be a finite set with cardinality $|S|$ and indexed by $\mathbb{Z}_+$, where $\mathbb{Z}_+$ is the set of non-negative integers. We represent a vector $\mathrm{x} \in \mathbb{R}^n$ as $\mathrm{x} = (x(0), x(1), \ldots, x(n-1))$. We refer to its i^{th} component by $\mathrm{x}(i)$, its dimension by $|\mathrm{x}|$, and its Euclidean norm by $\|x\|$. We define $\{0,1\}^{|S|}$ as the set of all vectors of length $|S|$ such that if $\mathrm{x} \in \{0,1\}^{|S|}$, then $\mathrm{x}(i) \in \{0,1\}$ for all $i \in \{0, 1, \ldots, |S|-1\}$. Similarly, $[0,1]^{|S|}$ is the set of all vectors of length $|S|$ such that if $\mathrm{x} \in [0,1]^{|S|}$, then $\mathrm{x}(i) \in [0,1]$ for all $i \in \{0, 1, \ldots, |S|-1\}$. A unit vector in $\mathbb{R}^n$ is e_i which is defined as

$$\mathrm{e}_i(k) = \begin{cases} 1 & k = i, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The indicator vector of a set $A \subseteq S$ is $\mathbb{1}_A$ and is defined as

$$\mathbb{1}_A(i) = \begin{cases} 1 & s_i \in A, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

For any two n-dimensional vectors x and y in $\mathbb{R}^n$, we say that $\mathrm{x} \leq \mathrm{y}$ if $x(i) \leq y(i)$ for all $i \in \{0, 1, \ldots, n-1\}$. Moreover, $\mathrm{x} < \mathrm{y}$ if $x(i) \leq y(i)$ for all $i \in \{0, 1, \ldots, n-1\}$ and there exists at least one $j \in \{0, 1, \ldots, n-1\}$ such that $\mathrm{x}(i) < \mathrm{y}(i)$. We define $\max(\mathrm{x}, \mathrm{y})$ and $\min(\mathrm{x}, \mathrm{y})$ as vectors in $\mathbb{R}^n$ such that

$$\max(\mathrm{x}, \mathrm{y}) = (\max(x(0), y(0)), \ldots, \max(x(n-1), y(n-1))),$$

and

$$\min(\mathrm{x}, \mathrm{y}) = (\min(x(0), y(0)), \ldots, \min(x(n-1), y(n-1))).$$

A Partially Ordered Set (POSet) is a set in which the elements are partially ordered with respect to a binary relation "$\leq$". Elements s_i and s_j in S are unordered if neither $s_i \leq s_j$ nor $s_j \leq s_i$. If $s_i \leq s_j$ and $s_i \neq s_j$, then $s_i < s_j$. A POSet S is a chain if it does not contain any unordered pair. The supremum and infimum of any $A \subset S$ are

$$\sup(A) = \min\{\bar{s} \in S \mid s \leq \bar{s} \ \forall \ s \in A\}, \text{ and}$$
$$\inf(A) = \max\{\underline{s} \in S \mid \underline{s} \leq s \ \forall \ s \in A\}.$$

The supremum and infimum of any pair s_i and s_j are represented as $s_i \vee s_j$ and $s_i \wedge s_j$ respectively. From [3], a POSet S is a lattice if for every pair of elements s_i and s_j in S

$$s_i \vee s_j \in S \text{ and } s_i \wedge s_j \in S.$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2019.2933993, IEEE Transactions on Control of Network Systems

B. Submodular Functions Over Lattices

We consider submodular functions that are real-valued functions defined on set products of the form

$$\mathcal{X} = \prod_{i=0}^{N-1} X_i.$$

In particular, our focus is on set products in which X_i is a lattice for all $i \in \{0, 1, \ldots, N-1\}$, and $\mathrm{x} \in \mathcal{X}$ is a vector, i.e., $\mathrm{x} = (x_0, x_1, \ldots, x_{N-1})$ where $x_i \in X_i$.

Definition 2.1: Let f be a real valued function defined on a lattice $\mathcal{X}$. Then, f is submodular if and only if for any pair x *and* y *in* $\mathcal{X}$

$$f(\mathrm{x} \vee \mathrm{y}) + f(\mathrm{x} \wedge \mathrm{y}) \leq f(\mathrm{x}) + f(\mathrm{y}).$$

For a function defined over a product of finite number of chains, submodularity can also be defined in terms of antitone differences. Let $\mathcal{X}$ be a product of N chains and x be an element in $\mathcal{X}$. Given $i \in \{0, 1, \ldots, N-1\}$, we define a vector $\mathrm{y}_i^{\mathrm{x}}$ as follows:

$$\begin{cases} \mathrm{y}_i^{\mathrm{x}}(j) = \mathrm{x}(j) & j \neq i, \\ \mathrm{y}_i^{\mathrm{x}}(j) \in X_i \text{ and } \mathrm{y}_i^{\mathrm{x}}(j) > \mathrm{x}(j) & j = i. \end{cases}$$

Therefore, $\mathrm{y}_i^{\mathrm{x}} > \mathrm{x}$ and is non-unique from construction. If $\mathrm{x}(i)$ is the largest element in X_i, we say that $\mathrm{y}_i^{\mathrm{x}}$ does not exist. A function $f : \mathcal{X} \to \mathbb{R}$ is antitone in i over $\mathcal{X}$ if

$$f(\mathrm{y}_i^{\mathrm{x}}) \leq f(\mathrm{x})$$

for all $\mathrm{x} \in \mathcal{X}$ and for all the possible vectors $\mathrm{y}_i^{\mathrm{x}}$. Then, from Thm. 3.2 in [3], we can verify whether a function defined on a product of finite number of chains is submodular or not as follows:

Definition 2.2: Let $\mathcal{X} = \prod_{i=0}^{N-1} X_i$ where X_i is a chain for all $i \in \{0, 1, \ldots, N-1\}$. A function $f : \mathcal{X} \to \mathbb{R}$ is submodular if

$$f(\mathrm{y}_i^{\mathrm{x}}) - f(\mathrm{x})$$

is antitone in j for all i and j in $\{0, 1, \ldots, N-1\}$, $i \neq j$, for all $\mathrm{x} \in \mathcal{X}$, and for all the possible vectors $\mathrm{y}_i^{\mathrm{x}}$.

If $X_i \subset \mathbb{Z}$ for all $i \in \{0, 1, \ldots, N-1\}$, then the above definition implies that f is submodular if $f(\mathrm{x} + \mathrm{e}_i) - f(\mathrm{x})$ is antitone, i.e.,

$$f(\mathrm{x} + \mathrm{e}_j + \mathrm{e}_i) - f(\mathrm{x} + \mathrm{e}_j) \leq f(\mathrm{x} + \mathrm{e}_i) - f(\mathrm{x}) \qquad (3)$$

for all i and j in $\{0, 1, \ldots, N-1\}$, $i \neq j$. If X_i's are continuous intervals of $\mathbb{R}$, then Def. 2.2 implies that f is submodular if

$$\frac{\partial^2 f}{\partial x_i \partial x_j}(\mathrm{x}) \leq 0$$

for all $\mathrm{x} \in \mathcal{X}$ and i and j in $\{0, 1, \ldots, N-1\}$, $i \neq j$. Thus, in the case of chain products, the condition in the above definition reduces the question of submodularity to comparing all pairs of cross differences.

III. Submodular Function Minimization

A brief overview of the tools and techniques for minimizing submodular set functions that are relevant to this work is presented in [30].

A. Submodular Minimization Over Ordered Lattices

In [20], it was shown that most of the results relating submodularity and convexity like efficient minimization via Lovász extension ([21]) can be extended to submodular functions over lattices. In particular, lattices defined by chain products were considered and an extension was proposed in the set of probability measures. It was proved that the proposed extension on the set of probability measures was convex if and only if the original function defined over the product of chains was submodular. Moreover, it was proved that minimizing the original function was equivalent to minimizing the proposed convex extension on the set of probability measures.

A greedy algorithm was also presented in [20] for computing the continuous extension of a submodular function defined over a finite chain product. This greedy algorithm will play a central role in our proposed solution approach to distributed motion planning with discrete inputs. Therefore, we present the algorithm here for the completeness of presentation. For details, we refer the readers to [20].

