Distributed Submodular Minimization And Motion Coordination Over Discrete State Space

Hassan Jaleel and Jeff S. Shamma

***Abstract*—Submodular set-functions are extensively used in large-scale combinatorial optimization problems arising in complex networks and machine learning. While there has been significant interest in distributed formulations of convex optimization, distributed minimization of submodular functions has not received significant attention. Thus, our main contribution is a framework for minimizing submodular functions in a distributed manner. The proposed framework is based on the ideas of Lovász extension of submodular functions and distributed optimization of convex functions. The framework exploits a fundamental property of submodularity that the Lovász extension of a submodular function is a convex function and can be computed efficiently. Moreover, a minimizer of a submodular function can be computed by computing the minimizer of its Lovász extension. In the proposed framework, we employ a consensus based distributed optimization algorithm to minimize set-valued submodular functions as well as general submodular functions defined over set products. We also identify distributed motion coordination in multiagent systems as a new application domain for submodular function minimization. For demonstrating key ideas of the proposed framework, we select a complex setup of the capture the flag game, which offers a variety of challenges relevant to multiagent system. We formulate the problem as a submodular minimization problem and verify through extensive simulations that the proposed framework results in feasible policies for the agents.**

I. Introduction

Submodular set 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. Moreover, submodularity is preserved under many natural operations and transformations as shown in [1]. However, unlike convex optimization in which distributed problem formulation and the design of efficient minimization algorithms is an active area of research (see e.g., [2] and the references therein), distributed minimization of submodular function has not received any research attention.

The main contribution of this work is a novel framework for the distributed minimization of a submodular function. One approach for minimizing a set function defined on the vertices of the unit hypercube is to formulate a relaxed problem in which the set function is replaced by its convex closure over the surface of the full hypercube. Then, the minimizers of the set function can be computed directly from the solution to the relaxed problem (see e.g. [3] and [4]). In general, computing the convex closure of a set function requires solving an optimization problem, which is computationally expensive. However, for submodular functions, the convex closure can be computed efficiently via a simple greedy heuristic that was proposed in [1] and is known as Lovász extension. In this work, we combine the ideas of Lovász extension of a submodular function and distributed minimization of convex functions for the first time and propose a novel framework for the distributed minimization of submodular set functions.

Submodular functions are popular in combinatorial optimization because they have numerous applications like resource allocation and welfare problem as in [5] and [6], large scale machine learning problems [7] and [8], and facility location and min-cut problems (see e.g., [9] and the references therein). Recently, submodular function maximization has been used extensively in multiagent systems for controllability of complex networks ([10] and [11]), influence maximization ([12] and [13]), and utility design for multiagent systems [14]. As a result, numerous combinatorial algorithms like [15], [16], [17], and [18] exist in the literature for distributed submodular maximization. However, there are no algorithms in the existing literature for distributed minimization of submodular functions.

Submodular function minimization has its roots in matroid theory for minimizing the rank function of a matroid, originally presented in [19]. It also plays a fundamental role in flow problems since max-flow min cut problem is a submodular minimization problem [20]. Moreover, the diminishing marginal returns property of a submodular function naturally models the concept of economies of scales and coalition formation. Thus, submodular minimization has been popular in economics and cooperative game theory as shown in [20] and [21]. In this work, we identify distributed motion coordination in multiagent system over discrete state space as a new application domain for submodular minimization. Distributed motion coordination and optimal path planning is an active area of research in multiagent systems. Most of the existing literature like [22], [23] and [24] focuses on these problems over continuous domains. In this work, we show that a large class of these problems can be formulated as a submodular minimization problem over discrete state space.

To plan the path of a single agent over a grid, classical search algorithms like A^* presented in [25] can be used effectively. However, as the number of agents or the size of the grid increase, the problem becomes intractable and centralized search algorithms have little practical significance. Numerous approaches have been presented for computing collision free paths for multiagent systems like computing paths offline with

Center for Electrical and Mechanical Science and Engineering, King Abdullah University of Science and Technology, Thuwal, KSA. Email: hassan.jaleel@kaust.edu.sa, jeff.shamma@kaust.edu.sa.

backup options in case of collision as in [26], formulating the problem as a dynamic network flow problem as presented in [27], and using computation fields as in [28].

In [29], a quantized consensus algorithm was presented for distributed averaging. This algorithm can be used to solve a rendezvous problem over a grid. In [30], motion coordination was proposed for multiagent systems that were assigned the task of locating a single agent in an indoor environment. The problem was formulated as an online optimization problem with a finite horizon in which the objective is to maximize a submodular set function.

We propose a unified approach for motion coordination over discrete space by showing that a large class of such problems can be formulated as submodular function minimization. In particular, we consider a version of capture the flag game presented in [31] and [32], which is played between two teams of mobile robots attack and defense. This game is selected because it has a complex setup with both collaborative and adversarial components, and it offers a variety of challenges that need to be addressed in typical problems in multiagent motion coordination. We formulate the problem from the perspective of defense team that is responsible for defending a defense zone in the playing arena from attack team. We show that this problem can be formulated as a submodular minimization problem and can be solved by applying the proposed framework. Formulating the problem as a submodular minimization problem allows us to effectively incorporate complex constraints and agent behaviors in the cost function. Finally, we verify through extensive simulations that the proposed framework results in a feasible policy for the defense team that enables it to effectively defend the defense zone.

II. Preliminaries

A. Notations

Let $S^{\text{val}} = \{s_0, s_1, \ldots, s_{m-1}\}$ be a finite set of labels with cardinality $|S^{\text{val}}|$ and indexed by $\mathbb{Z}_+$, where $\mathbb{Z}_+$ is the set of non-negative integers. A function f is a real valued set function on the set S^{val} if for each $A \subseteq S^{\text{val}}$, $f(A) \in \mathbb{R}$. For convenience and consistency of notation, we will always consider the index set $S = \{0, 1, \ldots, m-1\}$ of S^{val}. If $A \subseteq S$, $A = \{0, 1\}$ implies that the set A refers to the elements s_0 and s_1 of S^{val}. Similarly, $f(0)$ corresponds to function evaluation at s_0.

A vector $\mathrm{x} \in \mathbb{R}^n$ is represented as $\mathrm{x} = (x_0, x_1, \ldots, x_{n-1})$ and $\mathrm{x}(i)$ denotes its i^{th} value. We define $\{0,1\}^{|S|}$ as a 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 a 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 $i \in \{0, 1, \ldots, |S| - 1\}$ The indicator vector of a set $A \subseteq S$ is $\mathbb{1}_A$ and is defined as

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

For any two index vectors $\mathrm{x} = (x_0, x_1, \ldots, x_{n-1})$ and $\mathrm{y} = (y_0, y_1, \ldots, y_{n-1})$ in $\mathbb{R}^n$, $\mathrm{x} \geq \mathrm{y}$ if $x_i \geq y_i$ for all $i \in \{0, 1, \ldots, n-1\}$. Similarly, $\max(\mathrm{x}, \mathrm{y})$ and $\min(\mathrm{x}, \mathrm{y})$ are vectors in $\mathbb{R}^n$ defined as follows.

$$\max(x, y) = (\max(x_0, y_0), \max(x_1, y_1), \ldots, \max(x_n, y_n)),$$
$$\min(x, y) = (\min(x_0, y_0), \min(x_1, y_1), \ldots, \min(x_n, y_n)).$$

B. Overview of Submodularity

A set function $f : 2^S \to \mathbb{R}$ is submodular if and only if for any two subsets of S, say A and B, the following inequality holds

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B). \tag{2}$$

However, it is convenient to verify submodularity of a set function through the property of diminishing returns, which is as follows.

Definition 2.1: A function $f : 2^S \to \mathbb{R}$ is submodular if and only if for any two sets $A \subseteq S$ and $B \subseteq S$ such that $A \subseteq B$ and $k \in S \setminus B$

$$f(A \cup \{k\}) - f(A) \geq f(B \cup \{k\}) - f(B). \tag{3}$$

This property implies that the incremental increase in function value by adding an element in a small set is never smaller than the incremental increase of adding an element in a large set.

