Auxiliary States for Decentralized Optimization in Probabilistic Communication Networks

Noah Adhikari
Joshua Hug, Ed.
Lisa Yan, Ed.

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-114
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-114.html

May 16, 2025

Auxiliary States for Decentralized Optimization in Probabilistic Communication Networks

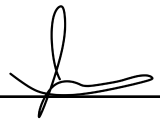
by Noah Adhikari

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:



Professor Joshua Hug
Research Advisor

5/16/2025

Date

Professor Lisa Yan
Second Reader

5/16/2025

Date

,

Auxiliary States for Decentralized Optimization in Probabilistic Communication Networks

by

Noah Adhikari

A thesis submitted in partial satisfaction of the requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Joshua Hug, Chair
Professor Lisa Yan, Second Reader

Spring 2025

Abstract

Auxiliary States for Decentralized Optimization in Probabilistic Communication Networks

by

Noah Adhikari

Master of Science in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Joshua Hug, Chair

Large optimization problems often require distribution and communication across several nodes. One class of such problems is consensus optimization, where agents must agree upon an optimal solution to these problems in a decentralized manner. Distributed primal-dual methods such as the consensus alternating direction method of multipliers (C-ADMM) are applicable when the communication network is static. However, dynamic communication is less well-studied; prior work has adapted C-ADMM to only a small class of dynamic networks. We generalize C-ADMM further by introducing auxiliary states, which capture information that may affect communication and model communication probabilities as a function of both endogenous and exogenous factors. In addition, we propose a novel, generalized C-ADMM variant, called ASV-ADMM, designed for dynamic communication graphs with auxiliary state-dependent edge transition probabilities. We evaluate ASV-ADMM on several scenarios with state-dependent network topologies wherein agents distributively optimize a global objective.

Acknowledgments

Huge thanks and credit to Katherine (Katie) How for helping run experiments and drafting some initial results as part of an ME292B Fall 2024 final project.

An immeasurable thank you to the following lovely people:

Josh Hug, for advising me throughout this whole journey and supporting my pivot to a topic that was well outside your expertise (sorry about that!)—your eagerness to understand, learn, and mentor has been immensely helpful;

Lisa Yan, for graciously agreeing to be my second reader on such unfamiliar material, and for providing wonderful guidance on a moment's notice;

Negar Mehr, Jingqi Li, and Kartik Nagpal, for their guidance on multi-agent control and games;

Junsang Yoon, for giving surprisingly constructive feedback after I treated him to Popeyes;

Mommy, Daddy, and Maya, for their unconditional love and support since my literal day one—I would not have graduated without you;

Kaitlyn, for making every day a little less lonely, and for paying impeccably close attention in your feedback;

and finally, to all my friends, colleagues, peers, and everyone else: you have brightened my days, shaped who I am, and made my life profoundly better—thank you! I could not have done this without you.

Contents

1. Introduction

In recent years, dataset volumes have grown substantially across several sectors [1], [2], making computational space and time constraints increasingly prohibitive and necessitating the division of workloads across multiple computing nodes.

Proper division of workloads depends on the problem domain. In particular, there are two distinct computational contexts to consider (pictorial examples are shown in Figure 1.1):

> *Centralized* settings, where central nodes distribute data to worker nodes and reduce workers' intermediate computations into a final result. The final result need only be available on the central nodes; intermediate worker computations may be discarded or transformed.
>
> *Decentralized* settings, where no nodes are strictly designated for distribution, collection, and/or coordination. In contrast with centralized settings, final computational results must be accessible to all nodes, rather than only a central subset of the nodes.

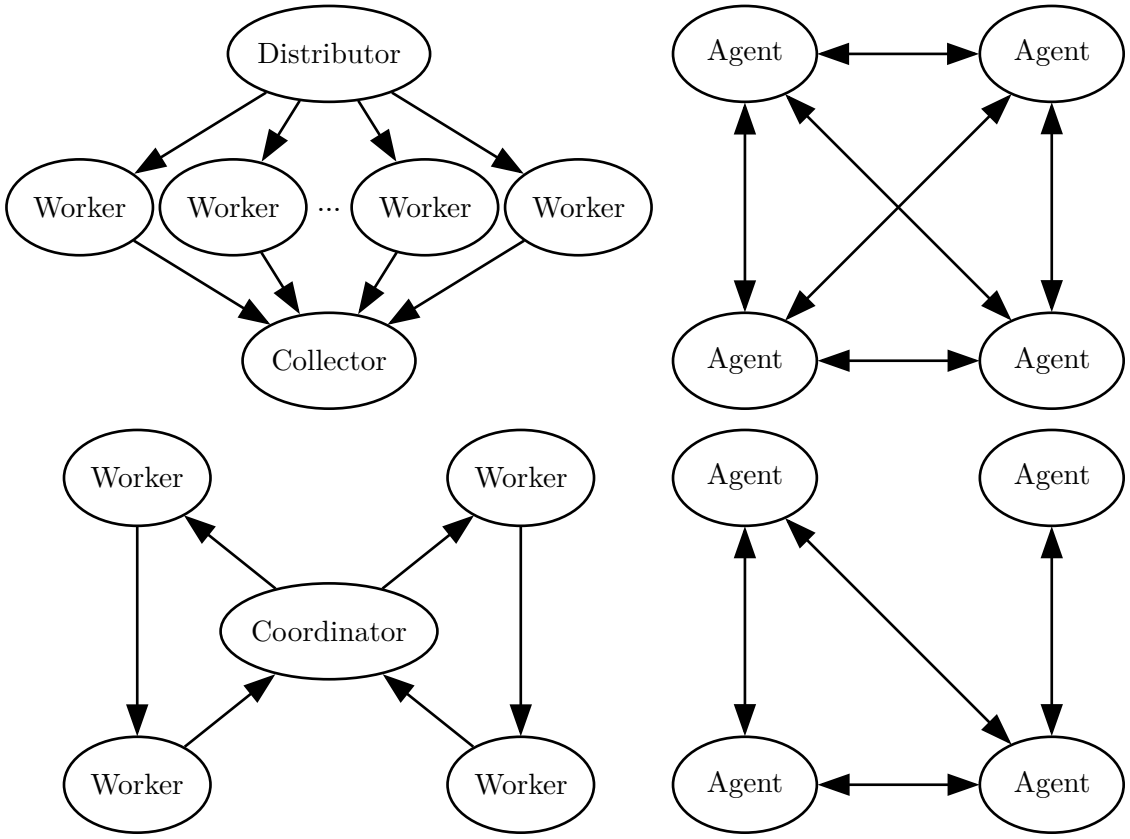


Figure 1.1: Examples of centralized (left column) and decentralized (right column) computation schemes. Workers in the centralized setting are not necessarily prohibited from communicating with one another and a decentralized graph need not be dense.

Though methods such as MapReduce [3] are suitable for centralized domains, they are not applicable in every scenario. For example, on a colossal dataset, where no one node can store all

data centrally, data must already be distributed across the computation nodes, rendering single-node distribution/collection impossible.

Another case, known as *federated learning*, is when individual agents may not wish to share sensitive data, but still wish to collectively fit a model using their data [4].[1] Moreover, agents may only have the capability to communicate with a small subset of other agents, so communication and data sharing become nontrivial. Moreover, agents may have differing abilities, beliefs, or desires, which makes central coordination more challenging. The additional considerations in decentralized settings make them more complex than their centralized counterparts.

Both centralized and decentralized settings often give rise to optimization problems with the goal of finding the best set of parameters to optimize some objective. This is often formulated as minimizing cost with respect to a vector-valued variable. There are many ways to solve such problems, including direct, gradient-based, adaptive, primal-dual, and metaheuristic methods [5], [6], [7].

A popular method for decentralized optimization is C-ADMM, the consensus alternating direction method of multipliers [8]. This method assumes a static communication network, where agents are able to communicate with the same set of agents at all time. This is a strong assumption —in many real-world applications, a dynamic communication graph is necessary to accurately represent a problem, as communication often is imperfect and can fail [9]. Communication failure is often tied to some auxiliary factors that may be related (endogenous) or unrelated (exogenous) to the optimization variable [10]. For example, suppose agents are collectively covering some area; then an endogenous factor is spatial position, and an exogenous factor is battery life. Though prior work has explored the endogenous case [10], the general case has not yet been formalized, to the best of our knowledge.

Prior work in decentralized optimization in dynamic networks includes Time-Varying ADMM (TV-ADMM) [11], a variant of C-ADMM allowing for failures in communication links. TV-ADMM considers only a small class of dynamic networks where communication links fail with some known, unchanging probabilities. A natural extension of TV-ADMM is to allow these probabilities to vary.

With this motivation, we mathematically formulate auxiliary states, which intuitively influence communication probabilities. This extends TV-ADMM to the case where communication probabilities may no longer be unchanging and instead are assumed to be a function of auxiliary states. This generalization of TV-ADMM is called Auxiliary State-Varying ADMM (ASV-ADMM). We evaluate ASV-ADMM against C-ADMM in a synthetic environmental experiment, the motivation of which is given in Section 4.

Our results indicate that ASV-ADMM is a suitable choice for decentralized optimization in dynamic networks where communication failure probabilities are difficult or impossible to capture with a single scalar value, but instead vary with respect to auxiliary factors. The experiment

[1]The term *agent* is used to denote a single computing machine, often with some associated physical state, as is assumed to be in a robotics context (similar to "node," "worker," or "instance"). The term "agent" is chosen as in multiagent settings, there is often no clear central machine, and agents may have differing opinions or desires.

and auxiliary state formulation motivates several avenues for future research in decentralized optimization, detailed in Section 7.

In this report, single- and multi-agent optimization methods are reviewed in order to provide a comprehensive motivation and introduction for ASV-ADMM, then ASV-ADMM is evaluated in comparison to the baseline C-ADMM in a resource availability scenario. A summary of ADMM variants discussed is given in the table below.

Method	**Description**	**Origination**
ADMM	Decomposable method of multipliers with alternating optimization blocks	[12]
C-ADMM	Suitable for distributed optimization subject to consensus constraints. Removes sequential dependencies in block updates by incorporating neighboring updates in communication graph.	[5]
TV-ADMM	Generalizes C-ADMM to probabilistic communication graphs with fixed failure probabilities by incorporating importance weights to primal-dual updates	[11]
ASV-ADMM	Generalizes TV-ADMM to a broader class of graph dynamics by modeling communication as a function of auxiliary states	Main contribution of this report

Table 1: A summary of ADMM variants discussed in this report.

2. Background

2.1. Notation

A subscript $_i$ denotes the variables associated with agent i (similarly for $_j$ or $_{ij}$). A parenthesized superscript, e.g. $^{(t)}$, denotes a time-dependent quantity. In this work, this is usually discrete-time (so $t \in \mathbb{N}$).[2]

Unless otherwise stated, $\|v\|$ refers to the Euclidean or ℓ_2-norm $\|v\|_2 = \sqrt{v^T v}$.

$\preceq$ or $\succeq$ refer to elementwise comparison on vectors.

$\mathcal{D}(S)$ denotes a (computable) probability distribution over the set S.

2.2. Constrained Minimization

A classical optimization problem is the constrained minimization problem

$$\begin{aligned} &\min_{\theta} J(\theta) \\ &\text{subject to } g(\theta) = 0 \end{aligned} \tag{1}$$

where $\theta \in \mathbb{R}^m$ is the main optimization variable, $J : \mathbb{R}^m \to \mathbb{R}$ is a real-valued loss or cost function, and $g : \mathbb{R}^m \to \mathbb{R}^k$ is a vector-valued function of k real-valued constraints concatenated into a vector-valued constraint.

Many real-world problems may be expressed as a constrained optimization problem, including problems in healthcare [13], power allocation [14], and economics [15].

Inequality constraints ($h(\theta) \preceq 0$) are notably omitted in this formulation, as they can be represented by introducing nonnegative slack variables of the form $s \succeq 0$ and reformulating them as $h(\theta) + s = 0$, an equality constraint that can be included in g. The non-negativity constraint on s is also an inequality, but it may be enforced as an equality using the elementwise square of some other vector, i.e., $s = s' \odot s'$ (where $\odot$ refers to the Hadamard, or elementwise, product of two vectors).

Lagrangian duality broadens the optimization problem in Equation 1 by introducing Lagrange multipliers $\lambda \in \mathbb{R}^k$. The Lagrangian $\mathcal{L} : \mathbb{R}^m \times \mathbb{R}^k \to \mathbb{R}$ of Equation 1 is

$$\mathcal{L}(\theta, \lambda) = J(\theta) + \lambda^T g(\theta) \tag{2}$$

and Equation 1 may be rewritten as the *primal* problem[3]

$$\min_{\theta} \Big(\max_{\lambda} \mathcal{L}(\theta, \lambda) \Big) \tag{3a}$$

with the *dual* problem being

[2] It is sometimes more convenient to let t range continuously (so $t \in \mathbb{R}_{\geq 0}$), e.g. in Section 3.1. In this case, discrete-time update rules would have to be reformulated under a continuous-time paradigm, e.g. differential equations.

[3] This is often stated using infimum and supremum instead of minimum and maximum. In this work they are assumed to be equivalent, i.e., sets are assumed to contain maxima and minima.

$$\max_{\lambda}\Big(\min_{\theta}\mathcal{L}(\theta,\lambda)\Big) \tag{3b}$$

At a primal optimum p^*, the primal problem is equivalent to Equation 1 as if the equality constraints are unsatisfied, then the cost will be arbitrarily large, as there are no restrictions on the Lagrange multipliers $\lambda \in \mathbb{R}^k$. When the constraints are satisfied, $g(\theta) = 0$, so no value of λ affects the cost, and the expression reduces to $J(\theta)$.

The dual optimum d^* is always such that $p^* - d^* \geq 0$ (a proof is provided in Section A.1), and this is known as the *duality gap*. When the duality gap is not 0 (weak duality), the dual optimum is still useful as it provides a lower bound on the primal optimum. In addition, even when the primal is nonconvex, the dual is guaranteed to be a concave optimization problem (a proof is provided in Section A.2), which often makes it much more tractable than the primal.

2.3. Single-Agent Methods

The tractability advantage of the dual problem gives rise to several methods for solving Equation 1. These form the foundation of centralized constrained optimization upon which decentralized optimization and ASV-ADMM further build.

Many of the following methods (and those in Section 2.4) are summarized from the optimization courses [7], [16].

2.3.1. Dual Ascent

Dual ascent is a method for solving Equation 1 which keeps iterates for both the primal and dual variables and updates them in an alternating manner:

$$\begin{aligned}\theta^{(t+1)} &= \arg\min_{\theta}\mathcal{L}\big(\theta,\lambda^{(t)}\big)\\ \lambda^{(t+1)} &= \lambda^{(t)} + \alpha^{(t)}\nabla_{\lambda}\mathcal{L}\big(\theta^{(t+1)},\lambda^{(t)}\big) = \lambda^{(t)} + \alpha^{(t)}g\big(\theta^{(t+1)}\big)\end{aligned} \tag{4}$$

where $\alpha^{(t)}$ is the learning rate or step size at time t. The minimization in the θ update is not circular since it minimizes $\mathcal{L}$ before the Lagrange multipliers λ can take an effect (hence solving the dual, not the primal). Since the dual is guaranteed to be concave, the repeated arg minimization of θ is rather easily solved in comparison to the primal problem [7].

2.3.2. Augmented Lagrangian and Method of Multipliers

Dual ascent tends to exhibit poor convergence without stringent assumptions on J [7]. It can be made more numerically robust by adding a quadratic penalty term to the Lagrangian to form the *augmented Lagrangian* [16]

$$\mathcal{L}_a(\theta,\lambda) = J(\theta) + \lambda^T g(\theta) + \frac{\rho}{2}\|g(\theta)\|^2 \tag{5}$$

and performing similar primal/dual updates to dual ascent using minimization and gradients of the augmented Lagrangian $\mathcal{L}_a$ instead of $\mathcal{L}$. When $\alpha^{(t)} = \rho$, this is called the *method of multipliers* [16].