Let $\mathcal{X}$ be a product of N discrete sets with finite number of elements

$$\mathcal{X} = \prod_{i=0}^{N-1} X_i.$$

We assume that $X_i = \{s_0, s_1, \ldots, s_{m_i-1}\}$ is a chain for all i, which implies that the product set $\mathcal{X}$ is a lattice. Since X_i's are chains, we can order their elements and represent each set by the index set

$$X_i = \{0, 1, \ldots, m_i - 1\}.$$

Then, any $\mathrm{x} \in \mathcal{X}$ will be an index vector. Let $P(X_i)$ be the set of all probability measures on X_i. Then, $\mu_i \in P(X_i)$ is a vector in $[0, 1]^{m_i}$ such that

$$\sum_{j=0}^{m_i-1} \mu_i(j) = 1.$$

For a product set $\mathcal{X}$, let $\mathcal{P}(\mathcal{X})$ be the set of product probability measures, i.e., for any $\mu \in \mathcal{P}(\mathcal{X})$

$$\mu = \prod_{i=0}^{N-1} \mu_i, \ \mu_i \in P(X_i) \ \ \forall \ \ i.$$

A probability measure μ_i is degenerate if $\mu_i(j) = 1$ for some $j \in \{0, 1, \ldots, m_i - 1\}$. We define $F_{\mu_i} : X_i \to \mathbb{R}$ as

$$F_{\mu_i}(j) = \sum_{l=j}^{m_i-1} \mu_i(l).$$

Thus, F_{μ_i} is similar to cumulative distribution function but is reverse of it. Since μ_i is a probability measure, $F_{\mu_i}(0)$ is always equal to one. Therefore, we will ignore $F_{\mu_i}(0)$ and only consider $m_i - 1$ values to reduce dimension of the problem.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2019.2933993, IEEE Transactions on Control of Network Systems

For a probability measure μ_i on X_i, we define a vector ρ_i as

$$\rho_i = (F_{\mu_i}(1), F_{\mu_i}(2), \ldots, F_{\mu_i}(m_i - 1)). \tag{4}$$

Since

$$F_{\mu_i}(j+1) \leq F_{\mu_i}(j)$$

for all $j \in \{0, 1, \ldots, m_i - 1\}$, ρ_i is a vector with non-increasing entries. The equality $\rho_i(j) = \rho_i(j+1)$ occurs if and only if $\mu_i(j) = 0$. Thus, $\rho_i \in [0,1]^{m_i - 1}_{\downarrow}$ where

$$[0,1]^{m_i-1}_{\downarrow} = \{\tilde{\rho} \in [0,1]^{m_i-1} \ : \ \tilde{\rho}(i+1) \leq \tilde{\rho}(i) \ \forall \ i\}.$$

For a product set $\mathcal{X}$, we define the set $\Omega(\mathcal{X})$ as

$$\Omega(\mathcal{X}) = \prod_{i=0}^{N-1} [0,1]^{m_i-1}_{\downarrow}. \tag{5}$$

Let $\theta_{\rho_i} : [0,1] \to \{0, 1, \ldots, m_i - 1\}$ be an inverse map of ρ_i and is defined as

$$\theta_{\rho_i}(t) = \max\{0 \leq l \leq m_i - 1 : \rho_i(l) \geq t\}.$$

From the definition of θ_{ρ_i},

$$\theta_{\rho_i}(t) = \begin{cases} m_i - 1 & t < \rho_i(m_i - 1), \\ l & \rho_i(l+1) < t < \rho_i(l), \\ & l \in \{1, 2, \ldots, m_i - 2\}, \\ 0 & t > \rho_i(1). \end{cases}$$

The boundary values can be arbitrary and does not impact the overall setup. The definition of θ_{ρ_i} is extended to a product set $\mathcal{X}$ as follows

$$\theta_{\rho}(t) = \prod_{i=0}^{N-1} \theta_{\rho_i}(t), \tag{6}$$

where $t \in [0,1]$.

Let $f : \mathcal{X} \to \mathbb{R}$ be a real valued function defined over $\mathcal{X}$. Then, the greedy algorithm for computing an extension of f over a continuous space is presented in Alg. 1. The extension f^{ext} of f is given in (8) and the subgradient of f^{ext} is in (10). The algorithm requires sorting r values, which has a complexity of $O(r \log r)$, and r evaluations of the function, where r is defined in (7). We refer the reader to [20] for the details and the complexity analysis of the greedy algorithm.

It was proved in [20] that for a function $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is a product of N finite chains, $f^{\text{ext}}(\rho)$ is convex if and only if f is submodular. It was also proved that minimizing f over $\mathcal{X}$ is equivalent to minimizing f^{ext} over $\Omega(\mathcal{X})$, i.e,

$$\min_{\mathrm{x} \in \mathcal{X}} \ f(\mathrm{x}) = \min_{\rho \in \Omega(\mathcal{X})} f^{\text{ext}}(\rho),$$

and $\rho^* \in \Omega(\mathcal{X})$ is the minimizer for f^{ext} if and only if $\theta_{\rho^*}(t)$ is a minimizer for f for all $t \in [0,1]$. Therefore, by minimizing f^{ext} over $\Omega(\mathcal{X})$, we can find a minimizer for a submodular function f over an ordered lattice $\mathcal{X}$.

Algorithm 1 Greedy Algorithm

Require: $\rho = \prod_{i=0}^{N-1} \rho_i$.

1: Form a set Q as follows.

$$Q = \{\rho_0(1), \ldots, \rho_0(m_0 - 1), \rho_1(1), \ldots, \rho_1(m_1 - 1), \ldots, \rho_{N-1}(1), \ldots, \rho_{N-1}(m_{N-1} - 1)\}.$$

The number of elements in Q is

$$r = \sum_{i=0}^{N-1} m_i - N. \tag{7}$$

In the summation, we subtract N because $F_{\mu_i}(0)$ is neglected in ρ_i for all i.

2: Arrange all the r values of Q in decreasing order in the set Q_{dec}, i.e.,

$$Q_{\text{dec}} = \{\rho_{i_1}(j_1), \rho_{i_2}(j_2), \ldots, \rho_{i_r}(j_r)\},$$

such that

$$\rho_{i_1}(j_1) \geq \rho_{i_2}(j_2) \geq \ldots \geq \rho_{i_r}(j_r).$$

The ties are handled randomly. However, in the case of ties within ρ_i for some i, the order of the values are maintained.

3: Compute the extension of function f over the probability measures as follows

$$f^{\text{ext}}(\rho) = f(0) + \sum_{s=1}^{r} \mathrm{t}(s) \left(f(\mathrm{y}_s) - f(\mathrm{y}_{s-1})\right), \tag{8}$$

where

$$\mathrm{t}(s) = \rho_{i_s}(j_s) \ \ \forall \ \ s \in \{1, 2, \ldots, r\}.$$

Moreover, the vector $\mathrm{y}_s \in \mathcal{X}$ is

$$\mathrm{y}_s = \begin{cases} (0, 0, \ldots, 0) & s = 0, \\ \mathrm{y}_{s-1} + e_{i_s} & 1 \leq s \leq r-1, \\ (m_0 - 1, m_1 - 1, \ldots, m_{N-1} - 1) & s = r. \end{cases} \tag{9}$$

4: The subgradient of f^{ext} evaluated at ρ is

$$\partial f^{\text{ext}}\big|_{\rho} = \prod_{i=0}^{N-1} \partial f^{\text{ext}}\big|_{\rho_i}.$$

The j^{th} component of $\partial f^{\text{ext}}\big|_{\rho_i}$ is

$$\partial f^{\text{ext}}\big|_{\rho_i}(j) = f(\mathrm{y}_g) - f(\mathrm{y}_{g-1}), \tag{10}$$

where $g = \min\{s \in \{1, 2, \ldots, r\} : \mathrm{y}_s(i) = j\}$.

IV. Distributed Submodular Minimization

In this section, we present the main contribution of this work, which is a distributed algorithm for minimizing a submodular function defined over a product of N chains.

Consider a system comprising N agents, $\{v_0, v_1, \ldots, v_{N-1}\}$. The global objective is to minimize a cost function, which is the sum of N terms over a product set $\mathcal{X}$. Each agent has information about one term only in the global cost function. Thus, the agents need to solve the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2019.2933993, IEEE Transactions on Control of Network Systems

following optimization problem collaboratively

$$\min_{\mathrm{x}\in\mathcal{X}} \quad J(\mathrm{x}) = \sum_{i=0}^{N-1} J_i(\mathrm{x}), \tag{$\mathcal{P}$}$$

where

$$\mathcal{X} = \prod_{j=0}^{p-1} X_j, \quad |X_j| = m_j.$$

We assume that $J_i : \mathcal{X} \to \mathbb{R}$ is a submodular function and each X_j is a chain. Since the total cost is a sum of N submodular functions, it is also a submodular function.

