Negotiation Between Dynamical Systems with Connectivity Constraints

Yaniv Ben Shoushan

Negotiation Between Dynamical Systems with Connectivity Constraints

Research Thesis

Submitted in partial fulfilment of the requirements for the degree of Master of Science in Autonomous Systems and Robotics.

Yaniv Ben Shoushan

Submitted to the Senate of the Technion
Israel Institute of Technology.
Av , 5761 Haifa August 2016

This research thesis was done under the supervision of Assistant Professor Daniel Zelazo, from the Faculty of Aerospace Engineering.

I wish to thank my family, especially my wife and kids who stand by me at the hard times. I love you all and I couldn’t have done it without you.

Secondly I wish to thank my advisor, Professor Daniel Zelazo for the guidance and patience throughout my M.Sc, I leaned a lot.

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Multi-agent systems (MAS) have been an important area of study over the past few years. In this work we consider a MAS where a team of agents must coordinate to reach an agreement on their state in finite-time, while simultaneously attempting to minimize their own cost functions. Agents cooperate with each other by means of communication, and thus an additional requirement is the agents maintain their communication network during their trajectories - this leads to a constraint on the distances between neighbouring agents. We propose a real time sub-optimal solution for this agreement problem. The algorithm is based on the well-known dual sub-gradient algorithm for solving distributed optimization problems. To ensure the algorithm does not violate the distance connectivity constraints between neighbouring agents, we propose a modification to the sub-gradient algorithm that projects the Lagrange multipliers associated with the distance constraints onto a bounded set that ensures the distance between neighbouring agents remains within the prescribed limits. The main contribution for this work is an algorithm that solves the problem for the 2-agent case. We demonstrate the result using numerical simulations.

Abbreviations and Notations

Parameter	Size ,usage, Function
n	$\in \mathbb{R}$ The number of agents (constant)
T	$\in \mathbb{R}$, The OCP horizon constant
x_0/$x(t_0)$	$\in \mathbb{R}^{n\times 1}$, Agents initial position
t_0	$\in \mathbb{R}$, Optimization start time
t	$\in \mathbb{R}$, Optimization current time step
x	$x = [x_1^T, x_2^T, \dots, x_n^T]^T, \in \mathbb{R}^{n(T-t_0)\times 1}$, All agents trajectory position
u	$u = [u_1^T, u_2^T, \dots, u_n^T]^T, \in \mathbb{R}^{n(T-t_0)\times 1}$ All agents control
r_i	$\in \mathbb{R}$, Control weights of each agent
q_i	$\in \mathbb{R}$,State weight of each agent
ξ_i	$\in \mathbb{R}$, Preference state of each agent
$E(\mathcal{G})$	$\in \mathbb{R}^{n\times(n-1)}$, Incidence matrix of the graph
δ	$\in \mathbb{R}^{(n-1)(T-t_0)\times 1}$, Edges Lagrange multiplier - distance constraints
ζ	$\in \mathbb{R}^{(n-1)(T-t_0)\times 1}$, Edges Lagrange multiplier - distance constraints
μ	$\in \mathbb{R}^{(n-1)\times 1}$,Edges Lagrange multiplier - terminal constraints
$\overline{E}$	$\overline{E} = [E(\mathcal{G}) \otimes I_{(T-t_0)}], \in \mathbb{R}^{n(T-t_0)\times(n-1)(T-t_0)}$, Agents Incidence matrix of the graph at each time throughout the horizon
γ	$\gamma = E(\mathcal{G})\mu, \in \mathbb{R}^{n\times 1}$ Agents Lagrange multiplier - terminal constraints
λ	$\lambda = W\delta, \in \mathbb{R}^{n(T-t_0)\times 1}$, Agents Lagrange multiplier - distance constraints
β	$\beta = -W\zeta, \in \mathbb{R}^{n(T-t_0)\times 1}$, Agents Lagrange multiplier - distance constraints
$\mathbb{1}_N$	$\in \mathbb{R}^{N\times 1}$, All ones vector
$\mathbf{0}_N$	$\in \mathbb{R}^{N\times 1}$, All zeros vector
$\mathbb{R}_{\geq}$	The non-negative numbers
R	$\in \mathbb{R}$, The distance constraints radius
$\alpha_\delta, \alpha_\zeta, \alpha_\mu$	$\in \mathbb{R}$, Edges Lagrange multiplier step size rule - distance/terminal constraints
QP_i	The agents quadratic problem