2.3.3. Alternating Direction Method of Multipliers (ADMM)

The alternating direction method of multipliers (ADMM) [5], [12] further builds upon the method of multipliers by splitting the optimization variable θ into blocks and updating them similarly to the method of multipliers in an "alternating" manner.[4] This is also known as the "decomposable" method of multipliers.

Assuming that Equation 1 is decomposable, θ may be split into two blocks $\begin{bmatrix}\theta_A \\ \theta_B\end{bmatrix}$ such that $J(\theta) = J_A(\theta_A) + J_B(\theta_B)$.

The updates for ADMM are then similar to the method of multipliers, except that only one block is optimized at once:

$$\begin{aligned}
\theta_A^{(t+1)} &= \arg\min_{\theta_A} \mathcal{L}_a\left(\theta_A, \theta_B^{(t)}, \lambda^{(t)}\right) \\
\theta_B^{(t+1)} &= \arg\min_{\theta_B} \mathcal{L}_a\left(\theta_A^{(k+1)}, \theta_B, \lambda^{(t)}\right) \\
\lambda^{(t+1)} &= \lambda^{(t)} + \rho\nabla_\lambda \mathcal{L}_a\left(\theta_A^{(t+1)}, \theta_B^{(t+1)}, \lambda^{(t)}\right)
\end{aligned} \tag{6}$$

This generalizes to the case where J is decomposable into more than two blocks by repeatedly decomposing $J_A(\theta_A)$ or $J_B(\theta_B)$ further using the same technique.

2.4. Multi-Agent Methods

The methods covered in the preceding section all assume a single agent computing the optimal solution. There are several reasons for parallelization. The most common is performance; further, sometimes it is impossible to formulate certain settings as a centralized single-agent problem (Section 1). For example, agents may differ in abilities, beliefs, or desires. Amalgamating agents together into one single agent is possible, but is not ideal and often loses the nuance of imperfect information between agents. This section covers some multi-agent adaptations of the aforementioned single-agent methods.

Static Communication Graph

Parallelization necessitates a means of communication between agents. Suppose each agent is represented as a node in a graph (or network), with edges representing whether two agents are able to communicate. In the most general case, this graph may be undirected (bidirectional communication) or directed (explicit senders/receivers). For simplicity, we focus on the undirected case.

For now, the graph is connected. This formalizes the assumption that given enough time, information from any agent can propagate to any other agent, as there is a path of communication edges between any given pair of agents. Additionally, the network is *static* (unchanging). Both these restrictions will later be relaxed in accordance with the formulation in Section 2.4.2. In particular, N agents are represented as integral vertices in an undirected, connected, *static communication graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where

[4] The "alternating" does not refer to a change in sign but rather the axis upon which the optimization is performed.

$$\begin{aligned} \mathcal{V} &= \{1, ..., N\} \\ \mathcal{E} &= \{(i,j) : i \text{ can communicate with } j\} \end{aligned} \tag{7}$$

and $\mathcal{N}_i \triangleq \{j : (i,j) \in \mathcal{E}\}$ denotes the set of neighbors of agent i in $\mathcal{G}$.

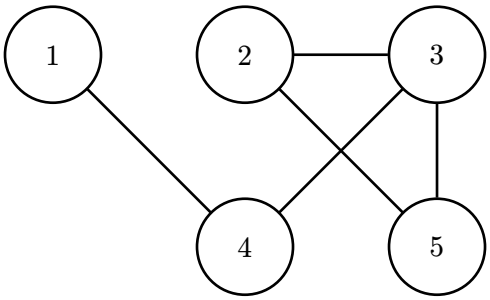


Figure 2.1: An example of a communication graph $\mathcal{G}$ with $N = 5$ agents.

2.4.1. Consensus ADMM (C-ADMM)

Now that multiagent communication has been formalized, consider solving an optimization problem in a distributed manner. In particular, consider a multiagent system where agents must come to a consensus on a solution θ to a problem similar to Equation 1. In contrast to the single-agent methods, each agent keeps its own *opinion* of the solution as a local variable θ_i and evaluates θ_i using an agent-specific cost function J_i. Notably, θ_i is *local* to agent i (it can be communicated to other agents, but only agent i is responsible for storing and controlling it), and J_i is *private* (known only to agent i).

A natural setting for this is a learning problem where a dataset is distributed across several agents in a decentralized manner. Each agent's cost function then depends on its own subset of data, and since agent datasets differ, they each will (very likely) have differing opinions on the optimal θ. However, they still wish to use their collective knowledge to come to a consensus for the optimal value of θ in the learned model.

With this motivation, suppose that Equation 1 is of the form

$$\min_{\theta} \sum_{i=1}^{N} J_i(\theta) \tag{8}$$

and to formalize agent opinions θ_i, we introduce the slack variable $z \in \mathbb{R}^m$ and reformulate Equation 8 as

$$\begin{gathered} \min_{z;\, \theta_1, ..., \theta_N} \sum_{i=1}^{N} J_i(\theta_i) \\ \text{subject to } \theta_i = z \text{ for all agents } i \end{gathered} \tag{9}$$

where the constraints $\theta_i = z$ are called the *consensus constraints*. Each cost function J_i is private to agent i. Since each agent stores its own θ_i and θ_i is not private, θ_i may be communicated to

other agents.[5]

One may approach this with traditional ADMM, but there are sequential dependencies in the updates that prevent the method from reaping the benefits of parallelization. In particular, the consensus optimization problem is a formulation of ADMM with the constraints

$$g\left(\begin{bmatrix}\theta_1\\ \vdots \\ \theta_N\end{bmatrix}\right) = \begin{bmatrix}\theta_1\\ \vdots \\ \theta_N\end{bmatrix} - \begin{bmatrix}z\\ \vdots \\ z\end{bmatrix} = 0 \tag{10}$$

Applying the traditional ADMM updates to Equation 9 means that at each iteration, agents depend on many others in order to perform one update. The alternating nature of ADMM means that θ_2's update depends on the updated θ_1, θ_3's update depends on the updated θ_2 and θ_1, and so on. θ_N in particular is quite problematic since it depends on *every* other agent's update. In other words, the θ_i updates have a serial dependence, which means that traditional ADMM is not parallelizable. Moreover, if $\mathcal{G}$ is sparse (e.g. Figure 2.2), then it can take a long time (worst case linear in $|\mathcal{V}| = N$) for agent N to collect all the information necessary to perform its update.

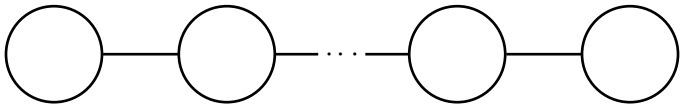


Figure 2.2: A linked-list communication graph topology, which is sparse and lends itself to the worst-case serial dependence in the traditional ADMM updates for consensus optimization.

Consensus ADMM (C-ADMM) remedies the problems traditional ADMM has with parallelization by restricting update dependencies to only *neighbors* in $\mathcal{G}$, where the neighbors of i in $\mathcal{G}$ are $\{j : (i,j) \in \mathcal{E}\}$. This removes the serial dependence as agents only need to incorporate neighboring opinions at each timestep so only one timestep is required to propagate required opinions (the aforementioned problematic agent N no longer depends on agents $1, ..., N-1$).

One concern is that faraway opinions are no longer propagated properly, since agents only communicate with their neighbors. Suppose agent i and j are far apart in the network, and agent i's update is of concern. Since (at least one of) i's neighbors will receive j's information eventually, j's opinion is not attenuated; it is simply included in a delayed neighbor update. Moreover, with traditional ADMM, j's update would need to propagate across the network regardless in order to update i (while earlier agents remain idle), so continuing to update neighboring opinions while i waits for j's opinion reduces idling across the system.

With this, Equation 9 can be reformulated as

[5]Agents must communicate their opinions θ_i in order to come to a consensus. Though there are indeed communication constraints in the decentralized/federated learning setting, it is assumed that the dataset is much more memory-intensive than θ and that θ is not sensitive (agents must come to a consensus on a solution θ), so communicating θ_i is not problematic.

$$\begin{aligned} &\min_{\theta_1,\ldots,\theta_N} \sum_{i=1}^{N} J_i(\theta_i) \\ &\text{subject to } \theta_i = \theta_j \text{ for all } (i,j) \in \mathcal{E} \end{aligned} \tag{11}$$

The updates for agent i are given by

$$\begin{aligned} &\text{initialize } \theta_i^{(0)} \text{ arbitrarily} \\ &\text{initialize } \lambda_i^{(0)} = 0 \\ &\theta_i^{(t+1)} = \arg\min_{\theta_i} \left\{ J_i(\theta_i) + (\theta_i)^T \lambda_i^{(t)} + \rho \sum_{j \in \mathcal{N}_i} \left\| \theta_i - \frac{\theta_i^{(t)} + \theta_j^{(t)}}{2} \right\|^2 \right\} \\ &\lambda_i^{(t+1)} = \lambda_i^{(t)} + \rho \sum_{j \in \mathcal{N}_i} \left(\theta_i^{(t+1)} - \theta_j^{(t+1)} \right) \end{aligned} \tag{12}$$

(For a complete derivation, see Section A.3.) These updates are derived from analyzing the Lagrangian of Equation 9 with slack variables and eliminating several intermediate variables by exploiting the symmetry of the communication graph.

Dynamic Communication Graph

The C-ADMM algorithm assumes that $\mathcal{G}$ is static. Unfortunately, in real-world multi-agent systems, communication may vary across timesteps as agents may move in and out of proximity, etc. [9]. Capturing this nuance in the communication graph formulation does not require much modification, and this gives rise to the *dynamic communication graph* $\mathcal{G}^{(t)}$, where

$$\begin{aligned} \mathcal{V} &= \{1, ..., N\} \\ \mathcal{E}^{(t)} &= \{(i,j) : i \text{ can communicate with } j \text{ at time } t\} \\ \mathcal{G}^{(t)} &= \left(\mathcal{V}, \mathcal{E}^{(t)}\right) \\ \mathcal{N}_i^{(t)} &= \left\{ j : (i,j) \in \mathcal{E}^{(t)} \right\} \end{aligned} \tag{13}$$

Note that in this formulation, vertices are static but edges are dynamic (i.e., agents cannot be introduced nor removed but communication may vary with time).

Unfortunately, removing the static constraint on $\mathcal{G}$ invalidates many of the theoretical convergence guarantees known for C-ADMM, since the system loses a lot of its symmetry.

2.4.2. Time-Varying ADMM (TV-ADMM)

TV-ADMM [11] attempts to generalize C-ADMM to a very specific class of dynamic communication graphs.[6] In particular, TV-ADMM assumes each edge $(i,j) \in \mathcal{E}^t$ is present with some known, constant probability p_{ij}.

[6]Note that in this report, the abbreviation "TV" means Time-Varying, not Total Variation, as may be the case in other literature.

Before, in Section 2.4.1, $\mathcal{G}$ was assumed to be connected. Now, since the edges are allowed to vary, this may no longer be the case. Often there are some connectedness assumptions made on dynamic communication networks using the union of communication edges over time to allow information to propagate through the network [9]. However, since the TV-ADMM formulation varies edges probabilistically, it is possible to not have union-connectedness given enough time. For this reason, in this report, we do not assume that G is connected given enough time. This will allow for more natural auxiliary states when defined in Section 3.1.

A natural application of this is a network in which links may be faulty with some known, constant probability. Different links may have different communication probabilities. For example, if two agents i and j fail to communicate 1% of the time, then $p_{ij} = 0.99$. If agents i and j are not able to communicate at all (in the static model), then $p_{ij} = 0$.

Whereas C-ADMM views neighboring opinion contributions equally since neighbors remain constant at all timesteps, TV-ADMM introduces an importance weight which weights unlikely neighbor contributions more highly than likely ones. The motivation is to make the updates more uniform to try and bring it closer to the C-ADMM updates.

In order to achieve this, neighboring connectivity probabilities are first normalized via a softmax operation

$$w_{ij}^{(t)} = \frac{\exp\big(p_{ij}\big)}{\sum\limits_{k \in \mathcal{N}_i^{(t)}} \exp(p_{ik})} \tag{14}$$

and the *importance weight* for the ij-link is denoted

$$\alpha_{ij}^{(t)} = \frac{1}{|N_i^{(t)}|\ w_{ij}^{(t)}} \tag{15}$$

where intuitively, the inverse relationship more greatly weights contributions from unlikely connections. TV-ADMM modifies the C-ADMM updates accordingly:[7]

$$\begin{aligned} \theta_i^{(t+1)} &= \arg\min_{\theta_i} \left\{ J_i(\theta_i) + (\theta_i)^T \lambda_i^{(t)} + \rho \sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(t)} \left\| \theta_i - \frac{\theta_i^{(t)} + \theta_j^{(t)}}{2} \right\|^2 \right\} \\ \lambda_i^{(t+1)} &= \lambda_i^{(t)} + \rho \sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(t)} \Big(\theta_i^{(t+1)} - \theta_j^{(t+1)} \Big) \end{aligned} \tag{16}$$

Inspired by [17], TV-ADMM replaces the last term $\rho \sum(...)$ (call it $h_i^{(t+1)}$) in the $\arg\min$ with a first-order Taylor approximation around the previous iterate $\theta_i^{(t)}$:

[7]The original TV-ADMM paper assumes a regularized cost function $J(\theta) + \mu r(\theta)$, where μ is a regularization strength parameter and $r(\theta)$ is the ℓ_1 or ℓ_2 norm. However, since this can be viewed as another cost function $J'(\theta)$, the regularization term $\mu r(\theta)$ is omitted here.

$$
\begin{aligned}
h_i^{(t+1)}(\theta_i) &\approx h_i^{(t+1)}\left(\theta_i^{(t)}\right) + \left(\nabla h_i^{(t+1)}\left(\theta_i^{(t)}\right)\right)^T \left(\theta_i - \theta_i^{(t)}\right) \\
&= h_i^{(t+1)}\left(\theta_i^{(t)}\right) + \left(\nabla h_i^{(t+1)}\left(\theta_i^{(t)}\right)\right)^T \theta_i - \left(\nabla h_i^{(t+1)}\left(\theta_i^{(t)}\right)\right)^T \theta_i^{(t)}
\end{aligned} \tag{17}
$$

Here, $\nabla h_i^{(t+1)}$ refers to an arbitrary (sub-)gradient of $h_i^{(t+1)}$, and since $h_i^{(t+1)}$ is in an $\arg\min$, the constant terms $h_i^{(t+1)}\left(\theta_i^{(t)}\right)$ and $-\left(\nabla h_i^{(t+1)}\left(\theta_i^{(t)}\right)\right)^T \left(\theta_i^{(t)}\right)$ may be dropped as they do not affect the minimization over θ_i.

Additionally, TV-ADMM includes a Bregman divergence term $\left\|\theta_i - \theta_i^{(t)}\right\|^2$ with weight $\beta^{(t)} = \beta_0\sqrt{2t}$ (where β_0 is tuned experimentally as in TV-ADMM) in order to reduce variance caused by the randomness of the network. This term, intuitively, makes agents firmer in their opinions as time elapses. Together with the first-order approximation of $h_i^{(t+1)}$, this yields the θ_i update provided in TV-ADMM:

$$
\theta_i^{(t+1)} = \arg\min_{\theta_i}\left\{ J_i(\theta_i) + (\theta_i)^T \lambda_i^{(t)} + \left(\nabla h_i^{(t+1)}\left(\theta_i^{(t)}\right)\right)^T \theta_i + \beta^{(t)}\left\|\theta_i - \theta_i^{(t)}\right\|^2 \right\} \tag{18}
$$

TV-ADMM is applicable when the communication probabilities p_{ij} are easily calculable, but even simple scenarios require thorough analysis and heuristics to determine appropriate values of p_{ij} [11]. In some cases, p_{ij} may even vary over time, which violates the assumptions made in TV-ADMM (fixed communication probabilities), so the method cannot be applied directly. This motivates modeling communication probabilities as a function of some other factors, so the importance weight α_{ij} can be determined more easily, making TV-ADMM more widely applicable. This is the basis for ASV-ADMM (Auxiliary State-Varying ADMM), an extension of TV-ADMM and the original contribution of this report, further discussed in the next section.