A set function $f : 2^S \to \mathbb{R}$ can be represented as a function on the vertices of the hypercube $\{0,1\}^{|S|}$. Without loss of generality, we will assume that $f(\phi) = 0$ for convenience. This representation is possible because each $A \subseteq S$ can be uniquely associated to a vertex of hypercube via indicator vector $\mathbb{1}_A$. An extension of f is a function defined on the hypercube $[0,1]^{|S|}$ such that it agrees with f on the vertices of the hypercube. One possible extension of a set function defined on the vertices of the hypercube is its convex closure [4]. Let D be the set of all distributions over 2^S and $\mathrm{E}_{A\sim d} f(A)$ be the expected value of f over 2^S, where the elements A are drawn from 2^S according to distribution d. Then the convex closure of f is defined as follows.

Definition 2.2: Let $f : \{0,1\}^{|S|} \to \mathbb{R}$ such that $f(\phi) = 0$. For every $\mathrm{x} \in [0,1]^{|S|}$, let $d_f(\mathrm{x})$ be a distribution over 2^S with marginal x such that

$$d_f(\mathrm{x}) = \underset{d \in D(\mathrm{x})}{\arg\min}\, E_{A\sim d} f(A) \tag{4}$$

where $D(x) \subseteq D$ is the set of all distributions with marginal x. Then the convex closure of function f at x is

$$f_{\text{cl}}(\mathrm{x}) = E_{A\sim d_f(\mathrm{x})} f(A)$$

It is important to highlight that the extension f_{cl} is convex for all set functions and does not require submodularity. However, computation of f_{cl} can be expensive because it involves solving the optimization problem in (4).

Example 1: Let $S = \{0, 1\}$, and for any $A \subseteq \mathcal{S}$

$$f(A) = \min\{|A|, 1\}.$$

The function f is equal to one everywhere except at $A = \phi$, where it is equal to zero. Let D be the set of distributions over

$$2^{\mathcal{S}} = \{\{0\}, \{1\}, \{0, 1\}, \phi\}$$

Suppose we want to compute f_{cl} at $\text{x} = [0.3 \;\; 0.5]$. One possible distribution is

$$d = [0.1 \;\; 0.3 \;\; 0.2 \;\; 0.4].$$

For d to be in $D(x)$, we need to show that its marginal is x. If the elements A are drawn from 2^S according to d, then

$$\Pr(0 \in A) = 0.1 + 0.2 = 0.3$$
$$\Pr(1 \in A) = 0.3 + 0.2 = 0.5$$

Thus, x is the marginal of the distribution d. To compute $f_{\text{cl}}(\text{x})$, we need to compute $d_f(\text{x})$ as defined in (4). For this particular example,

$$f_{\text{cl}}(\text{x}) = 0.5 \text{ at } \text{x} = [0.3 \;\; 0.5].$$

Therefore, each $\text{x}(i)$ can be considered as a probability measure over the set $\{0, 1\}$ for all $\text{x} \in [0, 1]^{|S|}$ and for all $i \in \{0, 1, \ldots, |S| - 1\}$.

Another approach for computing an extension of a set function was proposed in [1] and the resulting relaxed function is called the Lovász extension. The Lovász extension is defined with respect to submodular polyhedron.

Definition 2.3: Let $f : 2^S \to \mathbb{R}$ be a submodular function such that $f(\phi) = 0$. Then the submodular polyhedron $P(f)$ is

$$P(f) = \{s \in \mathbb{R}^{|S|} : s^T \mathbb{1}_A \leq f(A), \text{ for all } A \in 2^{|S|}\}.$$

The Lovász extension of the set function f at a point $\text{x} \in [0, 1]^{|S|}$ is

$$f_{\text{lv}}(\text{x}) = \max_{\text{s} \in P_f} \quad \text{s}^T \text{x} \tag{5}$$

It was proved in [1] that the Lovász extension f_{lv} is convex if and only if the function f is submodular. In fact, if f is submodular, the convex closure f_{cl} and the the Lovász extension f_{lv} are the same. Moreover, for a submodular function, instead of solving the linear program in (5), Lovász extension can be computed by a simple greedy algorithm as follows. Let $\text{x} = (x_0, x_1, \ldots, x_{|S|-1})$ be a vector in $[0, 1]^{|S|}$. Find a permutation $(i_1, i_2, \ldots, i_{|S|})$ of $(0, 1, \ldots, |S| - 1)$ such that $(x_{i_1} \geq x_{i_2} \geq \ldots \geq x_{i_{|S|}})$. Then

$$f_{\text{lv}}(\text{x}) = \sum_{k=1}^{|S|-1} f(\{i_1, \ldots, i_k\})(x_{i_k} - x_{i_{k+1}}) + f(S) x_{i_{|S|}}$$

For the problem in Example 1 with $\text{x} = [0.3 \;\; 0.5]$, the desired permutation is $(i_1, i_2) = (1, 0)$ since $x_1 > x_0$. Thus, the Lovász extension is

$$f_{\text{lv}}(\text{x}) = f(\{1\})(0.5 - 0.3) + f(\{0, 1\})0.3 = 0.5$$

The main result regarding submodular minimization, as proved in Prop. 3.7 of [3], is that minimizing the Lovász extension f_{lv} of a set function f on $[0, 1]^{|S|}$ is the same as minimizing it on $\{0, 1\}^{|S|}$, which is the same as minimizing f on $2^{|S|}$. In other words, the following three problems are equivalent.

$$\min\{f(A) : A \subseteq S\}$$
$$\min\{f(X) : X \in \{0, 1\}^{|S|}\}$$
$$\min\{f_{\text{lv}}(\text{x}) : \text{x} \in [0, 1]^{|S|}\}$$

This equivalence implies that a submodular minimization problem in discrete domain can be solved exactly by solving a convex problem in continuous domain for which efficient algorithms exist like sub-gradient based algorithms. Furthermore, let $\partial f_{\text{lv}}\big|_{\text{x}}$ be a subgradient vector of f_{lv} evaluated at x. The k^{th} component of f_{lv} can be computed via a greedy algorithm as follows.

$$\partial f_{\text{lv}}(k)\big|_{\text{x}} = f(\{i_1, \ldots, i_k\}) - f(\{i_1, \ldots, i_{k-1}\}).$$

C. General Submodular Functions

The notion of submodularity is not restricted to set valued functions. Submodular functions can be defined more generally on set products. In this work, we are interested in real-valued functions defined on a product of finite sets like $f : \mathcal{X}^{\text{val}} \to \mathbb{R}$, where

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

We assume that X_i^{val} is a finite subset of $\mathbb{R}$ for all $i \in \{0, 1, \ldots, N - 1\}$, and each $\text{x}^{\text{val}} \in \mathcal{X}^{\text{val}}$ is an N dimensional vector in $\mathbb{R}^N$. Let m_i be the number of elements in X_i^{val} and

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

be the index set of X_i^{val}. We define the product of index sets as

$$\mathcal{X} = \prod_{i=0}^{N-1} X_i. \tag{6}$$

For each vector $\text{x}^{\text{val}} \in \mathcal{X}^{\text{val}}$, there is a unique index vector

$$\text{x} = (x_0, x_1, \ldots, x_{N-1}) \in \mathcal{X},$$

where x_i is the index of the element from X_i^{val} in x^{val}. For notational convenience, we assume that the function f is defined on the index set, i.e., $f : \mathcal{X} \to \mathbb{R}$.

We will consider submodular functions defined over a lattice. A set $\mathcal{X}$ is a lattice if

$$\text{x and y} \in \mathcal{X} \implies \max\{\text{x}, \text{y}\} \in \mathcal{X} \text{ and } \min\{\text{x}, \text{y}\} \in \mathcal{X}.$$

A set $\mathcal{A}$ is a sublattice of $\mathcal{X}$ if $\mathcal{A} \subseteq \mathcal{X}$, and it is a lattice. Every finite subset of $\mathbb{R}$ is a sublattice of $\mathbb{R}$. The product of sublattices is also a sublattice [9].