Table 1: Nomenclature

Chapter 1

Introduction

A common goal of multi-agent systems is to reach an agreement on some global objective through cooperation and coordination. One of the most studied problems in this venue is known as the consensus, or agreement protocol [1, 16–18]. While the agreement protocol has drawn much attention in the controls community, its origins trace to problems in distributed computation and optimization problems [21, 22]. Recently, the optimization community have also employed consensus-based strategies for problems in distributed optimization [31–36]. A major distinction between the consensus protocol used in the controls community and the optimization community is that in the former the agents often refer to *physical* entities (i.e., robots), while in the later agents are processing nodes.

Despite these differences, there have been recent works that blend both the control of physical systems with the solution of related optimization problems. In [37], each agent negotiates a consensus value based on some cost function using a distributed optimization algorithm before controlling the physical system to that value. Dual decomposition is used in [38] for an optimal distributed controller design. In [39], a dual decomposition method is used to dynamically determine an optimal velocity for a team of self-interested agents.

Recently, the works in [3, 40, 43] considered a distributed optimization problem coupled to a physical control system. A team of self-interested agents are tasked with achieving consensus on their state at a specified finite time. The self-interest of each agent is modelled by a quadratic cost function penalizing its distance to a desired state and its control energy. The authors proposed a distributed algorithm based on a dual decomposition sub-gradient approach, that negotiates the consensus value in real-time. Each iteration of the algorithm corresponds to the real passing of time, and at each step the agents must physically propagate their state in a direction they believe to be optimal at that moment; this algorithm was termed the *shrinking horizon preference agreement* (SHPA) algorithm. An analysis of the resulting dynamic system, providing measures on the quality of their generated trajectories as compared to a centralized computation of their solution.

1.1 Thesis Contribution and Outline

An underlying assumption in these previous works was that the information exchange network was static. In real-world applications, however, it is more common to assume networks that are *state-dependent*. In particular, the ability to communicate between agents is dependent on the transmission radius of the radios, and thus the distance between agents [7, 12, 41]. It is therefore of great interest to impose connectivity constraints on the multi-agent system.

In this direction, we consider an extension to the works [3, 40, 43] where now the communication network given at the initial time must be preserved throughout the evolution of the system trajectory. We assume that connectivity between neighbouring agents is a function of the distance between them. We propose a modification to the SHPA algorithm proposed in [3] that will ensure neighboring agents do not lose connectivity with their initial neighbors. In this work, we present the problem for a team of n agents, but provide an algorithmic solution for the 2-agent case. As the algorithm is based on the dual-decomposition algorithm in optimization, we propose a modification where the multiplier value updates are projected onto a special set to ensure that the agents do not violate the connectivity constraint. Throughout the work, numerical simulations are provided to demonstrate the results.

The thesis is outlined as follows. In Chapter 2, we will briefly present the mathematical tools used in this work. Chapter 3 will present the main problem we aim to solve and demonstrate (by numerical simulations). Chapter 4 discuses distributed strategies for solving the problem at hand, including a real-time algorithm for the 2-agent case. Chapter 5 provides a concluding remark on this work.