3. Auxiliary State-Varying ADMM (ASV-ADMM)

TV-ADMM assumes unreliable communication links with a fixed, known probability p_{ij}, and this assumption is quite restrictive, as agent communication in reality is often nonstatic. Communication may depend on many factors, including spatial factors such as proximity, obstacles, or dead zones, or temporal factors such as battery drainage. As such, it is beneficial to define something which allows us to capture the communication effectiveness as a function of these factors. This will allow for the extension of TV-ADMM to more complex communication failure models. We dub this extension Auxiliary State-Varying ADMM (ASV-ADMM), and introduce and motivate it in this section.

3.1. Auxiliary States

To avoid confusion with the "state" of an agent i typically referring to the optimization variable θ_i, it is helpful to define its *auxiliary state*, which intuitively represents the factors upon which an agent's communication depends.

The set of all possible auxiliary states that agent i can attain is called agent i's *auxiliary domain*, denoted via X_i. Each X_i is a subset of the *global auxiliary domain* $X \triangleq \bigcup_{i=1}^{N} X_i$, and $x_i^{(t)} \in X_i$ denotes the auxiliary state of agent i at iteration t.

Allowing X_i to vary for each i (as opposed to just having all $X_i = X$) allows for agent-specific constraints on agents' auxiliary states. For example, consider a scenario where agents can only roam a particular part of the environment, but must stay within a certain distance of a "home base." In this case, agents' "home bases" need not be identical, so their auxiliary domains should not be identical. This example is shown in Figure 3.1.

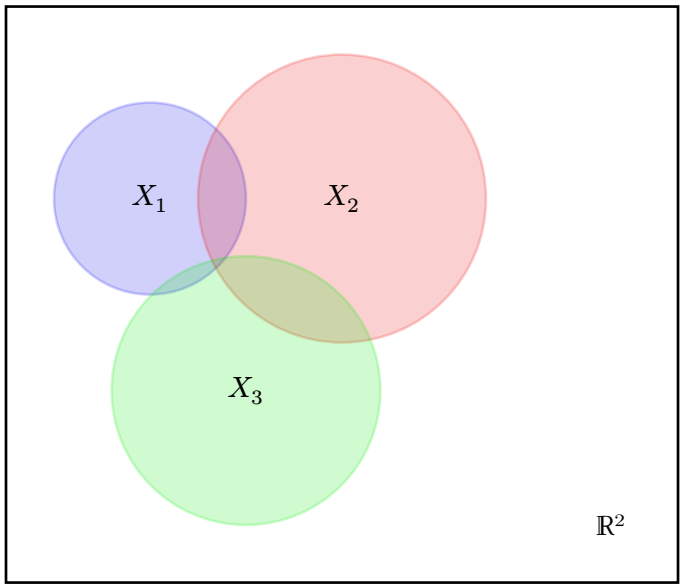


Figure 3.1: An example where agent auxiliary domains differ. In this case, agents can only roam a region within a particular distance of a point. In this example, $X = X_1 \cup X_2 \cup X_3 \subset \mathbb{R}^2$.

Auxiliary states evolve according to some auxiliary update rule

$$x_i^{(t+1)} = \text{update}\left(x_i^{(t)}, \varepsilon_i^{(t)}\right) \tag{19}$$

where $\varepsilon_i^{(t)} \sim \mathcal{D}(X)$ represents noise and may be agent-specific.

The concept of auxiliary states is similar to exogeneity (externality) in the literature [10]; however, auxiliary states may incorporate endogenous information in addition to exogenous information: X may (but need not) be entirely unrelated to the θ-optimization domain $\mathbb{R}^m$. In other words, agent communication may depend on (endogenous) factors related to their optimization variable, external (exogenous) factors, or a combination of both.

As an example, coverage agents seek to optimize their positions, which certainly affects proximity (endogenous), but may not consider battery life (exogenous). However, both of these factors may affect communication, and may be included in the global auxiliary domain X. As a result, the auxiliary state formulation is quite general in order to allow various communication models to fit this formulation.

3.2. Communication Probability Functions

The graph dynamics in TV-ADMM can be adapted to auxiliary states by allowing the communication graph edges each to vary with some time-dependent probability $p_{ij}^{(t)}$. That is, $(i,j) \in \mathcal{E}^{(t)}$ with probability $p_{ij}^{(t)}$.

In particular, assume that $p_{ij}^{(t)}$ is determined by some *known, static* function

$$p : X^2 \to [0,1] \tag{20}$$

of the auxiliary states $x_i^{(t)}$ and $x_j^{(t)}$ such that $p_{ij}^{(t)} = p\left(x_i^{(t)}, x_j^{(t)}\right)$. Since $\mathcal{G}$ is undirected, additionally impose the symmetry constraint $p_{ij}^{(t)} = p_{ji}^{(t)}$.

Note that even though we assume p is static, this model can still capture time-dependent graph transitions by simply adjoining t to x_i and treating the quantity $\left(x_i^{(t)}, t\right) \in X \cup \mathbb{N}$ as the new auxiliary state.

This is a generalization of the previous models. In the static communication model, p is

$$p\left(x_i^{(t)}, x_j^{(t)}\right) = \begin{cases} 1 \text{ if } (i,j) \in \mathcal{E} \\ 0 \text{ if } (i,j) \notin \mathcal{E} \end{cases} \tag{21}$$

where $X = \mathcal{V}$.

In TV-ADMM, p is simply $p\left(x_i^{(t)}, x_j^{(t)}\right) = p_{ij}$, where if $(i,j) \notin \mathcal{E}$, then $p_{ij} = 0$, and again $X = \mathcal{V}$. This represents a case where the communication link between agents i and j is faulty and fails with some (known or estimated) probability $1 - p_{ij}$.

Some natural examples of X and p are given below. As mentioned in Section 2.4.2, the connectedness assumption on $\mathcal{G}^{(t)}$ is relaxed since probabilistic graphs are not necessarily connected.

3.2.1. Distance Thresholding

One possibility for X is position, say, $\mathbb{R}^2$, for agents on a flat surface. The following are examples of some communication functions with these auxiliary states.

One proximity-based p-function is a threshold distance function, where $d_{ij}^{(t)} \triangleq \left\| x_i^{(t)} - x_j^{(t)} \right\|$, is

$$p\left(x_i^{(t)}, x_j^{(t)}\right) = \begin{cases} 1 \text{ if } d_{ij}^{(t)} \leq R \\ 0 \text{ if } d_{ij}^{(t)} > R \end{cases} \tag{22}$$

for some distance threshold R.

An example is given in Figure 3.2. This communication graph has the same vertices as in Figure 2.1 (assuming they were originally 1 unit away from their nearest neighbor). Now, though, the edges are assumed to be determined from the auxiliary domain $\mathbb{R}^2$ with a hard cutoff communication distance of $R = 1$ in Figure 3.2, leading to a much different set of edges. Vertex positions have been slightly jittered to indicate that agents are now moving in space. Note the discrepancy between the two edge sets when auxiliary states are introduced.

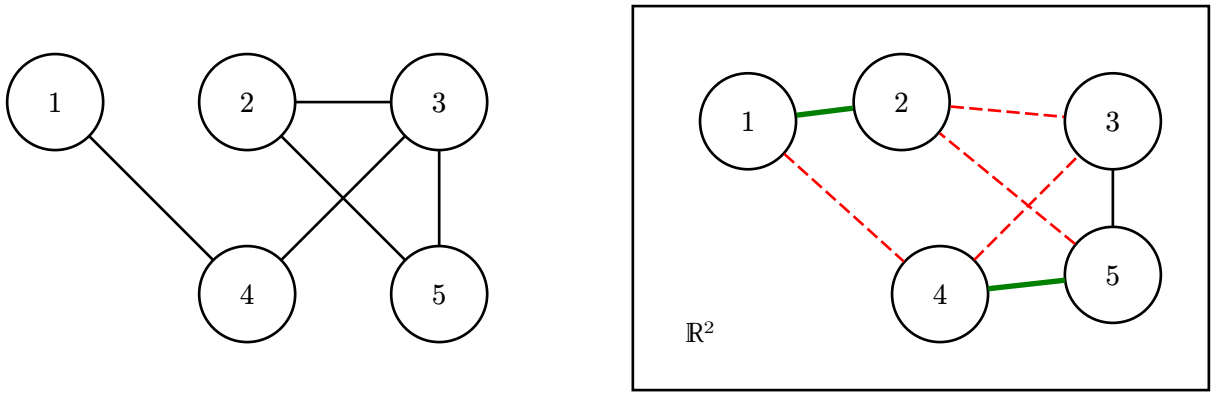


Figure 3.2: Left: Figure 2.1 copied for convenience. Right: The same communication graph with $X = \mathbb{R}^2$ under a hard distance thresholding p-function with $R = 1$ and agents slightly jittered. Red dashed edges indicate removed connections. Bolded green edges indicate newly formed connections.

A smoothed version of distance thresholding is also possible (with quadratic falloff rate κ):

$$p\left(x_i^{(t)}, x_j^{(t)}\right) = \begin{cases} 1 \text{ if } d_{ij}^{(t)} \leq R \\ \frac{1}{1+\kappa\left(d_{ij}^{(t)} - R\right)^2} \text{ if } d_{ij}^{(t)} > R \end{cases} \tag{23}$$

This is equivalent to

$$p\left(x_i^{(t)}, x_j^{(t)}\right) = \frac{1}{1 + \kappa\left(\max\left\{0, d_{ij}^{(t)} - R\right\}\right)^2} \tag{24}$$

When $\kappa = \infty$, the unsmoothed version is equivalent to the smoothed version.[8]

In general, it is difficult to visualize p as a function of two positions when $X = \mathbb{R}^n$, as there would be $2n$ inputs and a single real-valued output, requiring $2n + 1$ dimensions for visualization. However, since d_{ij} is always a scalar regardless of n and this p-function only depends on d_{ij}, one can plot p against d_{ij} instead. Figure 3.3 is generated with $R = 1$ and $\kappa = 1$.

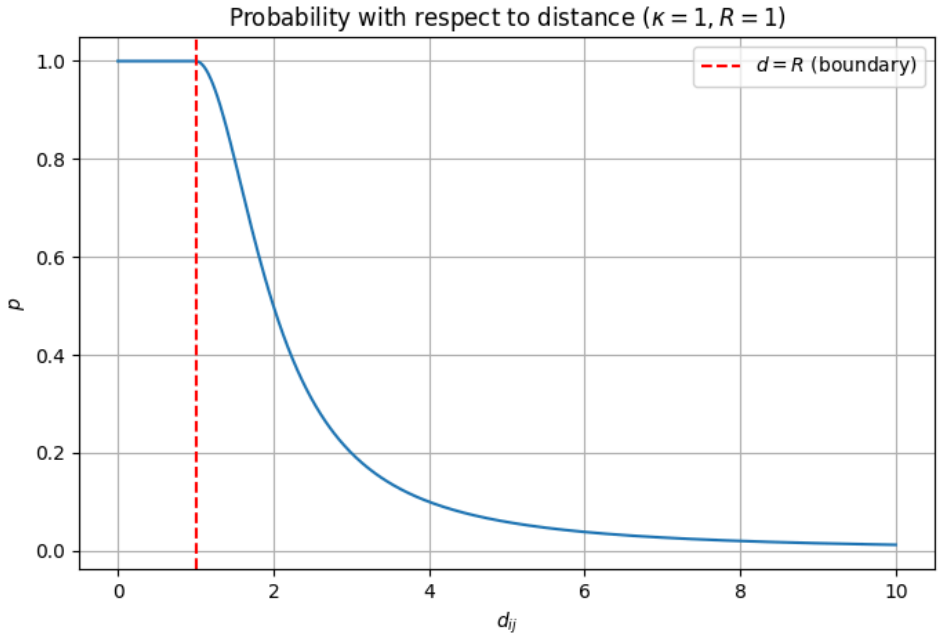


Figure 3.3: A smoothed distance thresholding p-function with distance on the horizontal axis. The red dashed line indicates the boundary at which communication begins to falter, i.e., p drops below 1.

One can see that when $d_{ij} < R$, communication is guaranteed ($p = 1$), and when $d_{ij} \gg R$, $p \approx 0$, as is natural for when agents are close or far apart.

Another workaround to have some visual intuition for agent communication is to restrict the auxiliary domain X to only have one dimension so $X = \mathbb{R}$. The communication interaction between two agents can then be visualized using only 3 dimensions since there are two inputs and a single output. These visualizations are provided in Section B.1.

As another workaround for visualization, consider $X = \mathbb{R}^2$ and fix one of the agents at a specific position, say, $(0, 0)$. Then, the other agent moves around freely in $\mathbb{R}^2$. We can then plot the probability p_{ij} as a function of $\mathbb{R}^2$ using only 3 dimensions. The contour plots in this section will take this form.

[8]Technically, at the boundary $d_{ij}^{(t)} = R$, the smoothed version is undefined when $\kappa = \infty$ due to the $\infty \cdot 0$ indeterminate form in the denominator. We define the boundary probability to be 1 in accordance with the equalities in Equation 22. If strict inequalities are desired, then the boundary probability should be 0.

Figure 3.4 provides an example of a smoothed distance thresholding p-function with $\kappa = 1$ and $R = 1$. Agent 1 is fixed at the origin $(0, 0)$ and agent 2 is able to freely roam the space, with $p(x_1, x_2)$ plotted on the colorbar from 0 to 1 as a function of agent 2's position. Note that concentric circles in Figure 3.4 represent d_{ij} so that the positive half of a cross-section through the origin of Figure 3.4 is identical to Figure 3.3.

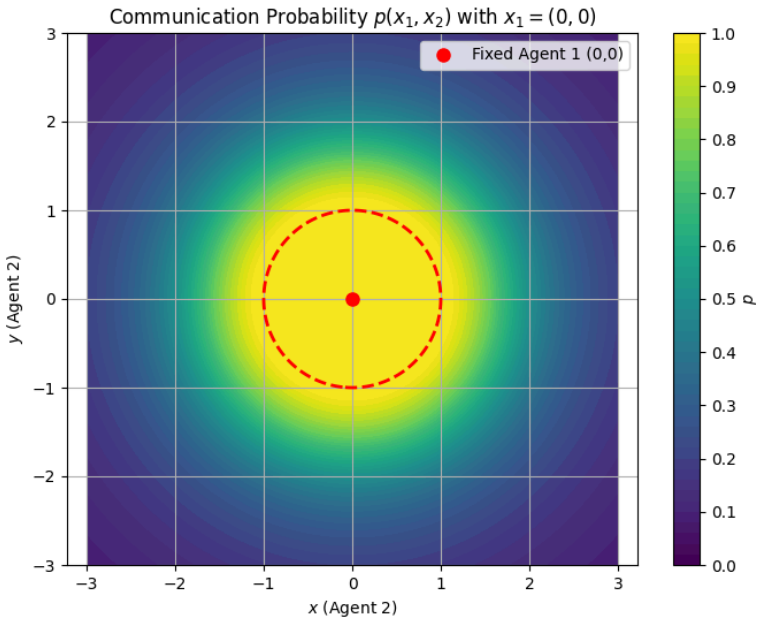


Figure 3.4: A smoothed distance-thresholding p-function with $X = \mathbb{R}^2$ and x_1 fixed at the origin $(0, 0)$. Red dashed lines indicate the boundary at which communication begins to falter, i.e., p drops below 1.

3.2.2. Dead Zones

"Dead zones," which limit communication, may also be incorporated with auxiliary states. One such p-function incorporating dead zones $D \subset X$ is

$$p\left(x_i^{(t)}, x_j^{(t)}\right) = \begin{cases} 0 \text{ if } x_i^{(t)} \in D \text{ or } x_j^{(t)} \in D \\ 1 \text{ otherwise} \end{cases} \tag{25}$$

When dead zone communication is not impossible, but less likely, one may use a nonzero probability if either agent is in a dead zone.