Definition 2.4: Let f be a real valued function defined on a sublattice $\mathcal{X}$ of $\mathbb{R}^N$. Then f is submodular if and only if for any pair of vectors $(\text{x}, \text{y}) \in \mathcal{X} \times \mathcal{X}$,

$$f(\text{x}) + f(\text{y}) \geq f(\max(\text{x}, \text{y})) + f(\min(\text{x}, \text{y})). \tag{7}$$

As with set valued functions, to verify submodularity of a given function, we need a criterion based on the property of diminishing returns. For set products, this property is defined in terms of antitone differences. A function $f : \mathcal{X} \to \mathbb{R}$ is antitone on $\mathcal{X}$ if for any $\text{x} \in \mathcal{X}$ and $\text{y} \in \mathcal{X}$

$$\text{x} \leq \text{y} \implies f(\text{x}) \geq f(\text{y})$$

Definition 2.5: Let $f(\mathrm{x},\mathrm{y})$ be a real valued function on $\mathcal{X} \subset \mathbb{R}^{n+m}$ such that $\mathrm{x} \in \mathbb{R}^n$ and $\mathrm{y} \in \mathbb{R}^m$ and $N = n + m$. If

$$f(\mathrm{x},\mathrm{w}) - f(\mathrm{x},\mathrm{y})$$

is antitone in x on $\mathcal{X}$ for all w and y in $\mathbb{R}^m$ such that (x,w) and (x,y) belong to $\mathcal{X}$ and $\mathrm{w} \geq \mathrm{y}$, then $f(\mathrm{x},\mathrm{y})$ has antitone differences in (x,y).

Thus, if x_1 and x_2 are in $\mathbb{R}^n$ such that $\mathrm{x}_1 \geq \mathrm{x}_2$, then f has antitone differences if

$$f(\mathrm{x}_1,\mathrm{w}) - f(\mathrm{x}_1,\mathrm{y}) \leq f(\mathrm{x}_2,\mathrm{w}) - f(\mathrm{x}_2,\mathrm{y}).$$

The conditions in Def. 2.5 do not distinguish between first and second variables because we can easily rearrange the above inequality as follows

$$f(\mathrm{x}_1,\mathrm{w}) - f(\mathrm{x}_2,\mathrm{w}) \leq f(\mathrm{x}_1,\mathrm{y}) - f(\mathrm{x}_2,\mathrm{y})$$

Definition 2.6: A function $f : \mathcal{X} \to \mathbb{R}$ has antitone differences on $\mathcal{X} \subset \mathbb{R}^N$ if it has antitone differences on all $(\mathrm{x}(j),\mathrm{x}(l))$ with $j \neq l$ given $\mathrm{x}(i)$ is fixed for all $i \notin \{j,l\}$.

A set X is a chain if for any x and y in X, either $\mathrm{x} \leq \mathrm{y}$ or $\mathrm{x} \geq \mathrm{y}$. Every subset of $\mathbb{R}$ is a chain. Based on the notion of antitone differences, submodularity on product sets was defined in [9] as follows.

Theorem 2.1: If X_i is a chain for $i = 0,\ldots,N-1$ and f has antitone differences on $\mathcal{X} = \prod_{i=0}^{N-1} X_i$, then f is submodular on $\mathcal{X}$.

Theorem 2.1 implies that if f is defined over $\mathbb{R}^N$ instead of being restricted to $\mathcal{X}$ and is twice differentiable on $\mathbb{R}^N$, then f is submodular if and only if

$$\frac{\partial^2 f}{\partial \mathrm{x}(j)\partial \mathrm{x}(i)}(\mathrm{x}) \leq 0, \tag{8}$$

i.e., all the cross second order derivatives are non-positive for all $i \neq j$ and $\mathrm{x} \in \mathbb{R}^N$ (see [9] for further details). In [33], it was shown that all the nice properties of submodular functions like efficient minimization via Lovász extension and greedy algorithm for computing the Lovász extension can be extended to general submodular functions defined on product sets like $\mathcal{X}^{\text{val}}$. These results significantly increase the scope of submodular minimization especially in the context of multiagent motion coordination over a discrete state space. In a forthcoming section, a multiagent motion coordination problem will be formulated such that X_i^{val} represents the state space of agent i.

The first step is to compute an extension of submodular functions defined on set products over a continuous space. For simple set functions, each $\mathrm{x}(i)$ was visualized as a probability measure on the set $\{0,1\}$. In the case of set products where each set contains more than two elements, the notion of a probability measures needs to be extended.

Let $\mu_{X_i} = (\mu_{X_i}(0), \mu_{X_i}(1), \ldots, \mu_{X_i}(m_i - 1))$ be a probability measure on the set $X_i = \{0,1,\ldots,m_i - 1\}$, i.e., for all $j \in X_i$,

$$0 \leq \mu_{X_i}(j) \leq 1, \qquad \sum_{j=0}^{m_i-1} \mu_{X_i}(j) = 1.$$

A probability measure μ_{X_i} is degenerate if $\mu_{X_i}(k) = 1$ for some k. Let $P(X_i)$ be the set of all probability measures on X_i and $P(\mathcal{X}) = \Pi_{i=1}^{N} P(X_i)$. For a set $X_i = \{0,1,\ldots,m_i - 1\}$, we define $F_{\mu_{X_i}} : X_i \to \mathbb{R}$ as

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

Thus, $F_{\mu_{X_i}}$ is similar to cumulative distribution function but is reverse of it. Since μ_{X_i} is a probability measure, $F_{\mu_{X_i}}(0)$ is always equal to one. Therefore, we will ignore $F_{\mu_{X_i}}(0)$ and only consider $m_i - 1$ values to reduce dimension of the problem. For a probability measure μ_{X_i} on set X_i, we define a vector $\rho_{\mu_{X_i}} \in [0,1]^{m_i-1}$ as

$$\rho_{\mu_{X_i}} = (F_{\mu_{X_i}}(1), F_{\mu_{X_i}}(2), \ldots, F_{\mu_{X_i}}(m_i - 1)) \tag{9}$$

For a product set $\mathcal{X}$, we define

$$\mu_{\mathcal{X}} = \prod_{i=0}^{N-1} \mu_{X_i} \text{ and } \rho_{\mu_{\mathcal{X}}} = \prod_{i=0}^{N-1} \rho_{\mu_{X_i}}$$

Since $F_{\mu_{X_i}}(l) \geq F_{\mu_{X_i}}(l+1)$ for all $l \in \{0,1,\ldots,m_i - 1\}$, $\rho_{\mu_{X_i}}$ is vector with non-increasing entries. The equality $\rho_{\mu_{X_i}}(l) = \rho_{\mu_{X_i}}(l+1)$ occurs if and only if $\mu_{X_i}(l) = 0$.

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

$$\theta_{\rho_{\mu_{X_i}}}(t) = \max\{l \in X_i : F_{\mu_{X_i}}(l) \geq t\}.$$

This implies that

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

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

$$\theta_{\rho_{\mu_{\mathcal{X}}}}(\mathrm{t}) = \prod_{i=0}^{N-1} \theta_{\rho_{\mu_{X_i}}}(t_i),$$

where $\mathrm{t} = (t_0, t_1, \ldots, t_{N-1})$. The greedy algorithm for computing a convex extension of f over a continuous space is presented in Algorithm 1. This algorithm was originally presented in [33] and is included here for the completeness of presentation. The extension f^{ext} of f is given in (10) and the subgradient of f^{ext} is in (11).

III. Distributed Submodular Minimization

In this section, we present the main result of this work which is a distributed algorithm for minimizing a submodular function over a lattice. Consider a system comprised of N agents, $\{a_0, a_1, \ldots, a_{N-1}\}$. The state space of agent a_i has dimension n_i, i.e.,

$$\mathrm{x}_i^{\text{val}} \in \mathcal{X}_i^{\text{val}}, \text{ where } \mathcal{X}_i^{\text{val}} = \prod_{p=0}^{n_i-1} X_p^{\text{val}},$$

Algorithm 1 Greedy Algorithm

Require: Product probability measures $\mu = \prod_{i=0}^{N-1} \mu_{X_i}$.