The cost function of each agent in ($\mathcal{P}$) depends on the entire decision vector, which is global information. However, we assume that each agent has access to local information only. The local information of agent v_i consists of the term J_i in the cost function. Moreover, each agent is allowed to communicate with a subset of other agents in the network. Therefore, no agent has direct access to any global information. The communication network is represented by a graph $\mathbb{G}(V, \mathcal{E})$, where $V = \{v_0, v_1, \ldots, v_{N-1}\}$ is the set of vertices and $\mathcal{E} \subseteq V \times V$ is the set of edges. An edge $(v_i, v_j) \in \mathcal{E}$ implies that agent v_i has access to the information of v_j. The neighborhood set of v_i contains v_i and the agents with which v_i can communication, i.e.,

$$N(v_i) = \{v_i\} \cup \{v_j \in V : (v_i, v_j) \in \mathcal{E}\}.$$

The communication network is represented algebraically by a weighted incidence matrix A defined as follows:

$$A(i,j) = \begin{cases} a_{ij} & v_j \in N(v_i), \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

where $a_{ij} \geq 0$ for all i and j.

To find a minimizer of $J(\mathrm{x})$, we propose a relaxation based approach in which we formulate a relaxed problem that is equivalent to ($\mathcal{P}$). The structure of the relaxed problem is as follows:

$$\min_{\rho\in\Omega(\mathcal{X})} \quad J^{\mathrm{ext}}(\rho) = \sum_{i=0}^{N-1} J_i^{\mathrm{ext}}(\rho). \tag{$\mathcal{P}1$}$$

In the relaxed problem, J^{ext} is the extension of J, which is computed through (8) in Alg. 1, and $\Omega(\mathcal{X})$ is the constraint set defined in (5). To find a solution to ($\mathcal{P}1$), we implement Alg. 2, which is the consensus based projected subgradient algorithm from [25]. Finally, we compute a minimizer for $J(\mathrm{x})$ in ($\mathcal{P}$) from the solution of ($\mathcal{P}1$) computed from Alg. 2.

Since the cost of each agent depends on the entire state vector, agents need global information to solve the optimization problem. The main idea in Alg. 2 is that each agent generates and maintains an estimate of the optimal solution based on its local information and communication with its neighbors. The local information of agent v_i is the cost function J_i. It solves a local optimization problem and exchanges its local estimate of the solution with its neighbors. Then, it updates its estimate of the optimal solution by mixing the information it received from its neighbors and the process is repeated. The details are presented in Alg. 2 from the perspective of agent v_i.

Algorithm 2 Distributed Submodular Minimization

To solve the optimization problem ($\mathcal{P}1$), agent v_i has to perform the following steps:

1: Select any $\rho \in \Omega(\mathcal{X})$, where $\Omega(\mathcal{X})$ is defined in (5). Set

$$\rho^i[0] = \rho,$$

2: At each time k, update $\rho^i[k-1]$ as follows
3: **for** $k = 1$ to iter **do**
4: **for** $j = 0$ to $p-1$ **do**

$$\nu_j^i = \sum_{w=0}^{N-1} a_{iw} \rho_j^w[k-1] \tag{12}$$

5: **end for**
6: Set

$$\rho^i[k] = \mathbb{P}_{\Omega(\mathcal{X})}\left(\nu^i - \gamma_k \partial J_i^{\mathrm{ext}}\big|_{\nu^i}\right), \tag{13}$$

where $\nu^i = \prod_{j=0}^{p-1} \nu_j^i$, and γ_k is the step size such that $\sum_k \gamma_k = \infty$ and $\sum_k \gamma_k^2 < \infty$. The term $\partial J_i^{\mathrm{ext}}\big|_{\nu^i}$ is a sub-gradient of J_i^{ext} evaluated at ν^i and can be computed using (10).

7: **end for**
8: Set $\hat{\rho}^i = \rho^i[\mathrm{iter}]$.
9: Agent v_i's estimate of an optimal solution for ($\mathcal{P}$) is

$$\hat{\mathrm{x}}^i[\mathrm{iter}] = \theta_{\hat{\rho}^i}(\hat{t}) \tag{14}$$

for some $\hat{t} \in [0, 1]$. Here $\theta_{\hat{\rho}^i}(\hat{t})$ is computed from (6).

In Alg. 2, agent v_i starts by initializing its estimate ρ^i of the optimal solution with a feasible product vector, i.e., $\rho^i[0] \in \Omega_{\mathcal{X}}$. To update $\rho^i[k]$ for all $j \in \{0, 1, \ldots, N-1\}$, v_i exchanges its local estimate with all the agents in its neighborhood set $N(v_i)\backslash\{v_i\}$. The estimates are updated in two steps, a consensus step and a gradient descent step. The consensus step is in (12) in which v_i computes a weighted combination of the estimates of $v_k \in N(v_i)$ by assigning weight a_{iw} to $\rho^w[k-1]$. The gradient descent step is in (13), in which the combined estimate ν^i is updated in the direction of gradient descent of J_i^{ext} evaluated at ν^i. Here, γ_k is the step size of the descent algorithm at time k. The gradient $\partial J_i^{\mathrm{ext}}\big|_{\nu^i}$ is computed through the greedy algorithm.

Finally, $\mathbb{P}_{\Omega(\mathcal{X})}(\xi^i)$ is the projection operator that projects ξ^i on the constraint set $\Omega(\mathcal{X})$. Let

$$\xi^i = \nu^i - \gamma_k \partial J_i^{\mathrm{ext}}\big|_{\nu^i},$$

Since ν^i and $\partial J_i^{\mathrm{ext}}$ are product vectors,

$$\xi^i = \prod_{j=0}^{p-1} \xi_j^i, \quad j \in \{0, 1, \ldots, p-1\}, \text{ and}$$
$$\xi_j^i = \nu_j^i - \gamma_k \partial J_i^{\mathrm{ext}}\big|_{\nu_j^i}.$$

Thus, the projection of ξ^i on $\Omega(\mathcal{X})$ can be decomposed into projecting each ξ_j^i on $[0,1]_{\downarrow}^{m_j-1}$ for which we solve the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2019.2933993, IEEE Transactions on Control of Network Systems

following problem.

$$\min_{\tilde{\rho}\in[0,1]^{m_j-1}} \|\tilde{\rho}-\xi_j^i\|^2 \quad \text{s.t.} \quad C\tilde{\rho}\leq 0_{m_j-2}, \tag{15}$$

where $0_{m_j-2}\in\mathbb{R}^{m_j-2}$ with all entries equal to 0 and $C\in\mathbb{R}^{(m_j-2)\times(m_j-1)}$ with entries equal to

$$C_{uz}=\begin{cases}-1 & \text{if} u=z,\\ 1 & \text{if} z=u+1,\\ 0 & \text{otherwise.}\end{cases}$$

The inequality constraints ensure that the solution to (15) has non-increasing entries, i.e.,

$$\tilde{\rho}(u+1)-\tilde{\rho}(u)\leq 0 \quad \forall \quad u\in\{0,1,\ldots,m_j-2\}.$$

The vector ρ^i is updated through Eqs (12) and (13) for iter number of iterations. After iter iterations, agent v_i's estimate of the optimal solution for $(\mathcal{P}1)$ is

$$\hat{\rho}^i=\rho^i[\mathrm{iter}].$$

Based on this estimate, agent v_i computes $\hat{x}^i$ as in (14), which is its estimate of the optimal solution for the problem in $(\mathcal{P})$

Proposition 1: *If the communication graph $G(V,\mathcal{E})$ and the corresponding adjacency matrix defined in (11) satisfy the following conditions:*

1) *$\mathbb{G}(V,\mathcal{E})$ is strongly connected.*
2) *There exists a scalar $\eta\in(0,1)$ such that $a_{ii}\geq\eta$ for all $i\in\{0,1,\ldots,N-1\}$.*
3) *For any pair of agents $(v_i,v_j)\in\mathcal{E}$, $a_{ij}\geq\eta$.*
4) *Matrix A is doubly stochastic, i.e., $\sum_{i=0}^{N-1}a_{ij}=1$ and $\sum_{j=0}^{N-1}a_{ij}=1$ for all i and j in $\{0,1,\ldots,N-1\}$*

Then,

$$\hat{\mathrm{x}}^i[\mathrm{iter}]\to \mathrm{x}^* \text{ as } \mathrm{iter}\to\infty, \tag{16}$$

where $\hat{\mathrm{x}}^i[\mathrm{iter}]$, computed in (14), is the estimate of agent v_i for $\mathrm{x}^*\in\mathcal{X}^*$, which is an optimal solution to the submodular minimization problem $(\mathcal{P})$ and $\mathcal{X}^*$ is the set of optimal solutions to the problem.