Figure 3.5 provides an example of how dead zones can affect communication. The same smoothed distance thresholding p-function in Figure 3.4 now has dead zones applied where communication is impossible. In particular, D consists of three circles with centers at $(-1, -1), (1, 1), (-1, 2)$, and radii of $\frac{1}{2}, \frac{1}{2}, 1$, respectively.

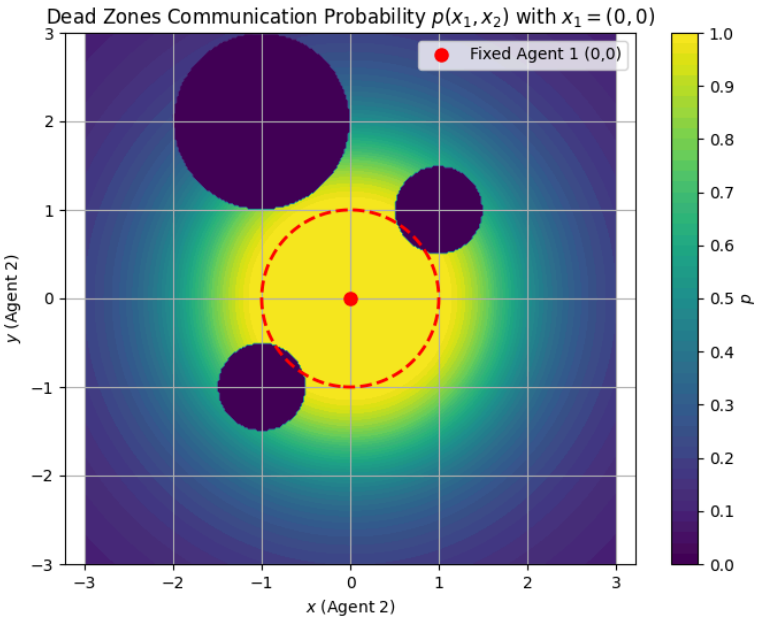


Figure 3.5: A smoothed distance-thresholding p-function with $X = \mathbb{R}^2$ and x_1 fixed at the origin with dead zones applied. Red dashed lines indicate the boundary at which communication begins to falter, i.e., p drops below 1.

3.2.3. Obstacles

Obstacles can also be included in a similar fashion to dead zones, though this may be more difficult to explicitly formulate mathematically. This p-function intuitively captures line-of-sight for impermeable obstacles. If one has a set of obstacles $O \subset X$, an impermeable obstacle formulation is

$$p\left(x_i^{(t)}, x_j^{(t)}\right) = \begin{cases} 0 \text{ if the line between } x_i^{(t)} \text{ and } x_j^{(t)} \text{ passes through any point in } O \\ 1 \text{ otherwise} \end{cases} \tag{26}$$

Similarly to dead zones partially inhibiting communication rather than making it outright impossible, permeable obstacles may be permissible as well (simply increase the communication probability representing permeability when communicating through obstacles to a nonzero value).

Figure 3.6 illustrates how line-of-sight can affect communication when obstacles are impermeable. This also uses the same smoothed distance thresholding p-function in Figure 3.4 and Figure 8, but now enforces line-of-sight between agents 1 and 2. O is the same as D in Figure 3.5: three circles with centers at $(-1, -1), (1, 1), (-1, 2)$, and radii of $\frac{1}{2}, \frac{1}{2}, 1$, respectively.

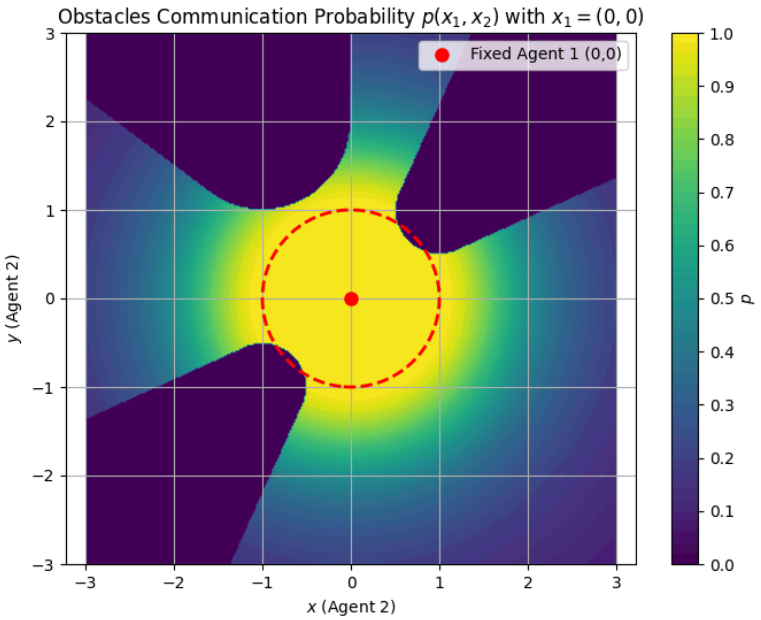


Figure 3.6: A smoothed distance-thresholding p-function with $X = \mathbb{R}^2$ and x_1 fixed at the origin with obstacles and line-of-sight applied. Red dashed lines indicate the boundary at which communication begins to falter, i.e., p drops below 1.

3.2.4. Temporal Factors

There may be auxiliary state information other than position that is helpful for capturing communication probabilities. For example, agents may have limited battery life. In this case, p should decrease with time, and X may simply be $\mathbb{N}$ (or $\mathbb{R}_{\geq 0}$ in a continuous-time setting).

One such p-function, simply characterizing whether agents are dead or alive, is

$$p\left(x_i^{(t)}, x_j^{(t)}\right) = \begin{cases} 1 \text{ if } t \leq B_i \text{ and } t \leq B_j \\ 0 \text{ otherwise} \end{cases} \tag{27}$$

where $B_i \in \mathbb{N}$ (or $\mathbb{R}_{\geq 0}$) is the amount of time agent i can stay alive (battery capacity), and we implicitly include i in the auxiliary state to determine B_i, so $X = \mathbb{N} \cup \mathcal{V}$ (or $\mathbb{R}_{\geq 0} \cup \mathcal{V}$). This can be smoothed in a similar manner to distance thresholding with B_i and B_j instead of R, with the interpretation of agents being more likely to have communication faults as their batteries deplete after a certain point. The communication falloff rate κ_i now allowed to be agent-specific as it depends on their individual battery capabilities which may be different.

$$p\left(x_i^{(t)}, x_j^{(t)}\right) = p(t_i, t_j) = \frac{1}{1 + \kappa_i(\max\{0, t_i - B_i\})^2 + \kappa_j\left(\max\{0, t_j - B_j\}\right)^2} \tag{28}$$

Note the unsmoothed formulation is the smoothed version with $\kappa_i = \kappa_j = \infty$.[9]

[9]Again, with a technicality at the boundaries $t_i = B_i$ or $t_j = B_j$, similarly to the previous footnote.

An example is given in Figure 3.7. Even though $X = \mathbb{R}_{\geq 0} \subset \mathbb{R}$ in this case, x_i does not represent spatial position, but a point in time, so this visualization does not represent the same concept as in the previous figures. This example is generated with $\kappa_i = \kappa_j = 1$, $B_i = 1$, and $B_j = 2$.

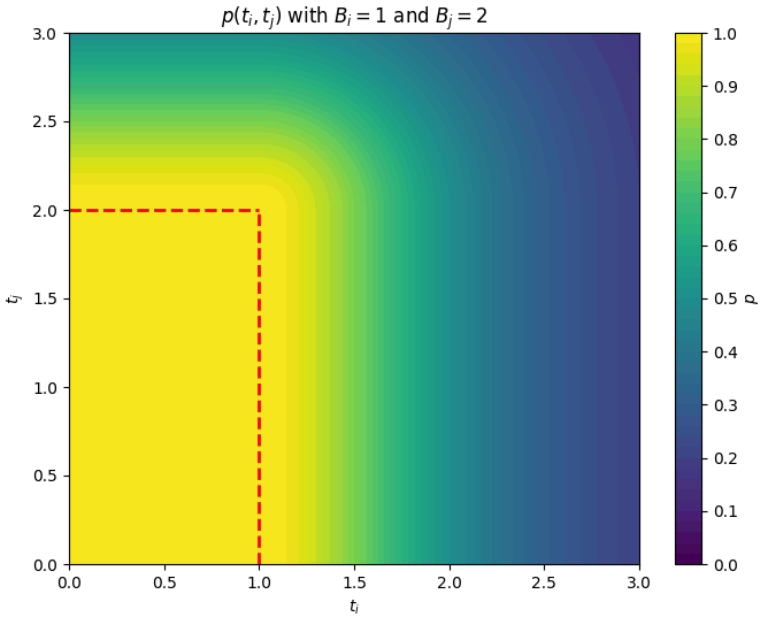


Figure 3.7: A plot of a smoothed distance-thresholding p-function of two agents' times $t_i, t_j \in \mathbb{R}$ with battery thresholds $B_i = 1$ and $B_j = 2$. Red dashed lines indicate the boundary at which communication begins to falter, i.e., p drops below 1.

These examples may also be combined such that the auxiliary domain captures multiple external factors and so does p. For example, perhaps battery charging could be incorporated into p so that if agents visit some recharge station, their batteries are refreshed. This would require auxiliary states to capture both spatial and temporal factors, and an additional component in X representing recent recharges.[10]

3.3. Auxiliary State-Varying ADMM (ASV-ADMM)

With auxiliary states, time-dependent probabilities p_{ij} can be represented via $p\left(x_i^{(t)}, x_j^{(t)}\right)$. Replacing the fixed p_{ij} terms in TV-ADMM with the time-dependent terms yields the novel (to the best of our knowledge) *auxiliary state-varying* ADMM (ASV-ADMM). The softmaxed weights $w_{ij}^{(t)}$ now incorporate the time-dependent probabilities $p_{ij}^{(t)}$:

$$w_{ij}^{(t)} = \frac{\exp\left(p_{ij}^{(t)}\right)}{\sum_{k \in \mathcal{N}_i^{(t)}} \exp\left(p_{ik}^{(t)}\right)} \tag{29}$$

[10]Instead of including the recharges in X, p could theoretically incorporate some more complicated internal state to remember if agents have recharged recently, but the auxiliary state formulation imposes the restriction that p is static, so p cannot have internal state.

and the importance weights $\alpha_{ij}^{(t)}$ and corresponding updates to θ_i and λ_i use the new formulation of $w_{ij}^{(t)}$. The updates are otherwise identical to those in Section 2.4.2.

4. Methodology

Suppose that mobile agents are trying to collectively learn some spatial field (e.g. temperature, resource availability, crowd density). In this scenario, agents roam the environment and collect data, then fit a model to that data. The single-agent case is conventional: the agent simply learns the model parameter θ using its local data. In the multi-agent case, however, each agent may have differing samples, so agents will likely have differing opinions on what model parameter θ is optimal (i.e., it is likely that $\theta_i \neq \theta_j \ \forall i \neq j$). They then need to communicate to come to a consensus on the *globally* optimal model parameter θ. Agents' local sample data might be difficult to communicate (for reasons discussed in Section 2.4.1), so C-ADMM is necessary to alleviate the struggles of traditional ADMM.

We would like to explore the convergence of ASV-ADMM under limited communication when compared with C-ADMM. With the above motivation, one way to formulate this scenario is to first allow agents to privately collect samples, then have them learn θ. After agents have collected sufficient samples, they communicate to come to a consensus on θ such that $\theta_i = \theta$. However, if the data collection process is expensive, this may not be ideal, and they may wish to collaboratively refine their global estimates whilst collecting data (rather than collecting everything in advance and then learning θ), to have an initial global estimate available more quickly.

In this case, agents must communicate *whilst* collecting samples, and hence communication naturally may depend on an auxiliary state (for example, in spatial field estimation, communication probability may decrease with distance). This case is more suited for ASV-ADMM than C-ADMM.[11]

4.1. Experimental Setup

Consider a distributed regression setting with the following properties:

- Agents repeatedly sample noisy data $y_i^{(t)} \in \mathbb{R}$ that depends on the auxiliary state $x_i^{(t)}$.
- There is some unknown global function $f : X \to \mathbb{R}$ (representing some quantity, e.g., temperature, as a function of position) which agents collectively attempt to approximate via $\hat{f} : X \to \mathbb{R}$, where $\hat{f}$ is some learned model with parameter θ.
- Agents seek to minimize the expected regression error J which captures the loss between f and $\hat{f}$ when aggregating over all samples.
- Agents repeatedly take samples $y_i^{(t)}$ by roaming their local environment $X_i \subseteq X$.
- Agents update their auxiliary states by taking a step towards a random point in X_i with some step size δ using a convex update.[12]
- Assume $y_i^{(t)} = f\left(x_i^{(t)}\right) + \eta_i^{(t)}$, where $\eta_i^{(t)}$ denotes zero-mean noise and is sampled i.i.d. from $\text{Gaussian}(0, \sigma)$.
- Agents only know their own sample data and can only communicate $\theta_i^{(t)}$ and $x_i^{(t)}$ to other agents (no samples $y_i^{(t)}$ are communicated).

[11] TV-ADMM cannot be applied here, assuming the communication probabilities p_{ij} are nonconstant.

[12] Here it is assumed that X_i is convex so that an agent can never "escape" X_i, that is, $x_i^{(t)} \in X_i$ for all time. However, one may modify the update so that it samples a random positional update and simply resamples if that would cause the auxiliary state to escape X_i.

- The global cost evaluates $\hat{f}$ on the collective local data $\left\{\left(x_i^{(t)}, y_i^{(t)}\right)\right\}$ across all agents and all t.

This is formulated mathematically below, where ${x_i}'^{(t)} \sim X_i$ independently and uniformly and T is the final timestep of evaluation:

$$\begin{aligned} x_i^{(t+1)} &= (1-\delta)x_i^{(t)} + \delta {x_i}'^{(t)} \\ J_i^{(t)}(\theta_i) &= \frac{1}{t}\sum_{t'=1}^{t} \left\| \hat{f}_i\left(x_i^{(t')}\right) - y_i^{(t')} \right\|^2 \\ &\text{Find} \underset{\theta_1,\ldots,\theta_N;\, z}{\arg\min} \frac{1}{N}\sum_{i=1}^{N} J_i^{(T)}(\theta_i) \\ &\text{subject to } \theta_i = z \quad \forall i \in \mathcal{V} \end{aligned} \tag{30}$$

One may notice that the function J_i is time-dependent due to the newly acquired samples at each timestep, and ADMM variants assume the cost function to be only dependent on the optimization variable θ. However, since the horizon length (T) is fixed in advance and the cost is only minimized at $t = T$, it is still in accordance with the ADMM formulation. In our experiment, though, for visual purposes, the mean squared error is evaluated at regular intervals.[13]

4.2. Experimental Details

Since the primary focus is on exploring the convergence under limited communication, and not so much on the sophistication of the model $\hat{f}$ given θ, we limit the class of $\hat{f}$ to be rather unexpressive. A simple but natural choice for f^* in the context of resource availability is a sum of Gaussians:

$$f^*(x_i) = \sum_{m=1}^{M} a_m \exp\left(\frac{-\|x_i - c_m\|^2}{2\sigma_m^2}\right) \tag{31}$$

where M is the total number of resource clusters with a_m, c_m, and σ_m representing the amplitude, center, and spread of the mth resource cluster, respectively. The goal is for agents to come to a consensus on the globally optimal a_m, c_m, and σ_m, representing the prevalence, location, and spread of resources, respectively.

We choose X to be the square $[-1, 1]^2$ so that x_i represents position in the 2D plane. We initialize agents uniformly at random in X.