1: Compute $\rho_\mu = \prod_{i=0}^{N-1} \rho_{\mu_{X_i}}$, where $\rho_{\mu_{X_i}}$ is defined in (9).

2: Form the set Q as follows.

$$Q = \{\rho_{\mu_{X_0}}(1), \ldots, \rho_{\mu_{X_0}}(m_0 - 1), \rho_{\mu_{X_1}}(1), \ldots, \rho_{\mu_{X_1}}(m_1 - 1), \ldots, \rho_{\mu_{X_{N-1}}}(1), \ldots, \rho_{\mu_{X_{N-1}}}(m_{N-1} - 1)\}.$$

The number of elements in Q is $r = \sum_{i=0}^{N-1} m_i - n$. In the summation, n is subtracted because $F_{\mu_{X_i}}(0)$ is neglected in $\rho_{\mu_{X_i}}$ for all i.

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

$$Q_{\text{dec}} = \{\rho_{\mu_{X_{i(1)}}}(j(1)), \rho_{\mu_{X_{i(2)}}}(j(2)), \ldots, \rho_{\mu_{X_{i(r)}}}(j(r))\},$$

where

$$\rho_{\mu_{X_{i(1)}}}(j(1)) \geq \rho_{\mu_{X_{i(2)}}}(j(2)) \geq \ldots \geq \rho_{\mu_{X_{i(r)}}}(j(r)).$$

The ties are handled randomly. However, in case of ties within $\rho_{\mu_{X_i}}$ for some i, the order of the values are maintained.

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

$$f^{\text{ext}}(\rho) = f(0) + \sum_{s=0}^{r-1} \mathrm{t}(s) \left(f(\mathrm{y}_s) - f(\mathrm{y}_{s-1}) \right), \quad (10)$$

where

$$\mathrm{t}(s) = \rho_{\mu_{X_{i(s)}}}(j(s)) \quad \text{for all } s \in \{0, 1, \ldots, r-1\}$$

The vector $\mathrm{y}_s \in \mathcal{X}$ is such that

$$\mathrm{y}_0 = (0, 0, \ldots, 0),$$
$$\mathrm{y}_{r-1} = (m_0 - 1, m_1 - 1, \ldots, m_{N-1} - 1),$$

and for any $s \in \{1, 2, \ldots, r-2\}$, $\mathrm{y}_s(i)$, the i^{th} value of y_s is

$$\mathrm{y}_s(i) = \theta_{\rho_{\mu_{X_i}}}(\tau) \qquad \tau \in (\mathrm{t}(s+1), \mathrm{t}(s))$$

5: The subgradient of f^{ext} evaluated at $\rho_{\mu_{\mathcal{X}}}$ is

$$\partial f^{\text{ext}}\big|_{\rho_{\mu_{\mathcal{X}}}} = \prod_{i=0}^{N-1} \partial f^{\text{ext}}_{\rho_{\mu_{X_i}}}.$$

The j^{th} component of $\partial f^{\text{ext}}_{\rho_{\mu_{X_i}}}$ is

$$\partial f^{\text{ext}}_{\rho_{\mu_{X_i}}}(j) = f(\mathrm{y}_g) - f(\mathrm{y}_{g-1}), \quad (11)$$

where $g = \min\{s \in \{0, 1, \ldots, r-1\} : \mathrm{y}_s(i) = j\}$.

and X_p^{val} is a finite subset of $\mathbb{R}$. We assume that the joint state space of all the agents, $\mathcal{X}^{\text{val}} = \prod_{i=1}^{N-1} \mathcal{X}_i^{\text{val}}$, is a lattice. As explained before, for convenience of notation, we assume that the state space of a_i is the index set

$$\mathcal{X}_i = \prod_{p=0}^{n_i - 1} X, \quad X = \{0, 1, \ldots, m-1\}.$$

Notice that the set X does not depend on the index p or i, i.e., the state space of all the agents is identical along all the dimensions. A probability measure on $\mathcal{X}_i$ is defined as

$$\mu_{\mathcal{X}_i} = \prod_{p=0}^{n_i - 1} \mu_{X,p}, \quad \mu_{X,p} \in P(X).$$

The joint state space of all the agents is $\mathcal{X} = \prod_{i=0}^{N-1} \mathcal{X}_i$. Each $\mathrm{x} \in \mathcal{X}$ is a vector in $\mathbb{R}^n$, where $n = \sum_{i=1}^{N} n_i$ and a probability measure on $\mathcal{X}$ is

$$\mu_{\mathcal{X}} = \prod_{i=0}^{N-1} \mu_{\mathcal{X}_i}$$

For a vector $\mathrm{x}_i \in \mathcal{X}_i$, we define $e(\mathrm{x}_i) = \prod_{p=0}^{n_i - 1} e_{x_p} \in P(\mathcal{X}_i)$ such that

$$e_{x_p}(k) = \begin{cases} 1 & \text{if } k = x_p, \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

Let $J_i : \mathcal{X} \to \mathbb{R}$ be the local cost of agent a_i. We assume that J_i is a submodular function for all $i \in \{0, 1, \ldots, N-1\}$. The objective is to solve the following optimization problem.

$$\min_{\mathrm{x} \in \mathcal{X}} \quad J(\mathrm{x}) \quad (13)$$

where

$$J(\mathrm{x}) = \sum_{i=0}^{N-1} J_i(\mathrm{x})$$

The cost function J is a sum of N submodular functions, so it is also a submodular function.

To solve this problem in a distributed manner, each agent needs to solve a local version of the global problem. Since J_i is a function of $\mathrm{x} \in \mathcal{X}$ for all $i \in \{0, 1, \ldots, N-1\}$, each agent requires the states of all the other agents in the network to compute a global solution. If all the agents can communicate with each other, the problem reduces to a centralized problem. In this setup, we assume that each agent can only communicate with a subset of other agents in the network. The communication network is represented by a graph $\mathbb{G}(V, \mathcal{E})$, where $V = \{a_0, a_1, \ldots, a_{N-1}\}$ is the set of vertices and $\mathcal{E} \subseteq V \times V$ is the set of edges. If $(a_i, a_j) \in \mathcal{E}$, it implies that agents a_i and a_j can communicate with each other. The closed neighborhood set of a_i contains a_i and the agents with which a_i can communication, i.e.,

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

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

$$A(i,j) = \begin{cases} c_{ij} & a_j \in N(a_i) \\ 0 & \text{otherwise} \end{cases}$$

where $c_{ij} \geq 0$ for all i and j. Following [34], we make following assumptions on network topology to ensure convergence of the distributed optimization algorithm.

1) The communication graph $\mathbb{G}(V, \mathcal{E})$ is connected.
2) There exists a scalar $\eta \in (0, 1)$ such that $c_{ii} \geq \eta$ for all i.
3) For any pair of agents $(a_i, a_j) \in \mathcal{E}$, $c_{ij} \geq \eta$.
4) Matrix A is row stochastic, i.e., $\sum_{i=0}^{N-1} c_{ij} = 1$ for all i and j in $\{0, 1, \ldots, N-1\}$

To solve the optimization problem, we will exploit the submodularity of the cost function J. It was shown in [33], that solving the problem in (13) is equivalent to solving the following problem.

$$\min_{\rho_{\mu_{\mathcal{X}}} \in \Omega_{\mathcal{X}}} \quad J^{\text{ext}}(\rho_{\mu_{\mathcal{X}}}) \tag{14}$$

where

$$J^{\text{ext}} = \sum_{i=0}^{N-1} J_i^{\text{ext}}(\rho_{\mu_{\mathcal{X}}})$$

is the extension of J and can be computed using (10) of the greedy algorithm. The constraint set

$$\Omega_{\mathcal{X}} = \prod_{i=0}^{N-1} \Omega_{\mathcal{X}_i},$$

where

$$\Omega_{\mathcal{X}_i} = \prod_{p=0}^{n_i - 1} \Omega_X, \text{ and}$$

$$\Omega_X = \{\omega \in [0,1]^{m-1} : \ \omega(k) \geq \omega(k+1) \forall 1 \leq k \leq m-2\}$$

Thus, $\rho_{\mu_{\mathcal{X}}} \in \Omega_{\mathcal{X}}$ implies that $\rho_{\mu_{X,p}} \in \Omega_X$ for all $p = \{0, 1, \ldots, n-1\}$, where $n = \sum_{i=0}^{N-1} n_i$.