Proof: Proposition 1 states that as the number of updates in Alg. 2 approaches infinity, the estimate of agent v_i of the global solution to $(\mathcal{P})$ approaches to an optimal solution of $(\mathcal{P})$. Thus, each agent can find a global minimizer of the submodular function $J(\mathrm{x})$ in $(\mathcal{P})$ in a distributed manner.

The proof of the proposition is as follows. As stated before, Alg. 2 is the consensus based projected subgradient algorithm presented in [25] for constrained convex optimization problems. Since the relaxed problem $(\mathcal{P}1)$ is a constrained convex optimization problem as shown in [20], we can deduce directly from [25] that if conditions 1 to 4 in the proposition statement are satisfied, then

$$\rho^i[\mathrm{iter}]\to\rho^* \text{ as } \mathrm{iter}\to\infty, \tag{17}$$

where $\rho^i[\mathrm{iter}]$, computed in (4), is agent v_i's estimate of the optimal solution for $(\mathcal{P}1)$ after iter iterations and ρ^* is an optimal solution of $(\mathcal{P}1)$.

Based on the results in [20], problems $(\mathcal{P})$ and $(\mathcal{P}1)$ are equivalent. Therefore, we can find an optimal solution to $(\mathcal{P})$ through an optimal solution to $(\mathcal{P}1)$. Let $\mathcal{X}^*\subseteq\mathcal{X}$ be the set of optimal solutions of $(\mathcal{P})$. From (14), the estimate of agent v_i for optimal solution to $(\mathcal{P})$ is $\hat{x}^i[\mathrm{iter}]$ and it depends on $\rho^i[\mathrm{iter}]$. Then, the fact that $\rho^i[\mathrm{iter}]$ approaches to an optimal solution ρ^* for all i implies that $\hat{x}^i[\mathrm{iter}]\to\mathrm{x}^*$ as $\mathrm{iter}\to\infty$, where $\mathrm{x}^*=\theta_{\rho^*}(t)$ belongs to $\mathcal{X}^*$ for all $t\in[0,1]$. ■

Note: An important note related to finding optimal solution to $(\mathcal{P})$ from the solution to $(\mathcal{P}1)$. We know from [20] that $\theta_{\rho^*}(t)\in\mathcal{X}^*$ for all $t\in[0,1]$. Let $t^i\in[0,1]$ be the value used by agent i to compute its optimal solution $\theta_{\rho^*}(t^i)$. If $|\mathcal{X}^*|=1$, i.e., $\mathcal{P}$ has a unique optimal solution x^*, then $\theta_{\rho^*}(t^i)=\mathrm{x}^*$ for all i. However, if $|\mathcal{X}^*|>1$, t^i and t^j can lead to different elements in $\mathcal{X}^*$ for $t^i\neq t^j$. Therefore, if there is an additional constraint that all the agents should select the same optimal solution, we need to set

$$t^i=\hat{t} \text{ for all } i\in\{0,1,\ldots,N-1\}$$

for some $\hat{t}\in[0,1]$. The agents can decide on a value of $\hat{t}$ by running a parallel consensus algorithm on t in Alg. 2.

In Alg. 2, there are two primary operations that an agent performs in every iterations. The first operation is the computation of subgradient, which is computed through Alg. 1. The complexity of this algorithm was already discussed in the previous section. The second operation is the projection of the updated estimate of the optimization vector on the constraint set by solving (15). This is an isotonic regression problem and can be solved by any quadratic program solver.

V. Distributed Motion Planning over Discrete Domain

Our second contribution is a novel application domain of submodular function minimization. We establish through a challenging setup of capture the flag game that submodular function minimization can play a fundamental role in motion planning under uncertain environments for multiagent systems in which each agent has discrete set of inputs.

As outlined in the classical "boids" model in [27], the motion of an individual agent in a multiagent system should be a combination of certain fundamental behaviors. These behaviors include collision avoidance, cohesion, and alignment. Cohesion corresponds to the tendency of the agents to remain close to each other, and alignment refers to the ability of the agents to align with a desired orientation and reach a desired goal point. In addition to these behaviors, agents should be able to avoid any obstacles in the environment.

We demonstrate that the behaviors in the "boids" model can be achieved by solving submodular minimization problems over discrete inputs. Our contributions are as follows:

- We propose potential functions ((20), (21), and (22)) to simulate attractive forces between agents and prove that these functions are submodular over the set of discrete inputs. We show that we can achieve go-to-goal, alignment, and cohesion behaviors based on these attractive potential functions.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2019.2933993, IEEE Transactions on Control of Network Systems

- We design a potential function (23) that simulates repulsive forces and prove that this function is submodular over the set of discrete inputs as well. This potential function generates repulsive forces between an agent and entire planes. We show that through this repulsive potential function, we can achieve collision avoidance and obstacle avoidance.
- We establish the effectiveness of the proposed motion planning approach based on submodular potential functions through an example setup, which is inspired from the capture the flag game as presented in [28] and [29]. Capture the flag is a challenging setup that involves two teams of agents competing against each other. We demonstrate through extensive simulations that by formulating the problem in terms of submodular potential functions and solving it using our proposed distributed submodular minimization framework, each agent can compute an effective motion plan for itself in a distributed manner.

A. Problem Formulation

Capture the flag game is played between two teams of agents, offense and defense, over a time interval of length T. We will refer to the members of the offense and defense teams as attackers and defenders respectively. The arena is a square region of area N_g^2 that is discretized into a two dimensional grid having $N_g \times N_g$ sectors as shown in Fig. 1. The discretized arena is represented by a set $\mathcal{G} = G \times G$, which is an integer lattice, i.e., each $z = (x, y)$ in $\mathcal{G}$ is a vector in $\mathbb{Z}^2$, where x and y belong to $G = \{0, 1, 2, \ldots, N_g - 1\}$.

A flag is assumed to be placed in the arena and the area surrounding it is declared as a defense zone. The defense zone $D = \{z_0^f, z_1^f, \ldots, z_{n_f-1}^f\}$ is a subset of $\mathcal{G}$ with n_f points. The points in D are stacked in a vector

$$\mathrm{z}^f = (z_0^f, z_1^f, \ldots, z_{n_f-1}^f), \quad \mathrm{z}^f \in \mathbb{Z}^{2n_f}$$

in which each $z_i^f = (x_i^f, y_i^f)$ is a point in $\mathbb{Z}^2$.

The grid points of the shaded region at the top of Fig. 1 comprise the defense zone. The objective of the offense is to capture the flag. The flag is considered captured if any attacker reaches a point in the defense zone. Once the flag is captured, the game stops and the offense wins. On the other hand, the objective of the defense is to stop the attackers from entering the defense zone either by capturing them or forcing them away. To defend the defense zone, there needs to be collaboration and cohesion among the defenders. An attacker is in captured state if its current location is shared by at least one defender. However, if that defender moves to a different location, the state of the attacker switches from captured to active. If no attacker can enter the defense zone for the duration of the game, the defense wins.

Let $P = \{a, d\}$ be a set of teams where a and d correspond to the teams of attackers and defenders respectively. Let n_p be the number of players and p_i be the i^{th} player in team $p \in P$. The locations of all the players in a team at time k are stacked in a vector $\mathrm{z}^p(k) \in \mathbb{Z}^{2n_p}$ where the location of p_i at time k is $z_i^p(k) = (x_i^p(k), y_i^p(k))$. The update equation for p_i is

$$z_i^p(k+1) = z_i^p(k) + u_i^p(k),$$

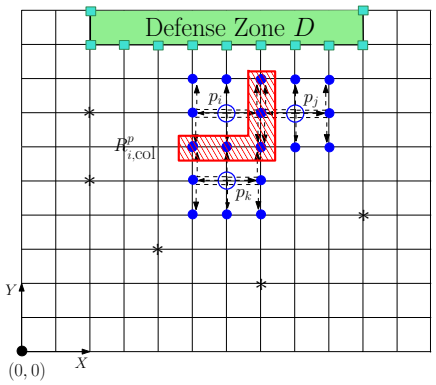


Fig. 1. Layout of the playing arena.

where $u_i^p(k) = (u_{x,i}^p(k), u_{y,i}^p(k)) \in U \times U$, and

$$U = \{-u^{\max}, -u^{\max}+1, \ldots, 0, \ldots, u^{\max}-1, u^{\max}\}.$$

Let $U_i^p = U \times U$ be the input set of player p_i. Then, the reachable set of p_i at time k is

$$R_i^p(k) = \{z \in \mathcal{G} : z = z_i^p(k) + u_i^p, \quad u_i^p \in U^p\},$$

i.e., $R_i^p(k)$ is the set of all points that p_i can reach in one time step. For notational convenience, we will drop k from the arguments. The reachable sets of players with $u^{\max} = 1$ are depicted in Fig. 1.