Our goal is to measure several types of losses. Local losses refer to agent losses on their own private dataset at a specific timestep, i.e., $J_i^{(t)}\left(\theta_i^{(t)}\right)$. Global loss refers to the loss when aggregating over all private datasets using the average agent estimate:

[13]The motivation here is allowing agents to refine their opinions of θ as they collect more samples, rather than having to collect all the samples first and then come to a consensus on θ, and allowing the visualization of the opinions over time. This is not in accordance with the required ADMM formulation, which requires J to be time-independent.

$$\frac{1}{N}\sum_{i=1}^{N} J_i^{(t)}\left(\overline{\theta}^{(t)}\right) \tag{32}$$

where $\overline{\theta}^{(t)}$ is the mean of agent opinions $\frac{1}{N}\sum_{i=1}^{N}\theta_i^{(t)}$. Consensus loss refers to the mean squared error between agent opinions

$$\frac{1}{N}\sum_{i=1}^{N}\left\|\theta_i^{(t)}-\overline{\theta}^{(t)}\right\|^2 \tag{33}$$

As is done in the TV-ADMM paper [11], C-ADMM and ASV-ADMM are evaluated using the global objective loss as well as the consensus loss (how much agents vary in their opinions). The local objective losses are also evaluated.

For hyperparameters, we use $\rho = 0.001$ and $\beta_0 = 0.001$ (for the Bregman divergence term).

For experimental parameters, we use $M = 1$ and set N to be between 3 and 10. This is to ease computation time, but also because convergence was more difficult to achieve at greater values of M, N. We set $\delta = 1$ so auxiliary states are updated fully uniformly randomly within X_i, and since $X = [-1, 1]^2$ is convex, the convex update rule suffices. σ varies depending on the experiment. T is chosen experimentally but depends on the amount necessary to illustrate convergence or divergence to a significant degree.

Probability Functions

In this experiment, we evaluate the performance of ASV-ADMM on several choices for p inspired by real-world scenarios. For our first experiment, we use the proximity-based model described in Equation 24 and vary κ, the rate of communication falloff, in the experiment.

In another experiment (Section 5.3), we introduce dead zones into the environment, where agents are unable to communicate as described in Section 3.2.2.

Simulation

To evaluate our algorithm, we implemented ASV-ADMM in Python with C-ADMM as a benchmark. For a fair comparison, since the Bregman divergence term $\beta^{(t)}\left\|\theta_i-\theta_i^{(t)}\right\|^2$ is not the main insight in ASV-ADMM, the presented results also incorporate it into the baseline C-ADMM update.

Note that though it may technically be possible due to the uniform randomness of our experiments, TV-ADMM is not evaluated here due to the difficulty of explicitly calculating p_{ij} in many of the experiments. This demonstrates one advantage of ASV-ADMM over TV-ADMM in its ease of use.

We use the SciPy optimization package [18] to solve the $\arg\min$ in the primal-dual updates.

5. Results

5.1. Distance-Based Falloff Rate

For this experiment, all simulation parameters except the falloff rate κ are held constant to compare the performance of ASV-ADMM and C-ADMM as p varies. The following simulations are run with $N = 5$ agents, $M = 1$, $\sigma = 0.1$ over 200 timesteps.

The below plots show the local, global, and consensus losses (as defined in Section 4.1) as the falloff rate κ is gradually increased. The results of ASV-ADMM are shown in blue and C-ADMM in red (as is the case for all of our experiments). These plots represent a single run; in ensemble simulations, a single divergent run could significantly skew the results.

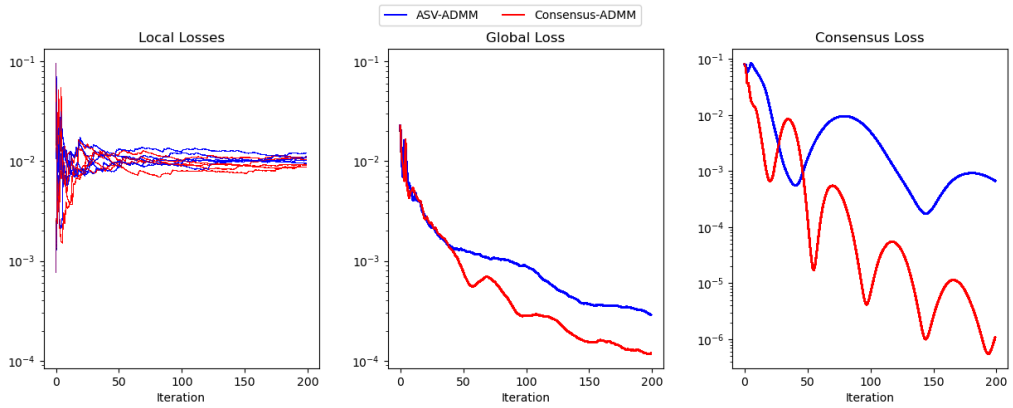


Figure 5.1: Falloff Rate of 0

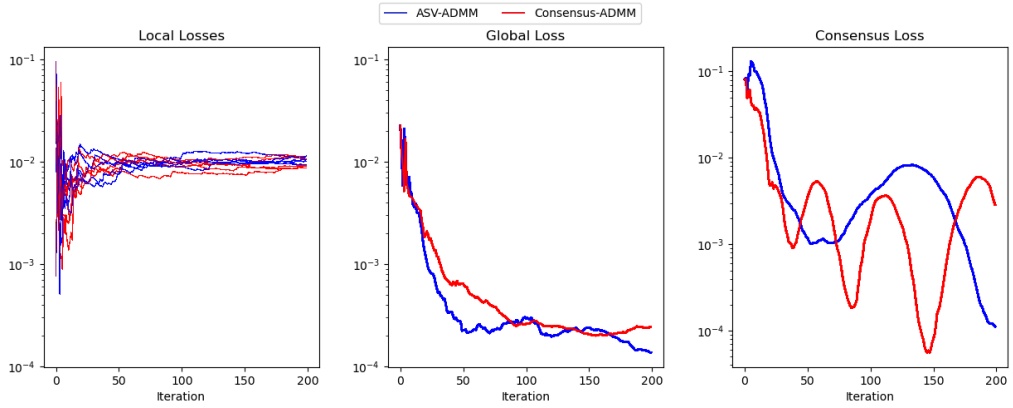


Figure 5.2: Falloff Rate of 12

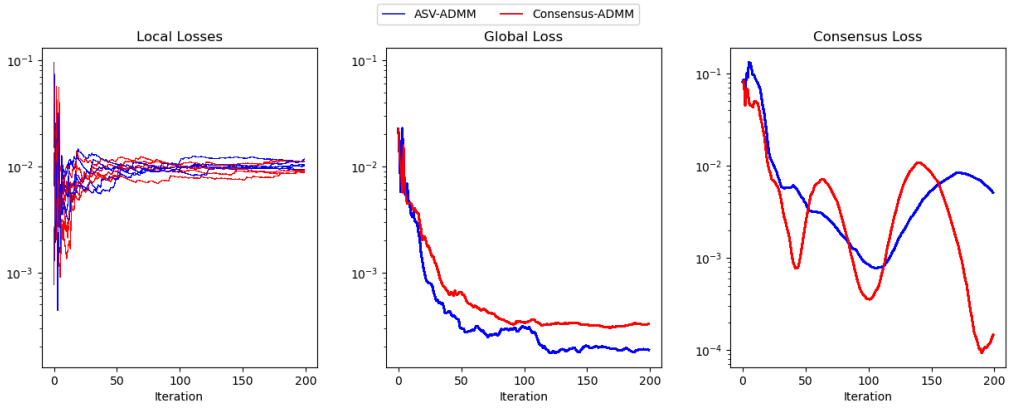


Figure 5.3: Falloff Rate of 25

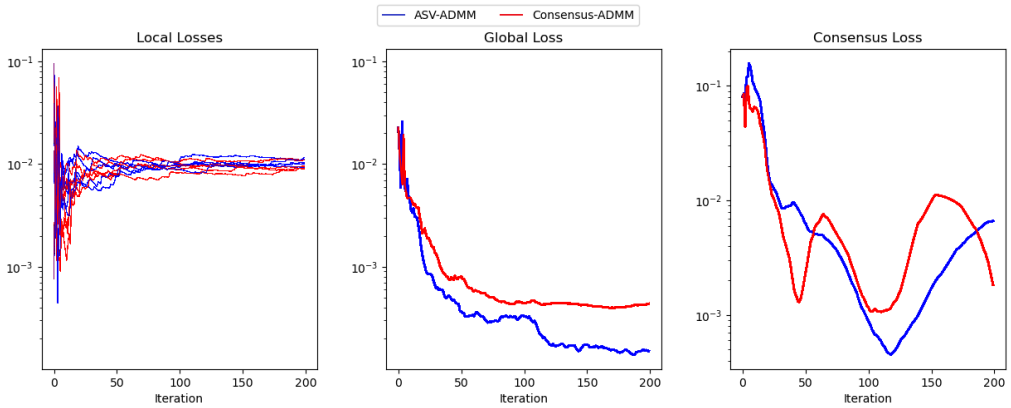


Figure 5.4: Falloff Rate of 50

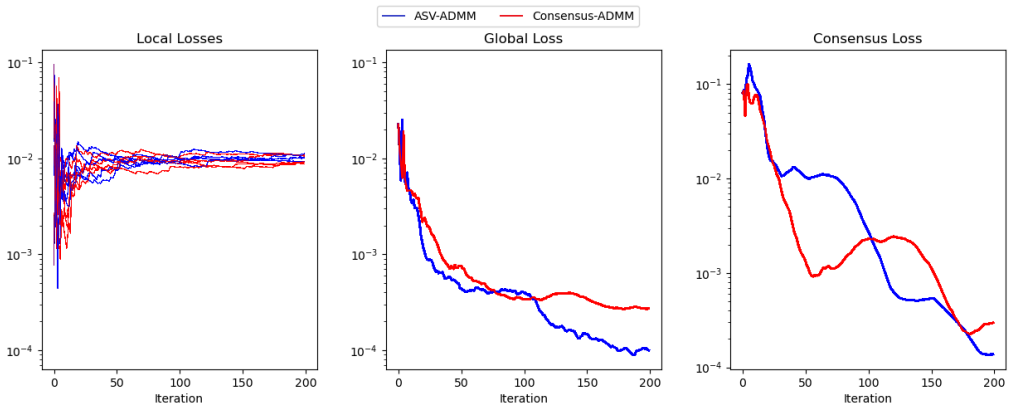


Figure 5.5: Falloff Rate of 100

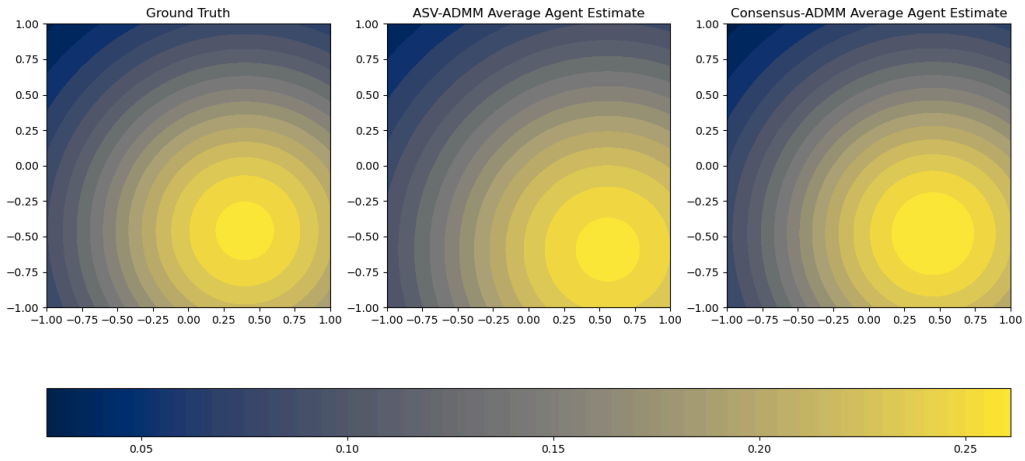


Figure 5.6: Final agent estimates with $\kappa = 6$. Color represents resource availability, with brighter colors indicating higher amounts of resources.

At lower falloff rates, the better performer is arbitrary. This is likely due to the communication graph still being static and connected enough for C-ADMM to perform well. However, as κ increases to 25 and above, ASV-ADMM becomes the better performer in global loss. We expect this is because as the communication graph switches more frequently, ASV-ADMM more equally weights agent contributions than C-ADMM, allowing agents to more easily incorporate information from unlikely neighbors.The consensus loss is often quite oscillatory; this is likely due to different agent opinions across the network that sway back and forth as information is propagated. However, it does generally decrease as time goes on, indicating agents are refining their opinions in a consensus manner.

5.2. Distance-Based Falloff Rate with Domain-Restricted Agents

In the previous experiment, every agent's data is drawn from the same distribution, which may allow C-ADMM to perform better than in real-world cases as agent-specific overfitting will generalize well to the global data. To investigate this in a more systematic manner, we run a simulation with the same objective and distance-based communication falloff but with $N = 4$ agents each restricted to their respective quadrants, i.e., $X_1 = (+, +)$, $X_2 = (-, +)$, $X_3 = (-, -)$, $X_4 = (+, -)$. This makes agent communication essential as they must share their local data since they are the only ones that can access it (without proper communication, agents will overfit to their local data). This experiment used $M = 1$ and $\sigma = 0.0$ over 200 timesteps. We chose no noise to highlight the communication conditions more easily in our results.

Figure 5.7 serves as a baseline comparison for this experiment with $\kappa = 0$, i.e., perfect communication.

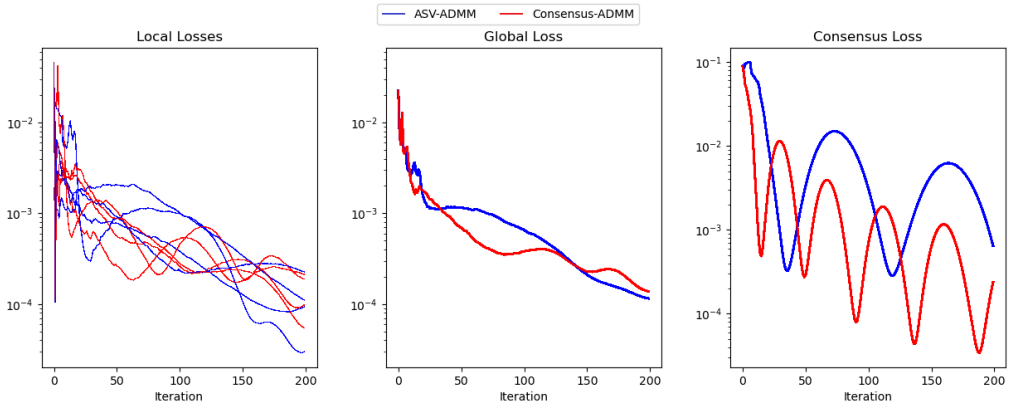


Figure 5.7: Quadrant Falloff Rate 0

The plots below illustrate when κ is increased to more severely limit communication:

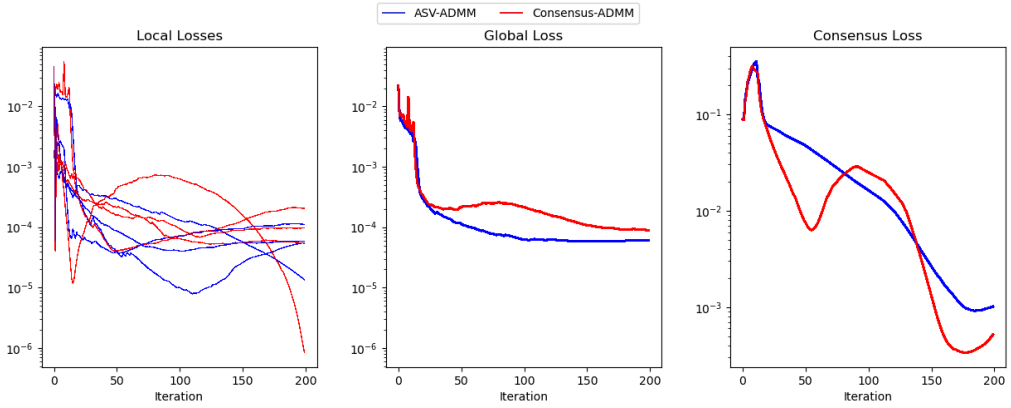


Figure 5.8: Quadrant Falloff Rate 50

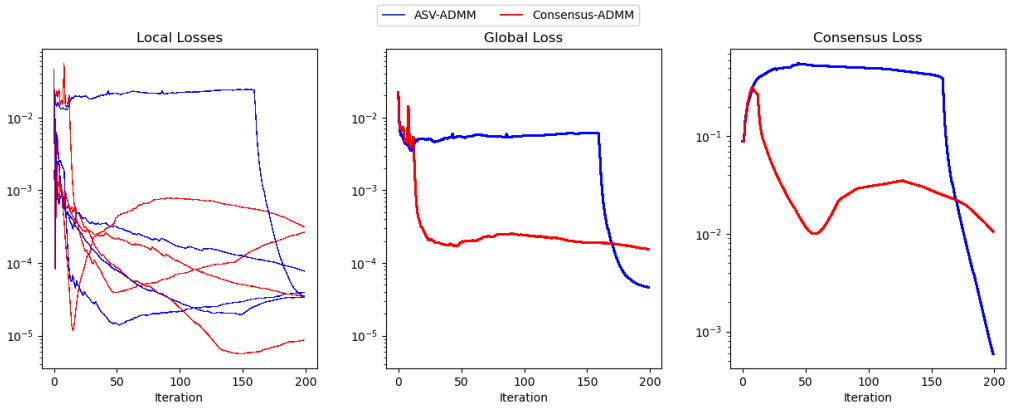


Figure 5.9: Quadrant Falloff Rate 100

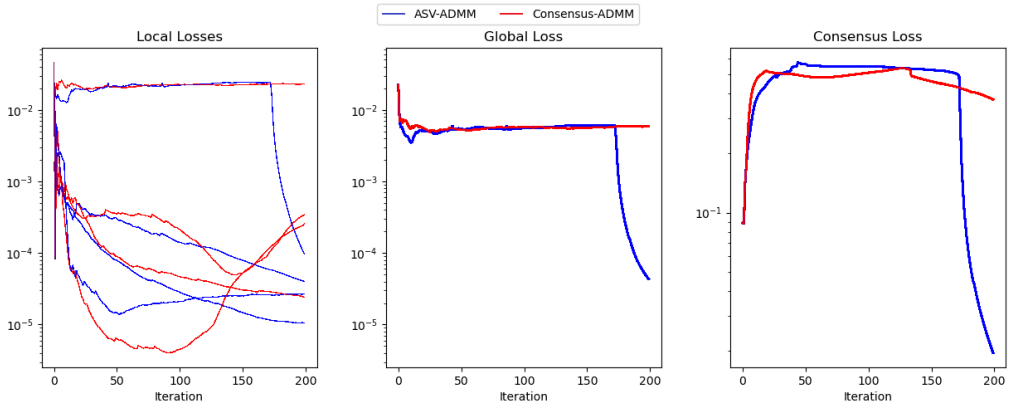


Figure 5.10: Quadrant Falloff Rate 203

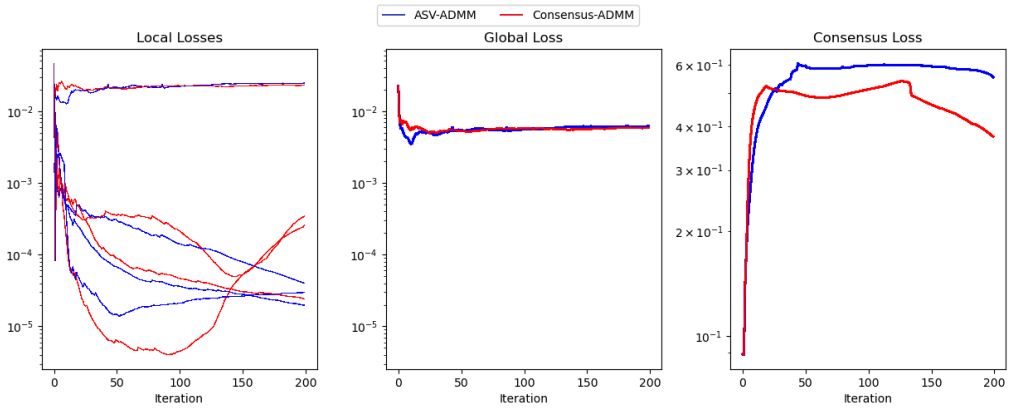


Figure 5.11: Quadrant Falloff Rate 225

Both ASV-ADMM and C-ADMM perform similarly from $\kappa = 1$ to $\kappa = 50$. However, at $\kappa = 100$, the behavior changes drastically where C-ADMM initially converges faster than ASV-ADMM, but near the end of the simulation, ASV-ADMM converges to a more optimal solution. As κ continues to increase to around 200 as shown in the last image, we can see that ASV-ADMM converges near the end while C-ADMM does not converge within the time frame. From the local losses, there appears to be an agent that has a vastly different opinion from everyone else, and C-ADMM is not able to reconcile this, but ASV-ADMM is near the end of the simulation. As κ continues to increase, both ASV-ADMM and C-ADMM diverge. Global and consensus losses are comparable for the cases where they both converge, and similar oscillatory behaviors as the previous experiment are observed but to a lesser extent.

5.3. Dead Zones

In this experiment, random circular regions are created in the environment where agents are unable to communicate with one another as seen below and in accordance with Equation 25. We see divergence in C-ADMM similar to communication falloff with a high value of κ and attribute this to the switching communication network and possibly "isolated" agents for some timesteps in this formulation throwing off agent updates.

The consensus losses corroborate this; one can see that C-ADMM agents struggle to come to similar parameters whereas ASV-ADMM is still able to come to a consensus. The local losses indicate that agents are willing to compromise some of their local opinion to form a better consensus, since it converges to their average (otherwise every local loss would mostly decrease). There may be some numerical instability at this scale in C-ADMM as the consensus loss does seem unreasonably large.

The following plots were generated with $N = 3$ agents and $\sigma = 0.1$ with $T = 50$.

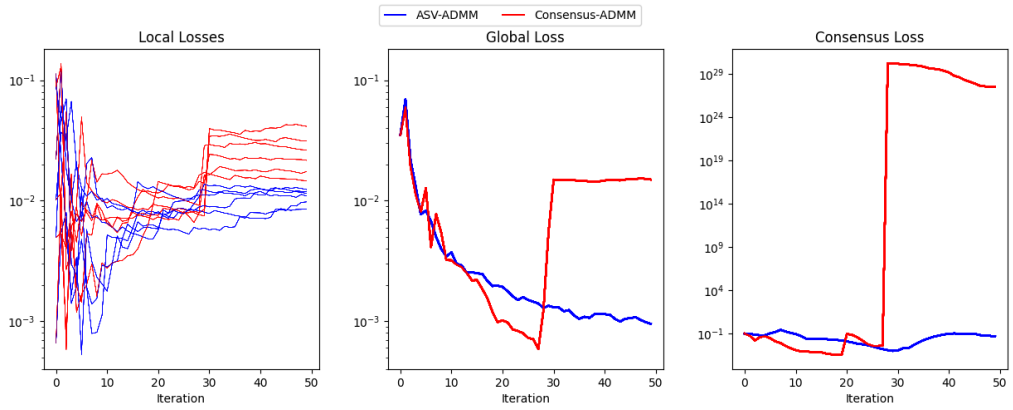


Figure 5.12: Dead zones loss curve

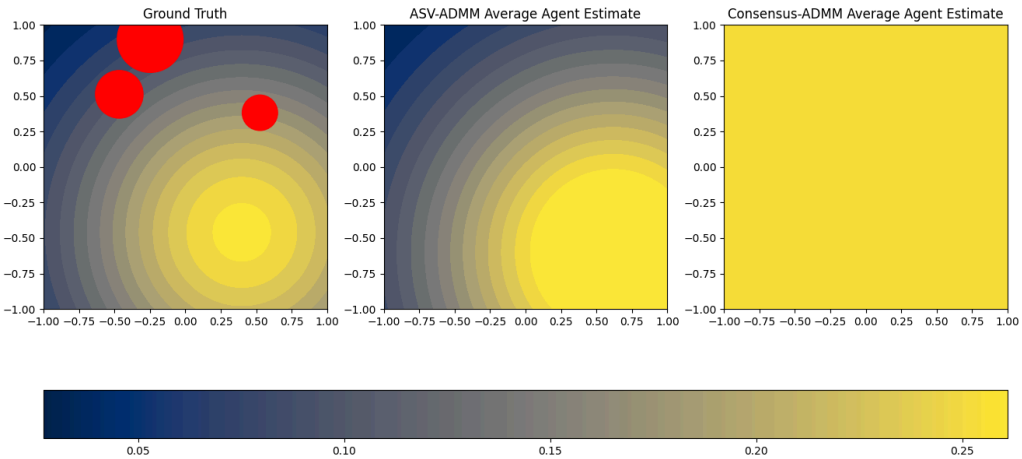


Figure 5.13: Dead zones final contour plot. Red areas represent dead zones. Final agent estimates with $\kappa = 6$. Color represents resource availability, with brighter colors indicating higher amounts of resources.

6. Discussion

Our loss curves use a logarithmic scale. Since our losses operate over many orders of magnitude, we felt this was necessary to accurately compare, though it does disproportionately highlight differences at lower orders of magnitudes.

From our experiments, it appears as though ASV-ADMM often performs similarly to C-ADMM and our results are inconclusive. Sometimes, though, the stochasticity of the environment and dynamics makes it difficult for agents to reach consensus, and eventually the frequent communication switching causes C-ADMM to diverge. ASV-ADMM often handles this better, keeping agents in agreement despite the erratic nature of the environment, until p becomes too harsh and both algorithms diverge. Though we ran many experiments with many parameters and hyperparameters, there was never a case where C-ADMM converged but ASV-ADMM diverged.

Though obstacle and temporal formulations were not explicitly tested, our intuition from these experiments suggests that obstacles would behave similarly to dead zones (as they are quite often mobile dead zones). With a battery drainage formulation, we expect that since agent communication falters over time, that the performance of the algorithms will initially be similar to the other experiments and cause divergence after agent batteries start to become problematic. Hence, we expect ASV-ADMM to be a good choice of decentralized optimization algorithm when convergence is difficult for C-ADMM even when its solution error is comparable to C-ADMM.

Though these results are promising, there are many caveats. In particular, the choice of hyperparameters (such as the consensus weight parameter ρ) substantially affects the performance of both algorithms, though this is the case for many algorithms in general.

Our results with convergence suggest that our algorithm requires some constraints on p as the communication graph may certainly evolve over time but still needs to allow information to flow from one side of the network to the other. We expect that with p that allows agents to communicate often, ASV-ADMM will perform similarly to C-ADMM. If p is too restrictive, both algorithms fail to converge. In some cases, where p falls in a happy medium between these two extremes, ASV-ADMM will outperform C-ADMM.

The importance weights of ASV-ADMM help agents come to consensus by weighting unlikely neighbor messages more significantly than likely ones. Since the weighting is inversely proportional to the likelihood, we expect that on average the updates will be more similar to that of a static communication graph. Moreover, assuming agent updates are roughly coupled with their auxiliary state as in our experiments, this helps prevent agents from overfitting to their local data, as C-ADMM will largely discard information from faraway agents.

However, if the environment is so stochastic or irregular that agent experiences are completely unrelated, it will be difficult to incorporate all of the conflicting updates. This is what we think causes both C-ADMM and ASV-ADMM to diverge in our experiment—neighboring updates become too dissimilar—and we expect similar pitfalls with highly irregular p even with ASV-ADMM.

Though the results indicate that ASV-ADMM only works better than C-ADMM in a handful of cases, it is almost always as performant as C-ADMM in highly switching communication networks. Still, we are surprised that the difference is so minor. We did evaluate the case where ASV-ADMM reduces to TV-ADMM on constant p-functions, but were not able to replicate the results of TV-ADMM [11] and did not notice significant differences between TV-ADMM and C-ADMM. We expect that C-ADMM, TV-ADMM, and ASV-ADMM are sensitive to hyperparameters and perhaps our chosen scenario does not differentiate between the two methods as clearly as possible. It also may be the case that differences only show on more complex scenarios or with N much larger than our small experiments.

Unfortunately, these simulations were run on a single node emulating independent multiagent behaviors. Since the probabilistic communication graph needs to check every (i, j) agent pair, the graph is quadratic in complexity and quickly becomes infeasible to solve without more compute. Moreover, consensus between agents becomes increasingly difficult to achieve as N changes or the complexity of the environment increases. For this reason, we limited M and N to be relatively small. Obstacles were not included in this experiment for similar computational complexity concerns—the line-of-sight calculation was rather costly especially when included in the quadratically many pairwise comparisons for communication failures.

7. Future Work and Conclusion

Communication-Exploration Tradeoff

A particularly interesting area of possible study, inspired by the spatial field estimation experiment, is balancing exploration with communication: How can agents explore uncharted territory (perhaps hindering communication) while efficiently propagating that information to other agents?

For example, suppose that we are using a distance-based formulation for p. There are two extremes for collecting samples as described in Section 4.1. They can all stick together and move around the space as one unit, in which case communication is not an issue, but this results in inefficient exploration. Alternatively, they could spread out as much as possible, perhaps according to Voronoi coverage control [19], but that may hinder communication, resulting in poorer global estimates. This is the communication-exploration tradeoff.

This question is not confined to the spatial field experiment and its sample collection procedure. When searching over possible optimization values θ in an efficient manner while balancing communication, and those values may affect communication, as though communication is endogenous, the same tradeoff applies.

This tradeoff has been explored often in the context of multi-agent coverage control [20], [21], [22], [23] but remains relatively unexplored in decentralized ADMM methods where the objective differs from coverage.

Game-Theoretic Formulation

One may apply game theory to the limited-communication decentralized setting. In addition to being a positive-sum game, where players seek to collectively minimize $J(\theta)$, there is an additional player. This player is perhaps adversarial and seeks to limit the communication of the other players to maximize $J(\theta)$.

Agents may also have surrogate objectives in addition to $J(\theta)$. For example, in the case of battery life, while they do want to minimize $J(\theta)$, they also have to stay alive enough to continue participating in the game and return to recharge. Perhaps communicating decreases J at the cost of battery life. Allowing agents to choose what "actions" to take (exploring, recharging, communicating, etc.) formulates a multi-player game, and game theory may be able to shed some light on optimal strategies.

Partial Communication Failures

Under ASV-ADMM's formulation, communication links at a given timestep are wholly present or absent. However, a more nuanced formulation would allow for partial failures in the communication links. For example, if there is some maximum bandwidth to communication links, the current formulation replaces that bandwidth with 0 upon communication failure; alternatively, one may only partially decrease the bandwidth to represent different levels of failure severity. The partial

communication failure formulation along with the possible game-theoretic formulation brings the ASV-ADMM formulation closer to that of congestion games in networks [24], [25].

Unknown p

A major drawback of ASV-ADMM is requiring p to be known a priori. This is a strong and often unrealistic assumption, as it is often impossible to capture everything that may influence the communication graph in p (or even X). Though one can certainly use ASV-ADMM with a simple model for p, including temporal or spatial factors, it is always assumed that p is known. Future research may handle the case where p is unknown and needs to be estimated in an online fashion, or an initial estimate is refined as agents experience more of the environment. This brings the formulation closer to reinforcement learning, where agents use their past experiences to determine what to do in the future.

Convergence Analysis

Though there is some empirical intuition about when and why ASV-ADMM is a good choice, more theoretical analysis is needed, particularly on convergence. There is no convergence analysis for TV-ADMM, so it is difficult to provide convergence analysis for ASV-ADMM. Many of the convergence analyses for C-ADMM break down due to the lack of temporal symmetry in the system when allowing for a dynamic communication network.