The problem in (14) is a constrained convex optimization problem. For distributed minimization of this problem, we use consensus based projected subgradient algorithm in our proposed framework. In this algorithm, as presented in [34], each agent is only allowed to communicate with the agents in its neighborhood set. Since the cost of each agent depends on the states of all the other agents in the network, each agent maintains an estimate of the states of all the agents. The main idea is the use of consensus to ensure that the estimates of all the agents in the network converge to same values.

In Algorithm 2, a_i starts by initializing the state of each agent a_j in the network with some feasible values $\mathrm{x}_j^i \in \mathcal{X}_j$. The probability measure at time 0, $\mu^i_{\mathcal{X}_j}[0]$, is initialized with the degenerate measure $e(\mathrm{x}_j^i)$ and the corresponding vector $\rho_{\mu^i_{\mathcal{X}_j}}[0] = \rho_{e(\mathrm{x}_j^i)}$. To update the estimates $\rho_{\mu^i_{\mathcal{X}_j}}$ for all $j \in \{0, 1, \ldots, N-1\}$, a_i exchanges its estimates with all the agents in its neighborhood set $N(a_i)$. The estimates are updated in two steps, a consensus step and a gradient descent step. The consensus step is in (15) in which a_i computes a weighted combination of the estimates of $a_k \in N(a_i)$ by assigning weight c_{ik} to $\rho_{\mu^k_{\mathcal{X}_j}}$. The gradient descent step is in (16), 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_l is the step size. The gradient $\partial J_i^{\text{ext}}\big|_{\nu^i}$ is computed through the greedy algorithm.

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

$$\xi^i = \nu^i - \gamma_l \partial J_i^{\text{ext}}.$$

Here $\xi^i = \prod_{j=1}^{N-1} \xi_j^i$ such that each $\xi_j^i = \prod_{p=0}^{n_j - 1} \xi_{j,p}^i$. Thus, the projection of ξ^i on Ω can be decomposed into projecting each $\xi_{j,p}^i$ on X for which we solve the following quadratic program..

$$\min_{\rho \in [0,1]^{m-1}} \|\rho - \xi_{j,p}^i\|^2 \tag{17}$$
$$\text{s.t.} \qquad C\rho < 0_{m-2}$$

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

$$C_{uv} = \begin{cases} -1 & \text{if } u = v \\ 1 & \text{if } v = u+1 \\ 0 & \text{otherwise} \end{cases}$$

Algorithm 2 Distributed Submodular Minimization

To solve problem in (14), a_i performs the following steps:

Require: Select any $\mathrm{x}_j^i \in \mathcal{X}_j$ for all $j \in \{0, 1, \ldots, N-1\}$.

Set $\mu^i_{\mathcal{X}_j}[0] = e(\mathrm{x}_j^i)$ as defined in (12) and

$$\rho_{\mu^i_{\mathcal{X}_j}}[0] = \rho_{e(\mathrm{x}_j^i)}.$$

Update $\rho_{\mu^i_{\mathcal{X}}}$ as follows

for $l = 1$ to iter **do**

  for $j = 0$ to $N-1$ **do**

$$\nu_j^i = \sum_{k=0}^{N-1} c_{ik} \rho_{\mu^k_{\mathcal{X}_j}}[l-1] \tag{15}$$

  end for

$$\rho_{\mu^i_{\mathcal{X}}}[l] = \mathbb{P}_{\Omega_{\mathcal{X}}}(\nu^i - \gamma_l \partial J_i^{\text{ext}}\big|_{\nu^i}), \tag{16}$$

  where $\nu^i = \prod_{j=0}^{N-1} \nu_j^i$.

end for

Set $\bar{\rho}^i = \rho_{\mu^i_{\mathcal{X}}}[\text{iter}]$. Compute $\bar{\mu}^i_{\mathcal{X}} = \prod_{j=0}^{N-1} \bar{\mu}^i_{\mathcal{X}_j}$ as follows

for $j = 0$ to $N-1$ **do**

$$\bar{\mu}^i_{\mathcal{X}_j}(k) = \begin{cases} \theta_{\bar{\rho}^i_j}(\bar{\rho}^i_j(k)) & k \in \{1, 2, \ldots, m-1\} \\ 1 - \bar{\rho}^i_{j,p}(1) & k = 0 \end{cases}$$

  Agent a_i's estimate of $\mathrm{x}_j^*(p)$, $p \in \{0, 1, \ldots, n_j - 1\}$ is

$$\bar{\mathrm{x}}_j^i(p) = \text{argmax}\{\bar{\mu}^i_{\mathcal{X}_j,p}(k) : k \in \{0, 1, \ldots, m-1\}\}.$$

end for

Agent a_i's estimate of optimal x^* is

$$\bar{\mathrm{x}}^i = (\bar{\mathrm{x}}^0, \bar{\mathrm{x}}^1, \ldots, \bar{\mathrm{x}}^{N-1}).$$

The estimate $\rho_{\mu^i_{\mathcal{X}}}$ is updated through Eqs (15) and (16) for a fixed number of iterations "iter". In [34], it is shown that if the number of iterations is sufficiently large, then

$$\bar{\rho}^i \to \rho^*_{\mathcal{X}}, \text{ where } \bar{\rho}^i = \rho_{\mu^i_{\mathcal{X}}}[\text{iter}].$$

Here, $\rho^*_{\mathcal{X}}$ is the vector with respect to the probability distribution $\mu^*_{\mathcal{X}}$, which corresponds to an optimal solution x^*. Since the solution to the relaxed problem should be a minimizer for the combinatorial problem, the optimal probability measure $\mu^*_{\mathcal{X}}$ should be a degenerate measure corresponding to the optimal solution x^*. However, because of timing constraints, Algorithm 2 may be allowed a limited number of iterations. Thus, $\bar{\mu}^i_{\mathcal{X}}$ is not guaranteed to be degenerate, and as shown in steps 8 and10 of the algorithm, a_i's estimate of the optimal vector $\mathrm{x}^*_j(p)$ corresponds to the index of $\bar{\mu}^i_{\mathcal{X}_j,p}$ that has maximum probability.

IV. Distributed Motion Coordination

In this section, we develop the novel application of submodular minimization, which is distributed motion coordination in multiagent systems. We will see that the proposed framework can play a fundamental role in distributed motion coordination for multiagent systems over a discrete state space.

As outlined in the classical "boids" model in [35], 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 implies the ability of the agents to align with a desired orientation. In addition to these behaviors, agents should be able to avoid any obstacles in the environment. We will argue that all of these behaviors can be effectively modeled as a submodular minimization problem.

We present our arguments in the context of an example setup that is inspired from capture the flag game presented in [31] and [32]. Capture the flag is a challenging setup that involves two teams of agents competing against each other. The members of the same team need to collaborate with each other to devise a mobility strategy that can stop the opposing team from achieving their objective. Thus, the setup has both collaborative and adversarial components involved in decision making, which makes it a challenging problem even for simple cases. We want to highlight that our objective is not to provide a solution to the well studied capture the flag game. Instead, our objective is to show that submodular minimization can be used effectively for such complex motion coordination problems over a discrete state space. Therefore, we will tailor the setup of this game such that the players require all the behaviors of Reynolds' boids in order to complete their objective.

A. Problem Formulation

The 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 $\mathcal{G} = G_x \times G_y$, which is a finite subset of $\mathbb{R}^2$, i.e., each $z = (x, y)$ in $\mathcal{G}$ is a vector in $\mathbb{R}^2$, where x and y belong to $\{0, 1, 2, \ldots, N_g - 1\}$.