Two players can collide if their reachable sets overlap with each other. Let

$$R_{i,\mathrm{col}}^p = \bigcup_{\substack{j=1 \\ j \neq i}}^{n_p} \left(R_i^p \cap R_j^p\right).$$

$R_{i,\mathrm{col}}^p$ is the set of all points in R_i^p that can result in a collision between p_i and the members of its team. The shaded region in Fig. 1 depicts $R_{i,\mathrm{col}}^p$ for p_i. To make the game more challenging and to add obstacle avoidance, we assume that some point obstacles are placed in the arena. Let

$$\mathcal{O} = \{z_1^{\mathrm{obs}}, \ldots, z_{n_{\mathrm{obs}}-1}^{\mathrm{obs}}\}$$

be the set of obstacle locations. In Fig. 1, the obstacles are represented by asterisks.

For player p_i, we combine the sets of points of possible collisions with other players and with obstacles as follows:

$$R_{i,\mathrm{avoid}}^p = R_{i,\mathrm{col}}^p \cup (R_i^p(k) \cap \mathcal{O}). \tag{18}$$

One approach to guarantee collision avoidance is to limit the effective reachable set of p_i to $R_i^p \backslash R_{i,\mathrm{avoid}}^p$, which is the set of points of R_i^p that are not included in $R_{i,\mathrm{avoid}}^p$. To avoid the points in $R_{i,\mathrm{avoid}}^p$, we compute $X_{i,\mathrm{avoid}}^p$ and $Y_{i,\mathrm{avoid}}^p$. These are sets of collision avoidance planes along x and y direction such that avoiding these entire planes guarantee that $z_i^p \in R_i^p \backslash R_{i,\mathrm{avoid}}^p$ in the next time step. We explain collision avoidance planes through examples in Fig. 2. In both the cases, the shaded regions are the sets where collisions can occur. In Fig. 2(a), if p_i avoids the entire plane $x = x_i^p + 1$ and p_j

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2019.2933993, IEEE Transactions on Control of Network Systems

avoids the entire plane $x = x_j^p - 1$, then p_i and p_j cannot collide at time $k+1$. In this case

$$X^p_{i,\text{avoid}} = \{x_i^p + 1\},\ Y^p_{i,\text{avoid}} = \emptyset,$$
$$X^p_{j,\text{avoid}} = \{x_j^p - 1\},\ Y^p_{j,\text{avoid}} = \emptyset.$$

Similarly, the avoidance planes in Fig. 2(b) are

$$X^p_{i,\text{avoid}} = \{x_i^p + 1\},\ Y^p_{i,\text{avoid}} = \{y_i^p + 1\},$$
$$X^p_{j,\text{avoid}} = \{x_j^p - 1\},\ Y^p_{j,\text{avoid}} = \emptyset,$$
$$X^p_{l,\text{avoid}} = \emptyset, Y^p_{l,\text{avoid}} = \{y_l^p - 1\}.$$

For $u_x^{\max} = u_y^{\max} = 1$, the sets $X^p_{i,\text{avoid}}$ and $Y^p_{i,\text{avoid}}$ can be computed easily.

Next, we formulate the problem form the perspective of defense team. In the game setup, we assume that at time k each defender knows the current locations of all the attackers. However, any mobility strategy for the defense team inherently depends on the strategy of the attack team, which is unknown to the defenders. Therefore, we implement an online optimization strategy, in which the defenders assume a mobility model for attackers. At each time k, the online optimization problem has the following structure.

$$\begin{aligned} \min_{\mathrm{u}^d \in \mathcal{U}^d} \quad & J(\mathrm{z}^d, \mathrm{u}^d, (\mathrm{z}^a)^+, \mathrm{z}^f, \mathrm{z}^{\text{obs}}), \\ \text{s.t.} \quad & (\mathrm{z}^d)^+ = \mathrm{z}^d + \mathrm{u}^d. \end{aligned} \tag{$\mathcal{P}2$}$$

where $\mathcal{U}^d = \prod_{i=0}^{n_d-1} U_i^d$ and $U_i^d = \{-1, 0, 1\} \times \{-1, 0, 1\}$.

In this problem formulation, z^a and z^d are the location vectors of the offense and defense teams at time k, and $(\mathrm{z}^a)^+$ is the location vector of offense at time $k+1$. To solve ($\mathcal{P}2$), defenders still need to know $(\mathrm{z}^a)^+$, which cannot be known at current time. Therefore, defenders assume a mobility model for attackers. The assumed model can be as simple as the shortest path form the current location of an attacker to the defense zone. The model can also be more sophisticated like a feedback strategy as presented in [29].

The cost function J in ($\mathcal{P}2$) is

$$J(\mathrm{z}^d, \mathrm{u}^d, (\mathrm{z}^a)^+, \mathrm{z}^f, \mathrm{z}^{\text{obs}}) = \sum_{i=0}^{n_d-1} J_i(\mathrm{z}^d, \mathrm{u}^d, (\mathrm{z}^a)^+, \mathrm{z}^f, \mathrm{z}^{\text{obs}}).$$

The total cost is the sum of the costs of individual defenders. The local cost of each defender is

$$J_i = \alpha_i^f J_i^f(z_i^d, u_i^d, \mathrm{z}^f) + \alpha_i^a J_i^a(z_i^d, u_i^d, (\mathrm{z}^a)^+) + J_i^d(\mathrm{z}^d, \mathrm{u}^d) + J_i^{\text{avoid}}(\mathrm{z}^d, u_i^d, \mathrm{z}^{\text{obs}}) + J_i^{\text{mob}}(u_i^d). \tag{19}$$

To avoid notational clutter, we will ignore function arguments unless necessary. The terms comprising the cost function are

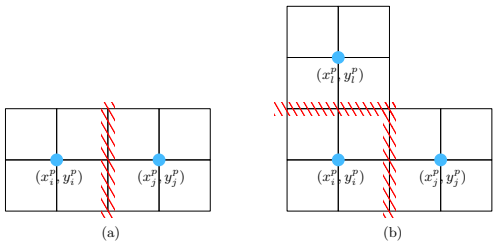


Fig. 2. Collision avoidance planes.

defined as follows.

$$J_i^f = \sum_{h=0}^{n_f-1} w_{ih}^f d((z_i^d)^+, z_h^f), \tag{20}$$

$$J_i^a = \sum_{g=0}^{n_a-1} w_{ig}^a d((z_i^d)^+, (z_g^a)^+), \tag{21}$$

$$J_i^d = \sum_{j=0}^{n_d-1} w_{ij}^d d((z_i^d)^+, (z_j^d)^+), \tag{22}$$

$$\begin{aligned} J_i^{\text{avoid}} = & \sum_{c_x \in X^d_{i,\text{avoid}}} d_{x,\text{avoid}}((z_i^d)^+, c_x) \\ & + \sum_{c_y \in Y^d_{i,\text{avoid}}} d_{y,\text{avoid}}((z_i^d)^+, c_y), \end{aligned} \tag{23}$$

$$J_i^{\text{mob}} = w_i^u(|u_{x,i}^d|^2 + |u_{y,i}^d|^2). \tag{24}$$

In the cost functions, w_{ih}^f, w_{ig}^a, w_{ij}^d, and w_i^u are non-negative weights. The function $d(z_i, z_j)$ is a distance measure between points z_i and z_j. It can be either of the following two functions:

$$\begin{aligned} & d(z_i, z_j) = (x_i - x_j)^2 + (y_i - y_j)^2, \text{ or} \\ & d(z_i, z_j) = |x_i - x_j| + |y_i - y_j|. \end{aligned} \tag{25}$$

The functions $d_{x,\text{avoid}}$ and $d_{y,\text{avoid}}$ are

$$d_{x,\text{avoid}}(z_i^d, c_x) = \zeta_1 e^{-\zeta_2(x_i^d - c_x)^2}, \text{ and}$$
$$d_{y,\text{avoid}}(z_i^d, c_y) = \zeta_1 e^{-\zeta_2(y_i^d - c_y)^2}$$

with $\zeta_1 \geq 1$ and $\zeta_2 \geq 1$.