Reducing Network Traffic

When communication bandwidth is limited, compression techniques are often used to reduce network traffic. This has been applied to decentralized ADMM in [26], and could also be adapted to ASV-ADMM. Similar to the event triggers in [26], perhaps one could incorporate the communication probabilities into whether or not the updates are communicated.

7.1. Conclusion

Several primal-dual methods were derived in both the single- and multi-agent setting. The motivation to adapt C-ADMM to more dynamic communication networks inspired the definition of auxiliary states to capture state-dependent communication information. With this in mind, TV-ADMM was generalized to a novel algorithm, ASV-ADMM, designed for switching communication networks influenced by auxiliary states. ASV-ADMM was evaluated using a spatial field experiment that compared it to the baseline C-ADMM.

We found that the results of our experiments indicate that ASV-ADMM does exhibit marginally better empirical convergence properties than the baseline C-ADMM, even when the Bregman divergence term was incorporated into the C-ADMM update for a fairer comparison. However, we were surprised that it only outperformed C-ADMM in a small minority of cases. We suspected this to be due to hyperparameter tuning, but the cases where it did outperform C-ADMM are still promising. In particular, in highly dynamic networks that can be modeled as a function of an auxiliary state, ASV-ADMM appears to be a good alternative to C-ADMM, exhibiting better empirical convergence in some scenarios.

A. Appendix: Derivations

A.1. Derivation of Weak Duality

Consider the optimization problem

$$\min_{\theta \text{ s.t. } g(\theta)=0} J(\theta)$$

The Lagrangian is

$$\mathcal{L}(\theta, \lambda) = J(\theta) + \lambda^T g(\theta)$$

The primal problem is given by

$$\min_{\theta} \max_{\lambda} \mathcal{L}(\theta, \lambda)$$

and the dual problem is given by

$$\max_{\lambda} \min_{\theta} \mathcal{L}(\theta, \lambda)$$

Any feasible θ must satisfy the constraints $g(\theta) = 0$, so assuming θ is feasible, we have

$$\mathcal{L}(\theta, \lambda) = J(\theta) + \lambda^T g(\theta) = J(\theta)$$

For any λ and feasible θ, we then have

$$\min_{\theta'} \mathcal{L}(\theta', \lambda) \leq \mathcal{L}(\theta, \lambda) = J(\theta)$$

Since this holds for any λ and feasible θ, consider the dual optimum λ^* and the primal optimum θ^*, where θ^* is assumed to be feasible, i.e., $g(\theta^*) = 0$. Then

$$\max_{\lambda} \min_{\theta} \mathcal{L}(\theta, \lambda) = \min_{\theta} \mathcal{L}(\theta, \lambda^*) \leq J(\theta^*) = \min_{\theta \text{ s.t. } g(\theta)=0} J(\theta)$$

In other words, the dual optimum is always less than or equal to the primal optimum.

A.2. Derivation of Dual Concavity

The dual problem is given by

$$\max_{\lambda} \min_{\theta} \mathcal{L}(\theta, \lambda)$$

Define

$$f(\lambda) \triangleq \min_{\theta} \mathcal{L}(\theta, \lambda)$$

so the dual problem is

$$\max_{\lambda} f(\lambda)$$

Fix $\alpha \in [0,1]$ and λ, μ in the dual space.

Since $f(\lambda) = \min_{\theta} \mathcal{L}(\theta, \lambda)$, for any λ, we have $f(\lambda) \leq \mathcal{L}(\theta, \lambda)$, so for any $\theta \in \mathbb{R}^m$,

$$\begin{aligned}\mathcal{L}(\theta, \alpha\lambda + (1-\alpha)\mu) &= \alpha\mathcal{L}(\theta, \lambda) + (1-\alpha)\mathcal{L}(\theta, \mu) \\ &\geq \alpha f(\lambda) + (1-\alpha) f(\mu)\end{aligned}$$

Since this applies for any choice of θ, we have that

$$f(\alpha\lambda + (1-\alpha)\mu) \geq \alpha f(\lambda) + (1-\alpha) f(\mu)$$

i.e., f is concave. Then the dual problem $\max_{\lambda} f(\lambda)$ is a concave optimization problem.

A.3. Derivation of C-ADMM Updates

The following derivation is inspired by [16].

The C-ADMM problem formulation is as follows:

$$\begin{gathered}\min_{\theta_1, \ldots, \theta_N} \sum_{i=1}^{N} J_i(\theta_i) \\ \text{subject to } \theta_i = \theta_j \text{ for all } (i,j) \in \mathcal{E}\end{gathered}$$

In order to make the augmented Lagrangian easier to analyze, introduce the slack variables $r_{ij} \in \mathbb{R}^m$, and let

$$r_i \triangleq \begin{bmatrix} \vdots \\ r_{ij} \\ \vdots \end{bmatrix}_{j \in \mathcal{N}_i}$$

so that the original problem may be reformulated as follows:

$$\begin{gathered}\min_{\substack{r_1, \ldots, r_N \\ \theta_1, \ldots, \theta_N}} \sum_{i=1}^{N} J_i(\theta_i) \\ \text{subject to } \theta_i = r_{ij} \text{ and } \theta_j = r_{ij} \text{ for all } (i,j) \in \mathcal{E}\end{gathered}$$

Forming the augmented Lagrangian (where λ_{ij} are the dual variables for $\theta_i = r_{ij}$)

$$\begin{aligned}\mathcal{L}_a = \sum_{i=1}^{N} \Bigg(J_i(\theta_i) &+ \sum_{j \in \mathcal{N}_i} \left(\lambda_{ij}^T (\theta_i - r_{ij}) + \mu_{ij}^T (\theta_j - r_{ij}) \right) \\ &+ \frac{\rho}{2} \sum_{j \in \mathcal{N}_i} \left(\left\| \theta_i - r_{ij} \right\|^2 + \left\| \theta_j - r_{ij} \right\|^2 \right) \Bigg)\end{aligned}$$

Doing this allows the splitting of the optimization variable into two blocks $r \triangleq (r_1, ..., r_N)$ and $\theta \triangleq (\theta_1, ..., \theta_N)$. The ADMM updates as described in Equation 6 are then (where e represents the dual iterates)

$$r^{(t+1)} = \arg\min_r \mathcal{L}_a\big(\theta^{(t)}, r, e^{(t)}\big)$$
$$\theta^{(t+1)} = \arg\min_\theta \mathcal{L}_a\big(\theta, r^{(t+1)}, e^{(t)}\big)$$
$$e^{(t+1)} = e^{(t)} + \rho\nabla_e \mathcal{L}_a\big(r^{(t+1)}, \theta^{(t+1)}, e^{(t)}\big)$$

Consider how the $\arg\min$ may be derived in the updates for r and θ. Taking the gradient of $\mathcal{L}_a$ with respect to each variable and setting it to zero yields the minimum with respect to each variable.

$$\nabla_{r_{ij}} \mathcal{L}_a = -\big(\lambda_{ij} + \mu_{ij}\big) - \rho\big(\theta_i - r_{ij}\big) - \rho\big(\theta_j - r_{ij}\big) \overset{\text{set}}{=} 0$$
$$-\big(\lambda_{ij} + \mu_{ij}\big) - \rho\big(\theta_i + \theta_j\big) + 2\rho r_{ij} = 0$$
$$r_{ij} = \frac{\theta_i + \theta_j}{2} + \frac{\lambda_{ij} + \mu_{ij}}{2\rho}$$
$$r_{ji} = \frac{\theta_j + \theta_i}{2} + \frac{\lambda_{ji} + \mu_{ji}}{2\rho}$$

$$\begin{aligned}\nabla_{\lambda_{ij}} \mathcal{L}_a &= \theta_i - r_{ij} \\ &= \theta_i - \frac{\theta_i + \theta_j}{2} + \frac{\lambda_{ij} + \mu_{ij}}{2\rho} \\ &= \frac{\theta_i - \theta_j}{2} + \frac{\lambda_{ij} + \mu_{ij}}{2\rho}\end{aligned}$$

$$\begin{aligned}\nabla_{\mu_{ij}} \mathcal{L}_a &= \theta_j - r_{ij} \\ &= \theta_j - \frac{\theta_i + \theta_j}{2} + \frac{\lambda_{ij} + \mu_{ij}}{2\rho} \\ &= \frac{\theta_j - \theta_i}{2} + \frac{\lambda_{ij} + \mu_{ij}}{2\rho}\end{aligned}$$

Now suppose at a single timestep t, $\lambda_{ij}^{(t)} = -\mu_{ij}^{(t)}$. Then $\lambda_{ij}^{(t)} + \mu_{ij}^{(t)} = 0$, so

$$\frac{\theta_i - \theta_j}{2} = \nabla_{\lambda_{ij}} \mathcal{L}_a = -\nabla_{\mu_{ij}} \mathcal{L}_a$$

By the dual ascent update,

$$\lambda_{ij}^{(t+1)} = \lambda_{ij}^{(t)} + \rho \nabla_{\lambda_{ij}} \mathcal{L}_a$$

and

$$\begin{aligned}
\mu_{ij}^{(t+1)} &= \mu_{ij}^{(t)} + \rho \nabla_{\mu_{ij}} \mathcal{L}_a \\
&= -\lambda_{ij}^{(t)} - \rho \nabla_{\lambda_{ij}} \mathcal{L}_a \\
&= -\lambda_{ij}^{(t+1)}
\end{aligned}$$

So if $\lambda_{ij} + \mu_{ij} = 0$ at initialization (e.g. $\lambda_{ij}^{(0)} = 0 = -\mu_{ij}^{(0)}$), then $\lambda_{ij}^{(t)} + \mu_{ij}^{(t)} = 0$ for all time t.

The r_{ij} and r_{ji} updates via the $\arg\min$ then become

$$r_{ij}^{(t+1)} = \frac{\theta_i^{(t)} + \theta_j^{(t)}}{2} = r_{ji}^{(t+1)}$$

Now define the agent-specific augmented Lagrangian $\mathcal{L}_{a;i}$ so that each agent only optimizes over the optimization variable θ_i and its respective constraints such that

$$\begin{aligned}
\mathcal{L}_a &= \sum_{i=1}^{N} \mathcal{L}_{a;i} \\
\mathcal{L}_{a;i} &= J_i(\theta_i) + \sum_{j \in \mathcal{N}_i} \left(\lambda_{ij}^T (\theta_i - r_{ij}) + \mu_{ij}^T (\theta_j - r_{ij}) \right) + \frac{\rho}{2} \sum_{j \in \mathcal{N}_i} \left(\|\theta_i - r_{ij}\|^2 + \|\theta_j - r_{ij}\|^2 \right)
\end{aligned}$$

In the θ update, since θ is decomposable into $(\theta_1, ..., \theta_N)$, there will be an $\arg\min_{\theta_i} \mathcal{L}_a\left(\theta_i, r_i^{(t+1)}, e^{(t)}\right)$, so consider

$$\arg\min_{\theta_i} \mathcal{L}_a = \arg\min_{\theta_i} \left(\sum_{j=1}^{N} \mathcal{L}_{a;j} \right)$$

If $j \neq i$ and $j \notin \mathcal{N}_i$, then $\mathcal{L}_{a;j}$ will not affect the minimization over θ_i, so this reduces to

$$= \arg\min_{\theta_i} \left(\mathcal{L}_{a;i} + \sum_{j \in \mathcal{N}_i} \mathcal{L}_{a;j} \right)$$

Dropping the terms that are constant with respect to θ_i,

$$= \arg\min_{\theta_i} \left(\underbrace{J_i(\theta_i) + \sum_{j \in \mathcal{N}_i} \left(\left(\lambda_{ij}^{(t)}\right)^T \theta_i + \frac{\rho}{2} \left\| \theta_i - r_{ij}^{(t)} \right\|^2 \right)}_{\text{from } \mathcal{L}_{a;i}} + \sum_{j \in \mathcal{N}_i} \underbrace{\left(\left(\mu_{ji}^{(t)}\right)^T \theta_i + \frac{\rho}{2} \left\| \theta_i - r_{ji}^{(t)} \right\|^2 \right)}_{\text{from } \mathcal{L}_{a;j}} \right)$$

$$= \arg\min_{\theta_i} \left(J_i(\theta_i) + \sum_{j \in \mathcal{N}_i} \left(\left(\lambda_{ij}^{(t)}\right)^T \theta_i + \frac{\rho}{2} \left\| \theta_i - r_{ij}^{(t)} \right\|^2 + \left(\mu_{ji}^{(t)}\right)^T \theta_i + \frac{\rho}{2} \left\| \theta_i - r_{ji}^{(t)} \right\|^2 \right) \right)$$

$$= \arg\min_{\theta_i} \left(J_i(\theta_i) + \sum_{j \in \mathcal{N}_i} \left(\left(\lambda_{ij}^{(t)} + \mu_{ji}^{(t)}\right)^T \theta_i + \frac{\rho}{2} \left(\left\| \theta_i - r_{ij}^{(t)} \right\|^2 + \left\| \theta_i - r_{ji}^{(t)} \right\|^2 \right) \right) \right)$$

Since $r_{ij}^{(t)} = r_{ji}^{(t)} = \frac{1}{2}\left(\theta_i^{(t)} + \theta_j^{(t)}\right)$ for all t,

$$= \arg\min_{\theta_i} \left(J_i(\theta_i) + \sum_{j \in \mathcal{N}_i} \left(\left(\lambda_{ij}^{(t)} + \mu_{ji}^{(t)}\right)^T \theta_i + \rho \left\| \theta_i - \frac{\theta_i^{(t)} + \theta_j^{(t)}}{2} \right\|^2 \right) \right)$$

$$= \arg\min_{\theta_i} \left(J_i(\theta_i) + \left(\sum_{j \in \mathcal{N}_i} \left(\lambda_{ij}^{(t)} + \mu_{ji}^{(t)}\right) \right)^T \theta_i + \rho \sum_{j \in \mathcal{N}_i} \left\| \theta_i - \frac{\theta_i^{(t)} + \theta_j^{(t)}}{2} \right\|^2 \right)$$

and define the aggregate dual variable

$$e_i^{(t)} \triangleq \sum_{j \in \mathcal{N}_i} \left(\lambda_{ij}^{(t)} + \mu_{ji}^{(t)}\right)$$

so the θ_i update is

$$\theta_i^{(t+1)} = \arg\min_{\theta_i} \left(J_i(\theta_i) + \left(e_i^{(t)}\right)^T \theta_i + \rho \sum_{j \in \mathcal{N}_i} \left\| \theta_i - \frac{\theta_i^{(t)} + \theta_j^{(t)}}{2} \right\|^2 \right)$$

and since the inner product is commutative, this is equivalent to

$$\theta_i^{(t+1)} = \arg\min_{\theta_i} \left(J_i(\theta_i) + (\theta_i)^T e_i^{(t)} + \rho \sum_{j \in \mathcal{N}_i} \left\| \theta_i - \frac{\theta_i^{(t)} + \theta_j^{(t)}}{2} \right\|^2 \right)$$

For the dual (e_i) update:

$$
\begin{aligned}
\lambda_{ij}^{(t+1)} &= \lambda_{ij}^{(t)} + \rho \nabla_{\lambda_{ij}} \mathcal{L}_a = \lambda_{ij}^{(t)} + \frac{\rho}{2}\left(\theta_i^{(t+1)} - \theta_j^{(t+1)}\right) \\
\mu_{ji}^{(t+1)} &= \mu_{ji}^{(t)} + \rho \nabla_{\mu_{ji}} \mathcal{L}_a = \mu_{ji}^{(t)} + \frac{\rho}{2}\left(\theta_i^{(t+1)} - \theta_j^{(t+1)}\right) \\
e_i^{(t+1)} &= \sum_{j \in \mathcal{N}_i} \left(\lambda_{ij}^{(t+1)} + \mu_{ji}^{(t+1)}\right) \\
&= \sum_{j \in \mathcal{N}_i} \left(\lambda_{ij}^{(t)} + \frac{\rho}{2}\left(\theta_i^{(t+1)} - \theta_j^{(t+1)}\right) + \mu_{ji}^{(t)} + \frac{\rho}{2}\left(\theta_i^{(t+1)} - \theta_j^{(t+1)}\right)\right) \\
&= \sum_{j \in \mathcal{N}_i} \left(\lambda_{ij}^{(t)} + \mu_{ji}^{(t)}\right) + \rho \sum_{j \in \mathcal{N}_i} \left(\theta_i^{(t+1)} - \theta_j^{(t+1)}\right) \\
&= e_i^{(t)} + \rho \sum_{j \in \mathcal{N}_i} \left(\theta_i^{(t+1)} - \theta_j^{(t+1)}\right)
\end{aligned}
$$

Since many of the slack variables introduced are no longer needed in the explicit update, e is renamed to the more conventional λ for dual variables (distinct from the λ_{ij} used in this derivation), yielding the C-ADMM updates given in Section 2.4.1.