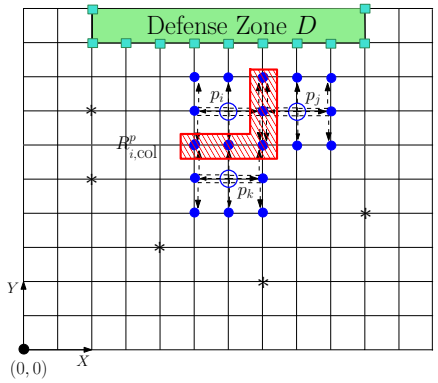


Fig. 1. Layout of the playing arena.

A flag is assumed to be placed in the arena and the area surrounding it is declared as a defense zone. The defense zone $D = \{z^f_0, z^f_1, \ldots, z^f_{n_f-1}\}$ is a subset of $\mathcal{G}$ with n_f points. The points in D are stacked in a vector $\mathrm{z}^f = (\mathrm{x}^f, \mathrm{y}^f)$, where

$$\mathrm{x}^f = (x^f_0, \ldots, x^f_{n_f-1}), \quad \mathrm{y}^f = (y^f_0, \ldots, y^f_{n_f-1}).$$

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 offense and defense teams respectively. Let n_p be the number of players in team $p \in P$. The locations of all the players in the team at time k are jointly represented by the vector $\mathrm{z}^p(k) = (\mathrm{x}^p(k), \mathrm{y}^p(k))$, where

$$\mathrm{x}^p(k) = (x^p_0(k), \ldots, x^p_{n_p-1}(k)),$$
$$\mathrm{y}^p(k) = (y^p_0(k), \ldots, y^p_{n_p-1}(k)).$$

The update equation for p_i is

$$z^p_i(k+1) = z^p_i(k) + u^p_{z,i}(k), \tag{18}$$

where $z^p_i(k) = (x^p_i(k), y^p_i(k))$ is the location of player p_i at

time k and $u^p_{z,i}(k) = (u^p_{x,i}(k), u^p_{y,i}(k)) \in U^p_x \times U^p_y$,

$$U^p_x \in \{0, \pm 1, \ldots, \pm u^{\max}_x\},$$
$$U^p_y \in \{0, \pm 1, \ldots, \pm u^{\max}_y\}.$$

The reachable set of p_i at time k is the set of all points it can reach in one time step.

$$R^p_i(k) = \{z \in \mathcal{G} : z = z^p_i(k) + u^p_{z,i}, \quad u^p_{z,i} \in U^p_x \times U^p_y\}.$$

The reachable sets of players with $u^{\max}_x = u^{\max}_y = 1$ are depicted in Fig. 1. Two players can collide if their reachable sets overlap with each other. Let

$$R^p_{i,\text{col}}(k) = \bigcup_{\substack{j=1 \\ j \neq i}}^{n_p} \left(R^p_i(k) \cap R^p_j(k)\right),$$

i.e., $R^p_{i,\text{col}}(k)$ is the set of all points in $R^p_i(k)$ that can result in a collision between p_i and the members of its team. The shaded region in Fig. 1 depicts $R^p_{i,\text{col}}$ for p_i. One approach to guarantee collision avoidance is to limit the effective reachable set of p_i to $R^p_i(k) - R^p_{i,\text{col}}(k)$, which is the set of points of $R^p_i(k)$ that are not included in $R^p_{i,\text{col}}(k)$.

Finally, to make the game more challenging and to add obstacle avoidance, we assume that some point obstacles are placed in the arena. Let

$$\mathrm{z}^{\text{obs}} = (\mathrm{x}^{\text{obs}}, \mathrm{y}^{\text{obs}})$$

be the vector of locations of all the obstacles. In Fig. 1, the obstacles are represented by asterisks.

In this work, we are interested in an optimal mobility policy for the defense team only. To compute an optimal mobility policy for the defense team over time interval $[0, T]$ in which defenders coordinate with each other to guard the defense zone and capture the attackers, we formulate a trajectory optimization problem.

$$\min_{\mathrm{u}^d_z(0),\ldots,\mathrm{u}^d_z(T-1)} \sum_{k=0}^{T-1} J(\mathrm{z}^d(k), \mathrm{u}^d_z(k), \mathrm{z}^a(k+1), \mathrm{z}^f, \mathrm{z}^{\text{obs}})$$
$$\text{s.t.} \qquad \mathrm{z}^d(k+1) = \mathrm{z}^d(k) + \mathrm{u}^d_z(k), \qquad (\mathcal{P})$$

where $\mathrm{u}^d_z(k) = (\mathrm{u}^d_x(k), \mathrm{u}^d_y(k))$ and

$$\mathrm{u}^d_x(k) = (u^d_{x,0}(k), u^d_{x,1}(k), \ldots, u^d_{x,n_d-1}(k)),$$
$$\mathrm{u}^d_y(k) = (u^d_{y,0}(k), u^d_{y,1}(k), \ldots, u^d_{y,n_d-1}(k)).$$

Problem $\mathcal{P}$ is a trajectory optimization problem in which the objective is to compute optimal trajectories for the defenders. In the game setup, we assume that at time k each defender knows the current locations of all the attackers. However, the cost function in problem $\mathcal{P}$ depends on the future locations of the attackers. Therefore, we need an online optimization strategy like model predictive control (MPC), in which the defenders assume a mobility model for attackers over a prediction horizon. To simplify the problem setup, we will use roll out algorithm with one step look ahead as presented in [36]. At each time k, the online optimization problem has the following structure.

$$\min_{\mathrm{u}^d_z} \quad J(\mathrm{z}^d, \mathrm{u}^d_z, (\mathrm{z}^a)^+, \mathrm{z}^f)$$
$$\text{s.t.} \quad (\mathrm{z}^d)^+ = \mathrm{z}^d + \mathrm{u}^d_z. \qquad (\mathcal{P}1)$$

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 problem $\mathcal{P}1$, defenders still need to know $(\mathrm{z}^a)^+$, which cannot be known at current time. Therefore, defenders assume a mobility model for attackers. One possible model for attackers is that they will always move towards the defense zone in a straight line. We will use this model for the simulations in the next section.

The cost function J in problem $\mathcal{P}1$ is

$$J(\mathrm{z}^d, \mathrm{u}^d_z, (\mathrm{z}^a)^+, \mathrm{z}^f, \mathrm{z}^{\text{obs}}) = \sum_{i=0}^{n_d-1} J_i(\mathrm{z}^d, \mathrm{u}^d_z, (\mathrm{z}^a)^+, \mathrm{z}^f, \mathrm{z}^{\text{obs}})$$
$$J_i(\mathrm{z}^d, \mathrm{u}^d_z, (\mathrm{z}^a)^+, \mathrm{z}^f, \mathrm{z}^{\text{obs}}) = J^{\text{obs}}_i(z^d_i, u^d_{z,i}, \mathrm{z}^{\text{obs}}) + J^{\text{avoid}}_i(\mathrm{z}^d)$$
$$+ J^d_i(\mathrm{z}^d, \mathrm{u}^d_z) + \alpha^f_i J^f_i(z^d_i, u^d_{z,i}, \mathrm{z}^f) + \alpha^a_i J^a_i(z^d_i, u^d_{z,i}, (\mathrm{z}^a)^+). \qquad (19)$$

where

$$\alpha^a_i + \alpha^f_i = 1.$$

The terms comprising the cost function are defined as follows.