The local cost of each defender has five components, each of which is a potential function with the minimum value at the desired location. The term J_i^f in (20) models defensive behavior in which d_i stays close to the defense zone to protect it. The constant weight $w_{ih}^f \geq 0$ is the strength of attractive force between d_i and the point z_h^f in the defense zone. The cost term J_i^f is minimized when z_i^d is equal to a weighted average of the points in the defense zone. We assume that each defender d_i is assigned the responsibility of a subset D_i of the defense zone D, where

$$D_i \subseteq D \quad \text{and} \quad \bigcup_{i=1}^{n_d} D_i = D.$$

The weights are assigned as follows.

$$w_{ih}^f = \begin{cases} \frac{1}{|D_i|} & z_h^f \in D_i, \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2019.2933993, IEEE Transactions on Control of Network Systems

The term J_i^a defined in (21) models attacking behavior of the defenders. In this mode, the defenders actively pursue the attackers and try to capture them before they reach the defense zone. The function J_i^a is a weighted sum of square of the distances between d_i and the locations of the attackers at the next time step. For the simulations in the next section, we assume that defender d_i only pursues the attacker that is closest to D_i. Let

$$\delta(D_i, a_g) = \min\{d(z_h^f, z_g^a) : z_h^f \in D_i\}, \text{ and}$$
$$\delta(D_i) = \min\{\delta(d_i, a_g) : g \in \{1, \dots, n_a\}\}.$$

Here $\delta(D_i, a_g)$ is the minimum Manhattan distance of attacker a_g from D_i and $\delta(D_i)$ is the minimum of the distances of all the attackers from D_i. Then

$$w_{ig}^a = \begin{cases} c & \delta(D_i, a_g) = \delta(D_i), \\ 0 & \text{otherwise.} \end{cases} \tag{27}$$

where c is a scalar. In case of a tie, d_i selects an attacker randomly and starts pursuing it.

The behavior of each defender can be selected to be a combination of these two terms by tuning the parameters α_i^a and α_i^f such that

$$\alpha_i^a + \alpha_i^f = 1.$$

The values $\alpha_i^a = 1$ or $\alpha_i^f = 1$ corresponds to purely attacking or defensive behaviors for d_i. If these parameters are constant, the behavior of the defenders remain the same through out the game. We can also have an adaptive strategy based on feedback for adjusting the behavior of each defender. Let δ_{th} be a threshold distance and let α_{nom}^a and α_{nom}^f be the nominal weights assigned to J_i^a and J_i^f respectively at $\delta(D_i) = \delta_{\text{th}}$ such that

$$\alpha_{\text{nom}}^a + \alpha_{\text{nom}}^f = 1.$$

Then

$$\alpha_i^a = \frac{\alpha_{\text{nom}}^a e^{\beta(\delta_{\text{th}} - \delta(D_i))}}{\alpha_{\text{nom}}^a e^{\beta(\delta_{\text{th}} - \delta(D_i))} + \alpha_{\text{nom}}^f}, \tag{28}$$
$$\alpha_i^f = \frac{\alpha_{\text{nom}}^f}{\alpha_{\text{nom}}^a e^{\beta(\delta_{\text{th}} - \delta(D_i))} + \alpha_{\text{nom}}^f}.$$

where $\beta \in [0, 1]$ is a constant value.

For defender d_i, if $\delta(D_i) = \delta_{\text{th}}$, the parameters α_i^a and α_i^f are equal to their nominal values. If an attacker gets closer to D_i than δ_{th}, i.e., $\delta(D_i) < \delta_{\text{th}}$, the value of α_i^a increases exponentially and the value of α^f decreases. Thus, as the attackers move towards D_i, the weight assigned to J_i^a increases, and the behavior of d_i shifts towards attacking mode . However, if the attackers are not close to the D_i, i.e., $\delta(D_i) > \delta_{\text{th}}$, then the value of α_i^a reduces exponentially and the behavior of d_i becomes more defensive.

The third term J_i^d defined in (22) generates cohesion among the defenders. Minimizing J_i^d drives d_i towards the weighted average of the locations of all the other defenders at time $k+1$. We assume that the weights w_{ih}^d are positive, i.e.,

$$w_{ih}^d > 0 \text{ for all } i, h \text{ in } \{1, 2, \dots, n_d\}.$$

The function J_i^d depends on the next locations of all the defenders. We assume that each defender knows that current locations of all of its teammates. However, it does not know the behavior parameters of other defenders, i.e., α_i^a and α_i^f are private parameters of each player. Therefore, we need to implement a distributed optimization algorithm to minimize J_i^d. We will show through simulations that the proposed algorithm Alg. 2 can be used effectively to minimize J_i^d.

The fourth term J_i^{avoid} defined in (23) guarantees obstacle and collision avoidance by generating repulsion from the avoidance planes. The function $d_{x,\text{avoid}}((x_i^d)^+, c_x)$ is maximum when $(x_i^d)^+ = c_x$, where $c_x \in X_{i,\text{avoid}}^d$. Similarly, $d_{y,\text{avoid}}((y_i^d)^+, c_y)$ is maximum when $(y_i^d)^+ = c_y$, where $c_y \in Y_{i,\text{avoid}}^d$. By selecting ζ_1 large enough, we can guarantee that d_i avoids the planes in $X_{i,\text{avoid}}^d$ and $Y_{i,\text{avoid}}^d$, which ensures that it avoids $R_{i,\text{avoid}}^d$. Thus, minimizing (23) guarantees collision and obstacle avoidance. The purpose of ζ_2 is to control the region of influence of this barrier potential. Finally, the fifth term J_i^{mob} in (24) is the mobility cost of d_i.

Problem ($\mathcal{P}2$) is a combinatorial optimization problem because the set of inputs is discrete. We will now prove that the cost J in (19) is submodular and ($\mathcal{P}2$) is a submodular minimization problem.

Theorem 5.1: *Problem ($\mathcal{P}2$) with the cost function defined in (19)-(24) is a submodular minimization problem over*

$$\mathcal{U} = \prod_{i=0}^{n_d - 1} U \times U,$$

where $U = \{-u_{\max}, -u_{\max} + 1, \dots, 0, \dots, u_{\max} - 1, u_{\max}\}$.

Proof: Since U is a subset of $\mathbb{Z}$, we can use the criterion in (3) to verify submodularity of the the cost function. From (19), the cost of each agent J_i is a summation of five terms. We will show that each of these terms is submodular. Then, using the property that sum of submodular functions is also submodular, we prove the theorem.

The terms J_i^f, J_i^a, and J_i^d are weighted sums of the distance functions in (25). We verify that

$$d(\mathrm{z}) = (x_i - x_j)^2 + (y_i - y_j)^2,$$

is submodular for any $\mathrm{z} = (x_i, y_i, x_j, y_j)$. To prove that $d(\mathrm{z})$ is submodular, we need to show that

$$d(\mathrm{z} + \mathrm{e}_p + \mathrm{e}_q) - d(\mathrm{z} + \mathrm{e}_q) \leq d(\mathrm{z} + \mathrm{e}_p) - d(\mathrm{z}).$$

where e_p and e_q are unit vectors of dimension four.

The first scenario is that both e_p and e_q increment either the x or the y components in z. Without loss of generality, we assume that the x components are incremented, i.e., $p = 1$ and $q = 3$. Then,

$$[d(\mathrm{z} + \mathrm{e}_p + \mathrm{e}_q) - d(\mathrm{z} + \mathrm{e}_q)] - [d(\mathrm{z} + \mathrm{e}_p) - d(\mathrm{z})] =$$
$$- (1 - 2(x_i - x_j)) - (1 + 2(x_i - x_j)) = -2.$$

The second scenario is that out of p and q, one corresponds to an x component and the other corresponds to a y component. Let $p = 2$ and $q = 3$. Then,

$$[d(\mathrm{z} + \mathrm{e}_p + \mathrm{e}_q) - d(\mathrm{z} + \mathrm{e}_q)] - [d(\mathrm{z} + \mathrm{e}_p) - d(\mathrm{z})] =$$
$$(1 + 2(y_i - y_j)) - (1 + 2(y_i - y_j)) = 0.$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2019.2933993, IEEE Transactions on Control of Network Systems

Thus, the condition in (3) is satisfied for all possible scenarios, which proves that the function $d(\mathrm{z})$ is submodular. The same sequence of steps can be followed to verify the submodularity of Manhattan distance.