B. Appendix: Supplemental Figures

B.1. Two-Agent 1D Probability Function Visualizations

In the special case where $X = \mathbb{R}$ (so $n = 1$), only $2n + 1 = 3$ dimensions are required and p can be more easily visualized.

Note that these visualizations do **not** represent two-dimensional position ($x_i \in \mathbb{R}^2$)! It is simply a way to visualize the communication probability between two 1D agents as it is difficult to visualize agent communication in higher dimensions when neither agent is fixed.

B.1.1. Distance Thresholding

The first example is in Figure B.1, which illustrates communication probabilities given two agents with respective auxiliary states $x_1, x_2 \in \mathbb{R}$. In this case, diagonal lines parallel to $x_1 = x_2$ represent differing values for d_{ij} (so the nonnegative half of any cross-section using a line perpendicular to $x_1 = x_2$ is identical to Figure 3.3). The red dashed lines indicate the communication boundary when $d_{ij} = R = 1$.

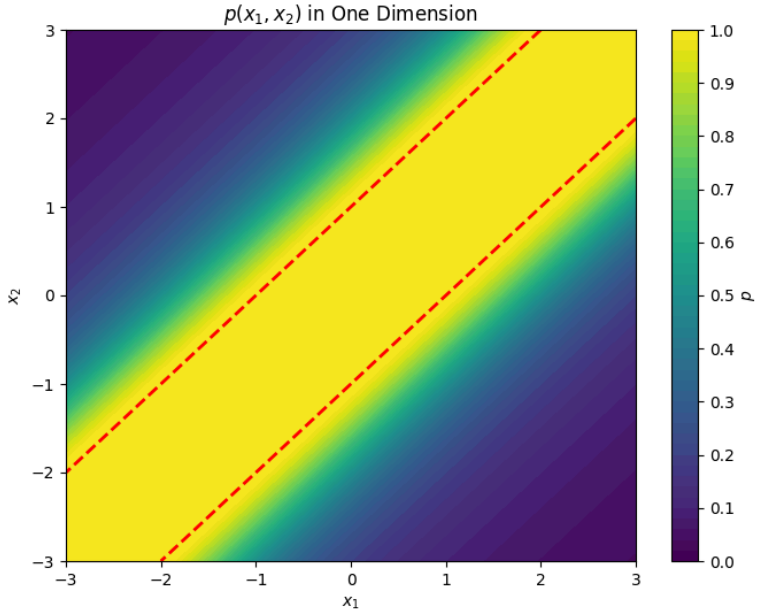


Figure B.1: A plot of a smoothed distance-thresholding p-function of two agents $x_1, x_2 \in \mathbb{R}$. Red dashed lines indicate the boundary at which communication begins to falter, i.e., p drops below 1.

B.1.2. Dead Zones

Once again, this is difficult to visualize when $X = \mathbb{R}^n$ and $n \geq 2$, though it can be done when $n = 1$. An example is shown in Figure B.2 where $X = \mathbb{R}$ (so $n = 1$) and $D = [-2, -1] \cup [0, 1] \subset X$. In this visualization, the distance falloff from Figure B.1 is still present, only dead zones can now completely block communication.

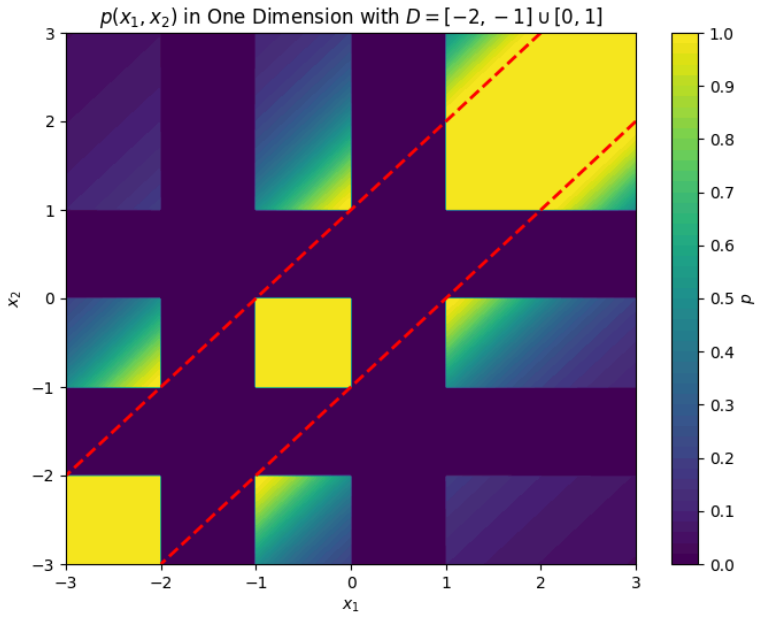


Figure B.2: A plot of a smoothed distance-thresholding p-function of two agents $x_1, x_2 \in \mathbb{R}$ with dead zones $D = [-2, -1] \cup [0, 1] \subset X$. Red dashed lines indicate the boundary at which communication begins to falter, i.e., p drops below 1.

B.1.3. Obstacles

Though the distinction from dead zones would be more interesting in higher dimensions, as mentioned earlier, it is difficult to visualize when $n \geq 2$. Still, similarly to before, when $X = \mathbb{R}$, it is possible to visualize the $n = 1$ case. In this case, both agents must lie in the same interval (unimpeded by the obstacle intervals in $O \subset \mathbb{R}$). Figure B.3 illustrates this using $O = [-2, -1] \cup [0, 1]$. The visualization again reuses the distance falloff from Figure B.1.

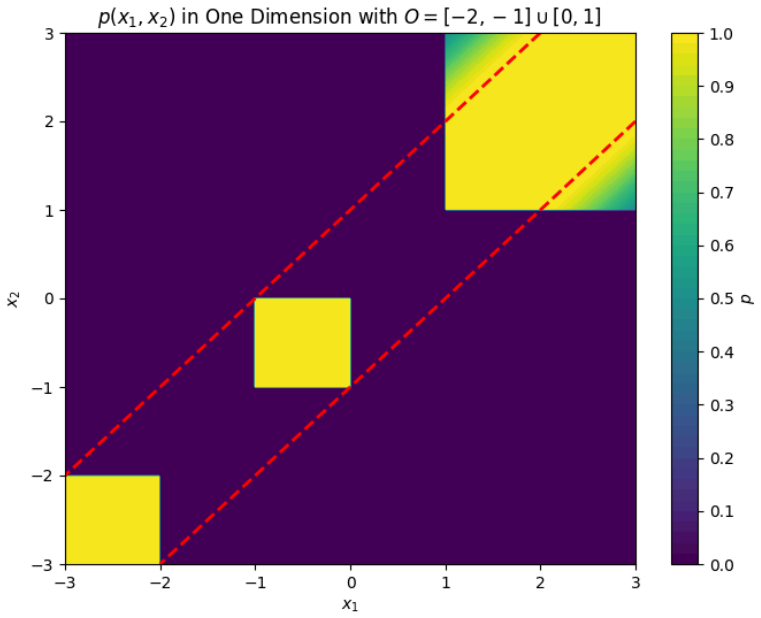


Figure B.3: A plot of a smoothed distance-thresholding p-function of two agents $x_1, x_2 \in \mathbb{R}$ with obstacles $O = [-2, -1] \cup [0, 1] \subset X$. Red dashed lines indicate the boundary at which communication begins to falter, i.e., p drops below 1.

Epilogue

All code used to run experiments and generate plots is available at https://github.com/noahadhikari/ME292B-proj.

This document was created using Typst. Unless you're extremely attached to *TeX, I highly recommend Typst due to its compactness, incremental compilation, more intuitive formatting, and general ease of use (and because its widespread adoption will only come with more users). The Typst source for this document is located at https://typst.app/project/rHNkBkgeLjibP8wcL36xE9.

References

[1] K. Rahul, R. Banyal, and N. Arora, "A systematic review on big data applications and scope for industrial processing and healthcare sectors," *Journal of Big Data*, vol. 10, p. , 2023, doi: 10.1186/s40537-023-00808-2.

[2] N. Elgendy and A. Elragal, "Big Data Analytics: A Literature Review Paper," 2014, pp. 214–227. doi: 10.1007/978-3-319-08976-8_16.

[3] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008, doi: 10.1145/1327452.1327492.

[4] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021, doi: https://doi.org/10.1016/j.knosys.2021.106775.

[5] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[6] E.-G. Talbi, *Metaheuristics: From Design to Implementation*, vol. 74. 2009, p. . doi: 10.1002/9780470496916.

[7] R. Tibshirani, "Convex Optimization 10-725 Lecture Slides," Carnegie Mellon University, 2019. [Online]. Available: https://www.stat.cmu.edu/~ryantibs/convexopt

[8] 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, doi: 10.1561/2200000016.

[9] F. Kuhn and R. Oshman, "Dynamic networks: models and algorithms," *SIGACT News*, vol. 42, no. 1, pp. 82–96, Mar. 2011, doi: 10.1145/1959045.1959064.

[10] I. Lobel, A. Ozdaglar, and D. Feijer, "Distributed multi-agent optimization with state-dependent communication," *Mathematical Programming*, vol. 129, no. 2, pp. 255–284, Jun. 2011, doi: 10.1007/s10107-011-0467-x.

[11] Z. Tian, Z. Zhang, and R. Jin, "Distributed ADMM for Time-Varying Communication Networks," in *2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)*, 2022, pp. 1–5. doi: 10.1109/VTC2022-Fall57202.2022.10012807.

[12] Z.-Y. Chen and Z.-P. Fan, "Distributed customer behavior prediction using multiplex data: A collaborative MK-SVM approach," *Know.-Based Syst.*, vol. 35, pp. 111–119, Nov. 2012, doi: 10.1016/j.knosys.2012.04.023.

[13] W. Crown *et al.*, "Application of Constrained Optimization Methods in Health Services Research: Report 2 of the ISPOR Optimization Methods Emerging Good Practices Task Force," *Value in Health*, vol. 21, no. 9, pp. 1019–1028, 2018, doi: https://doi.org/10.1016/j.jval.2018.05.003.

[14] Y. Su, W. Fan, L. Gao, L. Qiao, Y. Liu, and F. Wu, "Joint DNN Partition and Resource Allocation Optimization for Energy-Constrained Hierarchical Edge-Cloud Systems," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 3930–3944, 2023, doi: 10.1109/TVT.2022.3219058.

[15] P. T. Hultberg and D. S. Calonge, "Effective teaching of economics: A constrained optimization problem?," *The Journal of Economic Education*, vol. 48, no. 4, pp. 265–275, 2017, doi: 10.1080/00220485.2017.1353458.

[16] N. Mehr, "ME292B (Multi-Agent Control) Fall 2024 Lecture Notes," University of California, Berkeley, 2024. [Online]. Available: https://classes.berkeley.edu/content/2024-fall-meceng-292b-001-lec-001

[17] P. Zhao and T. Zhang, "Stochastic Optimization with Importance Sampling," 2015, [Online]. Available: https://arxiv.org/abs/1401.2753

[18] P. Virtanen *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020, doi: 10.1038/s41592-019-0686-2.

[19] M. Zhou, J. Li, C. Wang, J. Wang, and L. Wang, "Applications of Voronoi Diagrams in Multi-Robot Coverage: A Review," *Journal of Marine Science and Engineering*, vol. 12, no. 6, 2024, doi: 10.3390/jmse12061022.

[20] W. Luo and K. Sycara, "Voronoi-based Coverage Control with Connectivity Maintenance for Robotic Sensor Networks," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2019, pp. 148–154. doi: 10.1109/MRS.2019.8901078.

[21] C. Song and Y. Fan, "Coverage control for mobile sensor networks with limited communication ranges on a circle," *Automatica*, vol. 92, pp. 155–161, 2018, doi: https://doi.org/10.1016/j.automatica.2018.03.014.

[22] F. Pratissoli, B. Capelli, and L. Sabattini, "On Coverage Control for Limited Range Multi-Robot Systems," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 9957–9963. doi: 10.1109/IROS47612.2022.9982002.

[23] K. Nakamura, M. Santos, and N. E. Leonard, "Decentralized Learning With Limited Communications for Multi-robot Coverage of Unknown Spatial Fields." [Online]. Available: https://arxiv.org/abs/2208.01800

[24] M. Voorneveld, P. Borm, F. van Megen, S. Tijs, and G. Facchini, "Congestion Games and Potentials Reconsidered," *International Game Theory Review*, vol. 1, no. 3–4, pp. 283–299, 1999, doi: 10.1142/S0219198999000219.

[25] N. Bertrand, N. Markey, S. Sadhukhan, and O. Sankur, "Dynamic Network Congestion Games," in *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020)*, in Leibniz International Proceedings in Informatics (LIPIcs), vol. 182. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, pp. 1–23. doi: 10.4230/LIPIcs.FSTTCS.2020.40.

[26] Z. Zhang, S. Yang, and W. Xu, "Decentralized ADMM with compressed and event-triggered communication," *Neural Networks*, vol. 165, pp. 472–482, 2023, doi: https://doi.org/10.1016/j.neunet.2023.06.001.

[27] M. O. Jackson and Y. Zenou, "Games on Networks," *Handbook of Game Theory with Economic Applications*, vol. 4. Elsevier Science, pp. 95–163, Jul. 2014. [Online]. Available: https://ssrn.com/abstract=2148692

[28] G. Carnevale, N. Mimmo, and G. Notarstefano, "A Unifying System Theory Framework for Distributed Optimization and Games." [Online]. Available: https://arxiv.org/abs/2401.12623

[29] J. Li *et al.*, "Scenario-Game ADMM: A Parallelized Scenario-Based Solver for Stochastic Noncooperative Games." [Online]. Available: https://arxiv.org/abs/2304.01945

[30] N. S. Aybat and E. Y. Hamedani, "A distributed ADMM-like method for resource sharing over time-varying networks." [Online]. Available: https://arxiv.org/abs/1611.07393

[31] N. Bastianello, A. Simonetto, and R. Carli, "Distributed Prediction-Correction ADMM for Time-Varying Convex Optimization," in *2020 54th Asilomar Conference on Signals, Systems, and Computers*, 2020, pp. 47–52. doi: 10.1109/IEEECONF51394.2020.9443280.

[32] T.-H. Chang, "A Proximal Dual Consensus ADMM Method for Multi-Agent Constrained Optimization," *IEEE Transactions on Signal Processing*, vol. 64, no. 14, pp. 3719–3734, 2016, doi: 10.1109/TSP.2016.2544743.

[33] L. Cao, Y. Zheng, and Q. Zhou, "Consensus of Dynamical Agents in Time-Varying Networks," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10770–10775, 2008, doi: https://doi.org/10.3182/20080706-5-KR-1001.01826.

[34] S. Padakandla, "A Survey of Reinforcement Learning Algorithms for Dynamically Varying Environments," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–25, Jul. 2021, doi: 10.1145/3459991.