$$J^{\text{obs}}_i(z^d_i, u^d_{z,i}, \mathrm{z}^{\text{obs}}) = \sum_{q=0}^{n_{\text{obs}}-1} d_{z,\text{obs}}((z^d_i)^+, z^{\text{obs}}_q), \qquad (20)$$

$$J^{\text{avoid}}_i(\mathrm{z}^d) = \sum_{z_r \in R^d_{i,\text{col}}} d_{z,\text{obs}}((z^d_i)^+, z_r) \qquad (21)$$

$$J^d_i(\mathrm{z}^d, \mathrm{u}^d_z) = \sum_{j=0}^{n_d-1} w^d_{ij} d_z((z^d_i)^+, (z^d_j)^+), \qquad (22)$$

$$J^f_i(z^d_i, u^d_{z,i}, \mathrm{z}^f) = \sum_{h=0}^{n_f-1} w^f_{ih} d_z((z^d_i)^+, z^{\text{f}}_h) \qquad (23)$$

$$J^a_i(z^d_i, u^d_{z,i}, (\mathrm{z}^a)^+) = \sum_{g=0}^{n_a-1} w^a_{ig} d_z((z^d_i)^+, (z^a_g)^+), . \qquad (24)$$

where

$$d_z(z_i, z_j) = d(x_i, x_j) + d(y_i, y_j),$$
$$= (x_i - x_j)^2 + (y_i - y_j)^2.$$

and

$$d_{z,\text{obs}}(z^d_i, z^d_j) = \zeta_1 e^{-\zeta_2 d_z(z^d_i, z^d_j)} \qquad (25)$$

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

The local cost of each defender has five components. All of these components are potential functions with minima at the desired locations. The first term J^{obs}_i defined in (20) takes care of obstacle avoidance. The function $d_{z,\text{obs}}(z^d_i, z^{\text{obs}}_q)$ is maximum when $z^d_i = z^{\text{obs}}_q$. By selecting ζ_1 large enough, it can be guaranteed that d_i always avoids the obstacles. The purpose of ζ_2 is to control the decay of this barrier potential

when $z_i^d \neq z_j^d$. The second term J_i^{avoid} defined in (21) ensures collision avoidance among the members of the defense team. This function generates high potentials at all the points in the set $R_{i,\text{col}}^d$, which is the set of points where defenders can collide. This function is also based on $d_{z,\text{obs}}$, which results in a high potential at a point that is in the reachable set of multiple defenders. By avoiding the points in $R_{i,\text{col}}^d$, collision avoidance is guaranteed. 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 all the other defenders. We assume that the weights w_{ih}^d are positive, i.e.,

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

The fourth and fifth terms jointly model the behavior of a defender. The term J_i^f defined in (23) models defensive behavior of defender d_i in which it 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^{flag} 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

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

The weights are assigned as follows.

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

The term J_i^a defined in (24) 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

$$\delta(d_i, a_g) = \min\{\sqrt{d_z(z_g^a, z_h^f)} : z_h^f \in D_i\}$$
$$\delta(d_i) = \min\{\delta(d_i, a_g) : g \in \{1, \ldots, n_a\}\}.$$

Here $\delta(d_i, a_g)$ is the minimum Euclidean 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

$$w_{ig}^a = \begin{cases} 1 & \delta(d_i, a_g) = \delta(d_i) \\ 0 & \text{otherwise} \end{cases} \tag{27}$$

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 at which the behavior of a defender switches between attack and defense modes. Let α_{nom}^a and α_{nom}^f be the nominal weights assigned to J_i^a and J_i^f respectively at $\delta(d_i, k) = \delta_{\text{th}}$ 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^a reduces exponentially and the behavior of d_i becomes more defensive.

Problem $\mathcal{P}1$ 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 $\mathcal{P}1$ is a submodular minimization problem.

Theorem 4.1: The cost in problem $\mathcal{P}1$ as defined in (19)-(24) is a submodular function.

Proof: To prove that $J(\mathrm{z}^d, \mathrm{u}_z^d, (\mathrm{z}^a)^+, \mathrm{z}^f)$ is a submodular function of the decision vector u_z^d, we will use Thm. 2.1. Since J is well-defined over the entire planar region, not just on a discrete set, and is twice differentiable with respect to u_z^d everywhere, we can use the criterion in (8) to verify its submodularity property. The total cost J can be represented as

$$J(\mathrm{z}^d, \mathrm{u}_z^d, (\mathrm{z}^a)^+, \mathrm{z}^f, \mathrm{z}^{\text{obs}}) = J^{\text{obs}}(\mathrm{z}^d, \mathrm{u}_z^d, \mathrm{z}^{\text{obs}}) + J^{\text{avoid}}(\mathrm{z}^d) + J^a(\mathrm{z}^d, \mathrm{u}_z^d, (\mathrm{z}^a)^+) + J^f(\mathrm{z}^d, \mathrm{u}_z^d, \mathrm{z}^f) + J^d(\mathrm{z}^d, \mathrm{u}_z^d),$$

where each of these terms is the summation of the terms in (20)-(22) respectively. Therefore, we can verify the submodularity criteria on each component of J individually.

$$\frac{\partial^2 J^{\text{obs}}(\mathrm{z}^d, \mathrm{u}_z^d, \mathrm{z}^{\text{obs}})}{\partial u_{x,j}^d \partial u_{x,i}^d} = 0, \quad \frac{\partial^2 J^{\text{avoid}}(\mathrm{z}^d)}{\partial u_{x,j}^d \partial u_{x,i}^d} = 0$$
$$\frac{\partial^2 J^a(\mathrm{z}^d, \mathrm{u}_z^d, (\mathrm{z}^a)^+)}{\partial u_{x,j}^d \partial u_{x,i}^d} = 0, \quad \frac{\partial^2 J^f(\mathrm{z}^d, \mathrm{u}_z^d, \mathrm{z}^f)}{\partial u_{x,j}^d \partial u_{x,i}^d} = 0,$$
$$\frac{\partial^2 J^d(\mathrm{z}^d, \mathrm{u}_z^d)}{\partial u_{x,j}^d \partial u_{x,i}^d} = -2(w_{ij}^d + w_{ji}^d) < 0.$$

The same is true for derivatives with respect to y components, i.e., $u_{y,i}^d$ and $u_{y,j}^d$. Since all the cross second order derivatives

of J with respect to the decision variables are non-positive, based on Thm. 2.1 and (8), the cost function of $\mathcal{P}1$ is submodular. ■

To solve $\mathcal{P}1$, defender d_i can minimize the terms J_i^{obs}, J_i^{avoid}, J_i^a, and J_i^f locally without coordinating with its team members. All these terms depend only on the input of d_i along with other quantities that are known to d_i at time k, i.e., locations of obstacles, current locations of other defenders, locations of attackers, and the locations of the defense zone. Although d_i does not know the exact locations of the attackers, it uses an estimate of these locations based on the current locations of the attackers and the assumed mobility model. Therefore, all these terms can be minimized by using subgradient descent algorithm and greedy algorithm for computing the subgradient.

The term J_i^d depends on the future locations of all the defenders, because of which this term cannot be minimized locally. Thus, the distributed optimization algorithm presented in this work needs to be implemented to solve $\mathcal{P}1$. We assume that each defender can only communicate with a subset of the members of the defense team and the adjacency matrix representing the communication network topology satisfies the assumptions presented in Section III. Since $\mathcal{P}_1$ is an online optimization problem, the defenders need to run Algorithm 2 at each decision time to compute an update action. To be compatible with the presentation in this paper, let U_x and U_y be the index sets for the input sets of U_x^d and U_y^d. Then, Algorithm 3 is the online version of the distributed optimization algorithm presented in the previous section.

The problem formulations in $\mathcal{P}$ and $\mathcal{P}1$ covers all the major requirements of any motion coordination problem 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 Algorithm 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 solve motion coordination algorithms over discrete state space in a distributed manner.

V. 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 $T = 20$. The defense zone was located at the top of the field and consisted of the following points

$$D = \{(4, 20), (4, 19), (5, 19), \ldots, (17, 19), (17, 20)\}$$

The number of players in both the teams was four, i.e., $n_d = n_a = 4$. The initial locations of the attackers and defenders were

$$\mathrm{x}^d(0) = (7, 9, 11, 12), \quad \mathrm{y}^d(0) = (18, 18, 18, 18).$$
$$\mathrm{x}^a(0) = (2, 10, 15, 20), \quad \mathrm{y}^a(0) = (2, 2, 2, 2).$$

There were six obstacles in the field located at

$$\mathrm{x}^{\text{obs}} = (4, 8, 6, 9, 13, 12), \quad \mathrm{y}^{\text{obs}} = (13, 10, 14, 14, 13, 15)$$

The layout of the field is shown in Fig. 2(a). 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. For obstacle avoidance $\eta_1 = 200$ and $\eta_2 = 50$, and for collision avoidance $\eta_1 = 50$ and $\eta_2 = 50$ for all the defenders. For cohesion among the defenders, the following weights were used

$$W^d = 0.01 \begin{bmatrix} 0.0 & 0.7 & 0.2 & 0.1 \\ 0.5 & 0.0 & 0.4 & 0.1 \\ 0.1 & 0.4 & 0.0 & 0.5 \\ 0.1 & 0.2 & 0.7 & 0.0 \end{bmatrix}$$

where $W_{ij}^d = w_{ij}^d$. To switch between attacking and defense modes, we selected $\alpha_{\text{nom}}^a = \alpha_{\text{nom}}^f = 0.5$ and the threshold distance $\delta_{\text{th}} = 15$.