The function J_i^{avoid} is the summation of the terms $d_{x,\text{avoid}}$ and $d_{y,\text{avoid}}$. Since each of these terms is only a function of single decision variable, the second order comparison condition in (3) will be satisfied with equality. The same argument is valid for J_i^{mob}. Since all the functions in J_i are submodular, J_i is submodular for all i, which concludes the proof. ∎

Since ($\mathcal{P}2$) is a submodular minimization problem, we can use our proposed framework for distributed submodular minimization to find an optimal plan for the defenders. In this setup of capture the flag game, the defense team needs to solve ($\mathcal{P}2$) at each decision time as depicted in Alg. 3. The cost of each defender in ($\mathcal{P}2$) depends on z^d, z^f, z^{obs}, $(\mathrm{z}^a)^+$ and u^d. The first three terms correspond to the current locations of all the defenders, location of the defense zone, and the locations of obstacles, all of which are known to each defender. The fourth term corresponds to the locations of the attackers in the next time step, which each defender computes based on the assumed model for attackers. The final term is the decision of all the defenders, which is not known and cannot be computed locally. Therefore, each defender has to minimize its cost J_i over the set of decision variables u^d.

In Alg. 3, defender d_i runs Alg. 2 for fixed number of iterations iter. The output of Alg. 2 is $\hat{\mathrm{u}}^{d_i}$, which is d_i's estimate of the decision vector u^d. Because of real-time constraints, the number of iterations iter has to be a small value. Therefore, the estimates of the defenders of the decision vector are not guaranteed to be same. Defender d_i uses its own estimate of the optimal decision vector $\hat{\mathrm{u}}^{d_i}$ to update its location and the process is repeated.

Problem ($\mathcal{P}2$) includes important requirements for a class of motion planning problems in multiagent systems. We can generate collision free paths for multiple agents starting from given initial conditions to terminal conditions while avoiding any obstacles in the environment. Moreover, the online implementation presented in Alg. 3 provides the capability of handling unmodeled system dynamics and uncertainties in the environment. Thus, we can claim with confidence that submodular minimization is a natural paradigm for modeling motion coordination problems in multiagent systems over discrete state space. Moreover, the framework proposed in this work can effectively compute feasible motion plans in a distributed manner.

VI. Simulation

We simulated the capture the flag game with the following setup. The size of the grid was 20×20 and the game was played over a time interval of length $T = 40$. The defense zone was located at the top of the field. The number of players in both the teams was four, i.e., $n_d = n_a = 4$. There were six point obstacles placed in the field. The detailed layout of the field with the locations of the defense zone, attacker, defenders, and the obstacles is presented in Fig. 3(a).

Algorithm 3 Online Distributed Submodular Minimization

At each decision time k, d_i needs to perform the following steps:

1: **for** $k = 0$ to $T - 1$ **do**

2: Apply Alg. 2 for fixed number of iterations iter. The output of the algorithm is $\hat{\mathrm{u}}^{d_i}$.

3: Update the state

$$z_i^d(k+1) = z_i^d(k) + \hat{u}_i^{d_i}.$$

4: **end for**

The defense zone is the set of squares at the top of the field. The obstacles are represented by asterisks, attackers by diamonds, and defenders by circles. The responsibility set of each defender d_i was D_i and is shown in the figure.

The parameters in J_i^{avoid} were set as $\zeta_1 = 200$ and $\zeta_2 = 5$. The weights in (27) for J_i^a was $c = 20$ for each defender. For cohesion among the defenders in J_i^d, the following weights were used

$$W^d = \begin{bmatrix} 0.0 & 0.5 & 0.1 & 0.01 \\ 0.5 & 0.0 & 0.1 & 0.01 \\ 0.01 & 0.1 & 0.0 & 0.5 \\ 0.01 & 0.1 & 0.5 & 0.0 \end{bmatrix},$$

where $W_{ij}^d = w_{ij}^d$. To switch between attacking and defense modes, we selected $\alpha_{\text{nom}}^f = 0.9$ and $\alpha_{\text{nom}}^f = 0.1$ and $\beta = 0.7$. The simulations were performed with Manhattan distance for the function $d(z_i, z_j)$ as defined in (25) and for four different values of threshold distance, $\delta_{\text{th}} \in \{5, 10, 15, 20\}$. We also simulate a scenario with different value of δ_{th} for each defender.

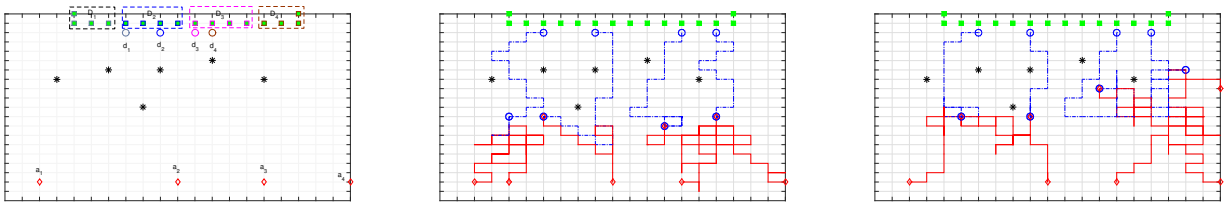
The defenders assumed that the attackers always try to minimize their distance form the defense zone. However, the actual strategy of the attackers was based on feedback that depended on their distance from the defenders. Each attacker had two basic modes: attack base and avoid defender. It adjusted the weights assigned to each of these modes depending on its minimum distance from the defenders.

At each decision time, attacker i computed two positions. To enter the defense zone, it computed the location in its neighborhood that minimized its distance from the defense zone. To avoid defenders, it also computed the location that maximized its distance from the nearest defender. Let $\eta_{i,\text{base}}$ be the weight assigned to attack base mode and $\eta_{i,\text{avoid}}$ be the weight assigned to avoid defender mode. Let $\eta_{\text{base}}^{\text{nom}}$ and $\eta_{\text{avoid}}^{\text{nom}}$ be the nominal values if the minimum distance between an attacker and the defenders was equal to some threshold value Δ_{th}. Let $\Delta_i^a(k)$ be the minimum distance between attacker i and the defenders at time k. Then

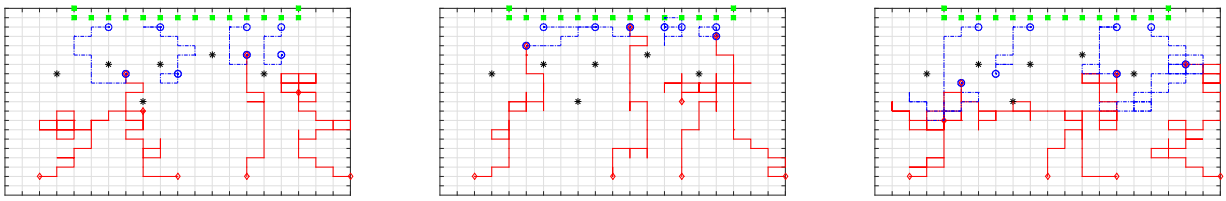
$$\eta_{i,\text{avoid}}(k) = \frac{\eta_{\text{avoid}}^{\text{nom}} e^{\kappa(\Delta_{\text{th}} - \Delta_i^a(k)}}{\eta_{\text{base}}^{\text{nom}} + \eta_{\text{avoid}}^{\text{nom}} e^{\kappa(\Delta_{\text{th}} - \Delta_i^a(k)}},$$
$$\eta_{i,\text{base}}(k) = \frac{\eta_{\text{base}}^{\text{nom}}}{\eta_{\text{base}}^{\text{nom}} + \eta_{\text{avoid}}^{\text{nom}} e^{\kappa(\Delta_{\text{th}} - \Delta_i^a(k)}}.$$

where $\kappa \in [0, 1]$ is a constant value. If $\Delta_i^a(k) < \Delta_{\text{th}}$, the value of $\eta_{i,\text{avoid}}(k)$ increases because a defender is closer than

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2019.2933993, IEEE Transactions on Control of Network Systems



(a) Playing arena. (b) $\delta_{\text{th}} = 20$. (c) $\delta_{\text{th}} = 15$.



(d) $\delta_{\text{th}} = 10$. (e) $\delta_{\text{th}} = 5$. (f) $\delta_{\text{th}} = (20, 8, 8, 20)$.

Fig. 3. Layout of the playing arena and trajectories of the offense and defense teams for different values of δ_{th} in (28).

the threshold value. However, as $\Delta_i^a(k)$ increases, $\eta_{i,\text{avoid}}(k)$ keeps on decreasing. Thus, at each decision time k, the attacker decides to attack the base with probability $\eta_{i,\text{base}}(k)$ and avoid defenders with probability $\eta_{i,\text{avoid}}(k)$. In the simulation setup, we selected $\eta_{\text{avoid}}^{\text{nom}} = 0.7$, $\eta_{\text{base}}^{\text{nom}} = 0.3$, $\Delta_{\text{th}} = 4$ and $\kappa = 0.9$.