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

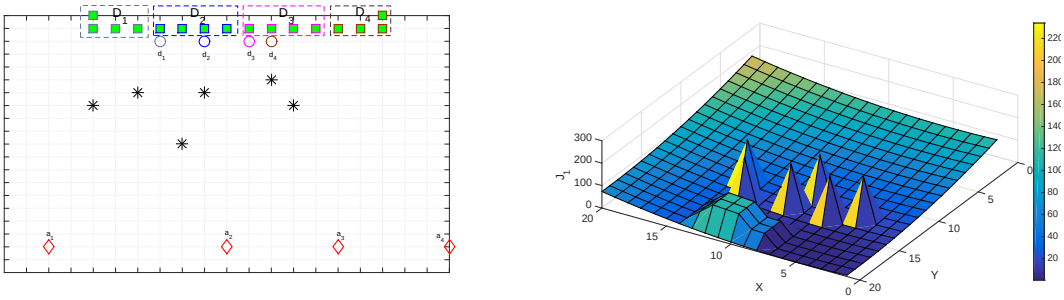
Algorithm 3 Online Distributed Submodular Minimization

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

1: **for** $k = 0$ to $T - 1$ **do**
2: Select the initial estimate $\mathrm{u}_z^i \in \prod_{i=1}^{n_d}(U_x \times U_y)$.
3: Apply Algorithm 2 with u_z^i as initial condition and $\bar{\mathrm{u}}_z^i$ as output.
4: Update the state

$$z_i^d(k+1) = z_i^d(k) + \bar{u}_{z,i}^i. \tag{29}$$

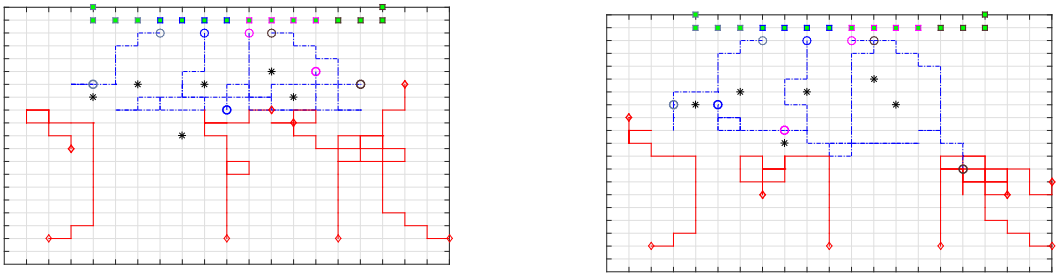
5: **end for**



(a) Layout of the playing arena.

(b) Cost function J_1.

Fig. 2. The layout of the playing arena with the initial locations of the offense and defense team is depicted in Fig. 2(a). The cost function J_1 of defender d_1 is shown in Fig. 2(b) such that the locations of the defenders and attackers are as shown in Fig. 2(a).



(a) $\beta = 0.2$

(b) $\beta = 0.95$.

Fig. 3. Trajectories of the offense and defense teams are plotted for different values of β in (28).

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

where $\kappa \in [0, 1]$ is a constant value. If $\Delta^a_i(k) < \Delta_{\text{th}}$, the value of $\eta_{i,\text{avoid}}(k)$ increases because a defender is closer than the threshold value. However, as $\Delta^a_i(k)$ increases, $\eta_{i,\text{avoid}}(k)$ keeps on decreasing. In the simulation setup, we selected $\eta^{\text{nom}}_{\text{avoid}} = 0.7$, $\eta^{\text{nom}}_{\text{base}} = 0.3$, $\Delta_{\text{th}} = 4$ and $\kappa = 0.9$.

In Fig. 2(b), the cost $J_1(\mathrm{z}^d, \mathrm{u}^d_z, (\mathrm{z}^a)^+, \mathrm{z}^f)$ of d_1 is demonstrated. For this figure, the location z^d_1 of d_1 is varied along the entire grid while keeping the locations of all the other attackers and defenders fixed at the initial locations as per the setup described above. The isolated peaks correspond to the locations of the obstacles and the crests correspond to high potential for collision avoidance.

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.

$$A = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 \\ 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 1/2 & 1/2 \end{bmatrix}$$

The simulation results are presented in Fig. 3 for two values of β, which controlled the transition of defenders behavior from defense to attack in (28). It can be observed from the simulation results that the attackers had more emphasis on avoiding the defenders than capturing the base. Consequently, non of the attackers could enter the defense zone. However, the defenders were unable to capture any attacker as well. In Fig. 3(b), the value of β was increased from 0.2 to 0.95, which switched the behavior of defenders from defensive to attacking more rapidly. As a result, one attacker was captured and the other attackers were pushed further away from the defense zone.

VI. CONCLUSION

In this work, we presented a distributed framework for minimizing submodular functions. The proposed framework was based on the property of submodular functions that their convex relaxation can be computed efficiently in polynomial time. Thus, a submodular function can be minimized over a discrete state space in polynomial time by minimizing its convex relaxation over a continuous domain. The proposed framework combined the ideas of convex relaxation of submodular functions and distributed optimization of convex functions. We also identified distributed motion coordination in multiagent systems as a new application domain for submodular minimization. We demonstrated through an example setup and verified via simulations that submodular minimization is a natural framework for distributed motion coordination in multiagent systems over discrete state space.

REFERENCES

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

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

[3] F. Bach, "Learning with submodular functions: A convex optimization perspective," *Foundations and Trends® in Machine Learning*, vol. 6, no. 2-3, pp. 145–373, 2013.

[4] S. Dughmi, "Submodular functions: Extensions, distributions, and algorithms. a survey," *arXiv preprint arXiv:0912.0322*, 2009.

[5] 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.

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

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

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

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

[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," *Communications*, 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] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] D. M. Topkis, *Supermodularity and Complementarity*. Princeton university press, 2011.

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

[22] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on automatic control*, vol. 51, no. 3, pp. 401–420, 2006.

[23] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[24] J. R. Marden, G. Arslan, and J. S. Shamma, "Cooperative control and potential games," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1393–1407, 2009.

[25] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[26] K. H. C. Wang and A. Botea, "Tractable multi-agent path planning on grid maps," in *IJCAI*, vol. 9, 2009, pp. 1870–1875.

[27] J. Yu and S. M. LaValle, "Multi-agent path planning and network flow," in *Algorithmic Foundations of Robotics X*. Springer, 2013, pp. 157–173.

[28] M. Mamei, F. Zambonelli, and L. Leonardi, "Distributed motion coordination with co-fields: A case study in urban traffic management," in *Autonomous Decentralized Systems, 2003. ISADS 2003. The Sixth International Symposium on*. IEEE, 2003, pp. 63–70.

[29] A. Kashyap, T. Başar, and R. Srikant, "Quantized consensus," *Automatica*, vol. 43, no. 7, pp. 1192–1203, 2007.

[30] G. Hollinger, S. Singh, J. Djugash, and A. Kehagias, "Efficient multi-robot search for a moving target," *The International Journal of Robotics Research*, vol. 28, no. 2, pp. 201–219, 2009.

[31] 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.

[32] 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.

[33] F. Bach, "Submodular functions: From discrete to continuous domains," *arXiv preprint arXiv:1511.00394*, 2015.

[34] 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.

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

[36] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.