Finally, for the distributed optimization algorithm, we assume that the communication network topology is a line graph and the adjacency matrix has the following structure.

$$A = \begin{bmatrix} 0.7 & 0.3 & 0 & 0 \\ 0.3 & 0.6 & 0.1 & 0 \\ 0 & 0.1 & 0.6 & 0.3 \\ 0 & 0 & 0.3 & 0.7 \end{bmatrix}.$$

The distributed subgradient algorithm was executed for $\text{iter} = 20$ iterations with $\gamma = 0.1$ and $\hat{t} = 0.7$. The simulation results are presented for four values of δ_{th}, which controlled the transition of defenders behavior from defense to attack in (28). In all the simulations, the behavior of the attackers was aligned with the values set for $\eta_{\text{avoid}}^{\text{nom}}$ and $\eta_{\text{base}}^{\text{nom}}$. The attackers had more emphasis on avoiding the defenders than capturing the base. Consequently, none of the attackers could enter the defense zone. However, the defenders were unable to capture all the attackers as well.

The effect of decreasing the value of δ_{th} can be observed by comparing the trajectories in Figs. 3(b)-3(e). With $\delta_{\text{th}} = 20$, the behavior of the defense team was set to be attacking, which is evident form Fig. 3(b). The defenders left the base in pursuit of the attackers and were able to capture three of them. As δ_{th} is reduced, the defensive behavior becomes more and more prominent. In Fig. 3(c) for $\delta_{\text{th}} = 15$, the defenders left the base area in pursuit of the attackers but were a little restrictive then the case with $\delta_{\text{th}} = 20$ in Fig. 3(b). The behavior of the defenders became more restrictive in Fig. 3(d) when $\delta_{\text{th}} = 10$. With $\delta_{\text{th}} = 5$, the behavior of the defenders was set to be defensive. Therefore, we can observe from Fig. 3(e) that all the defenders remained close to their assigned base area. Finally, we simulated the game with different δ_{th} for each defenders. From Fig. 3(f), we can observe that the defenders at the flanks were attacking and the center players were more defensive and guarded the defense zone.

In all the simulations, we can observe that the defenders managed to avoid collisions among themselves and with the obstacles. Thus, the proposed framework for the distributed minimization of submodular functions generated effective trajectories for our problem even though our problem was defined over partially ordered sets.

VII. CONCLUSION

We presented a framework for the distributed minimization of submodular functions over lattices of chain products. For this framework, we established a novel connection between a particular convex extension of submodular functions and distributed optimization of convex functions. This connection proved to be effective because that convex extension of submodular functions could be computed efficiently in polynomial time. Furthermore, the solution to the original submodular problem was directly related to the solution of the equivalent convex problem.

We also proposed a novel application domain for submodular function minimization, which is distributed motion coordination over discrete domains. We demonstrated through an example setup that we can design potential fields over state space based on submodular functions. We showed that we can achieve certain desired behaviors like cohesion, go to goal, collision avoidance, and obstacle avoidance by driving the agents towards the minima of these potential fields. Finally, we verified through simulations that the proposed framework for distributed submodular minimization can efficiently minimize these submodular potential fields online in a distributed manner.

REFERENCES

[1] S. T. McCormick, "Submodular function minimization," *Handbooks In Operations Research And Management Science*, vol. 12, pp. 321–391, 2005.

[2] J. Edmonds, "Submodular functions, matroids, and certain polyhedra," *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen*, vol. 11, 1970.

[3] D. M. Topkis, "Minimizing a submodular function on a lattice," *Operations research*, vol. 26, no. 2, pp. 305–321, 1978.

[4] ——, "Equilibrium points in nonzero-sum n-person submodular games," *Siam Journal on control and optimization*, vol. 17, no. 6, pp. 773–787, 1979.

[5] ——, *Supermodularity and Complementarity*. Princeton university press, 2011.

[6] J. Vondrák, "Optimal approximation for the submodular welfare problem in the value oracle model," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM, 2008, pp. 67–74.

[7] J. R. Marden and A. Wierman, "Distributed welfare games," *Operations Research*, vol. 61, no. 1, pp. 155–168, 2013.

[8] F. R. Bach, "Structured sparsity-inducing norms through submodular functions," in *Advances in Neural Information Processing Systems*, 2010, pp. 118–126.

[9] P. Stobbe and A. Krause, "Efficient minimization of decomposable submodular functions," in *Advances in Neural Information Processing Systems*, 2010, pp. 2208–2216.

[10] T. H. Summers, F. L. Cortesi, and J. Lygeros, "On submodularity and controllability in complex dynamical networks," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 1, pp. 91–101, 2016.

[11] A. Clark, B. Alomair, L. Bushnell, and R. Poovendran, *Submodularity in Dynamics and Control of Networked Systems*. Springer, 2016.

[12] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 199–208.

[13] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2014, pp. 946–957.

[14] J. R. Marden and A. Wierman, "Overcoming the limitations of utility design for multiagent systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 6, pp. 1402–1415, 2013.

[15] A. Nedic and A. Ozdaglar, "10 cooperative distributed multi-agent," *Convex Optimization in Signal Processing and Communications*, vol. 340, 2010.

[16] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause, "Distributed submodular maximization: Identifying representative elements in massive data," in *Advances in Neural Information Processing Systems*, 2013, pp. 2049–2057.

[17] F. Chierichetti, R. Kumar, and A. Tomkins, "Max-cover in map-reduce," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 231–240.

[18] G. E. Blelloch, R. Peng, and K. Tangwongsan, "Linear-work greedy parallel approximate set cover and variants," in *Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures*. ACM, 2011, pp. 23–32.

[19] S. Lattanzi, B. Moseley, S. Suri, and S. Vassilvitskii, "Filtering: A method for solving graph problems in map-reduce," in *Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures*. ACM, 2011, pp. 85–94.

[20] F. Bach, "Submodular functions: From discrete to continuous domains," *Mathematical Programming*, pp. 1–41, 2018.

[21] L. Lovász, "Submodular functions and convexity," in *Mathematical Programming The State of the Art*. Springer, 1983, pp. 235–257.

[22] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Puschel, "D-admm: A communication-efficient distributed algorithm for separable optimization," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2718–2723, 2013.

[23] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[24] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[25] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[26] S. M. LaValle, *Planning Algorithms*. Cambridge university press, 2006.

[27] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH computer graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[28] R. D'Andrea and R. M. Murray, "The roboflag competition," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 1. IEEE, 2003, pp. 650–655.

[29] G. C. Chasparis and J. S. Shamma, "LP-based multi-vehicle path planning with adversaries," *Cooperative Control of Distributed Multi-Agent Systems*, pp. 261–279, 2008.

[30] H. Jaleel and J. Shamma, "Distributed submodular minimization and motion coordination over discrete state space," *arXiv preprint arXiv:1709.07379*, 2017.

Hassan Jaleel Hassan Jaleel is an Assistant Professor of the Department of Electrical Engineering at the Lahore University of Management Sciences (LUMS) in Lahore, Pakistan. He received his M.S. and Ph.D. degrees in Electrical and Computer Engineering (ECE) with specialization in Systems and Control from the Georgia Institute of Technology, Atlanta, GA, USA. Prior to joining LUMS, Jaleel was a Postdoctoral Research Fellow at the King Abdullah University of Science and Technology (KAUST). His research interests are in the areas of complex networks, game theory, stochastic geometry, and online distributed optimization. Typical application domains of his research are sensor networks, swarm robotics, and irrigation networks. Jaleel was a Fulbright scholar from 2009–2013 and is a member of IEEE.

Jeff Shamma Jeff S. Shamma is a Professor of Electrical Engineering at the King Abdullah University of Science and Technology (KAUST), where he is also the Director of the Center of Excellence for NEOM Research at KAUST and principal investigator of the Robotics, Intelligent Systems & Control laboratory (RISC). Shamma is the former Julian T. Hightower Chair in Systems & Control in the School of Electrical and Computer Engineering at Georgia Tech, and he has held faculty positions at the University of Minnesota, The University of Texas at Austin, and the University of California, Los Angeles. Shamma received a Ph.D. in systems science and engineering from MIT in 1988. He is the recipient of an NSF Young Investigator Award, the American Automatic Control Council Donald P. Eckman Award, and the Mohammed Dahleh Award, and he is a Fellow of the IEEE and of the IFAC (International Federation of Automatic Control). Shamma is currently the deputy editor-in-chief for the IEEE Transactions on Control of Network Systems, an associate editor of the IEEE Transactions on Robotics (effective 2019), and a Distinguished Lecturer of the IEEE Control Systems Society. His most recent research has been in decision and control for distributed multiagent systems and the related topics of game theory and network science, with applications to cyberphysical and societal network systems.