Chapter 2

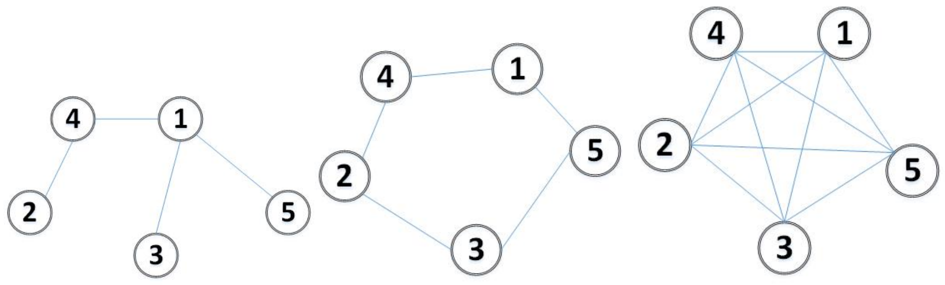
Mathematical Preliminaries

This section will provide background information regarding the mathematical methods which have been used in this research.

2.1 Graph Theory

A *graph* is a mathematical structure used to describe the relationships between objects. A graph, denoted $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is a pair consisting of a finite set of vertices $\mathcal{V} := \{v_1, \ldots, v_n\}$ and a set of edges $\mathcal{E} = \{e_1, \ldots, e_m\} \subseteq \mathcal{V} \times \mathcal{V}$. An undirected graph is a graph where the edges have no orientation, meaning the edge from $(v_j, v_k) = (v_k, v_j)$. A *spanning tree* is a minimally connected and undirected graph; thus, a spanning tree is a connected graph with $|\mathcal{E}| = n - 1$. The neighbourhood of a node v_i is defined as $N_i := \{v_j \in \mathcal{V} : (v_i, v_j) \in \mathcal{E}\}$. The complete undirected graph on n nodes, denoted K_n, is the graph where every node is connected to every other node. For undirected graphs, we omit the arrows in the pictorial representation of the graph. Examples of a spanning tree, cycle, and complete graph are shown in Figure 2.1.

A useful matrix associated with a graph $\mathcal{G}$ is the $n \times m$ incidence matrix, denoted $E(\mathcal{G})$ [2] - when the underlying graph $\mathcal{G}$ is understood, we simply write E. It is determined by the edges e_i of $\mathcal{G}$. The i^{th} column of E represents the edge e_i and has two non-zero entries: a $+1$ in row k and a -1 in row j, where e_i is the edge between vertex j and vertex k. This definition assumes that an arbitrary orientation (assignment of direction) is given to the graph. Thus, by definition, $E^T \mathbb{1} = 0$, where $\mathbb{1}$ is the vector with a 1 in each component. For the remainder of this thesis, all graphs are assumed to be connected and thus $Ker(E^T)$ is one



(a) A spanning tree. (b) A cycle graph. (c) Complete graph K_5.

Figure 2.1: Examples of undirected graphs.

dimensional [13]. In this work we consider only undirected graphs.

2.2 Convex Optimization

Optimization in general deals with finding the best solution to a problem subject to a set of constraints. Convex optimization problems are defined such that the objective function is convex and the constraint functions are convex. A convex function, $f : \mathbb{R}^n \rightarrow \mathbb{R}$, satisfies the inequality,

$$f(\alpha x + \beta u) \leq \alpha f(x) + \beta f(u),$$

for all $x, u \in \mathbb{R}^n$ and all $\alpha, \beta \in \mathbb{R}$, with $\alpha + \beta = 1$, $\alpha \geq 0, \beta \geq 0$. A convex optimization problem in standard form is expressed as,

$$\begin{aligned} p^* = \min_{x \in \mathbb{R}^n} \quad & f_0(x) \\ s.t. \quad & f_i(x) = 0,\ i = 1, \dots, \ell \\ & h_j(x) \leq 0,\ j = 1, \dots, m, \end{aligned} \tag{2.1}$$

where f_0, f_i, h_j are all convex scalar-valued functions.

For optimization problems in the form (2.1), one can define *Lagrange multipliers* for each of the constraints, and the corresponding *Lagrangian* function as,

$$\mathcal{L}(x, \lambda, \mu) = f_0(x, u) + \sum_{i=1}^{\ell} \lambda_i f_i(x) + \sum_{j=1}^{m} \mu_j h_j(x). \tag{2.2}$$

From the Lagrangian, the *dual function* and corresponding *dual problem* can be defined. The dual function is a function of the Lagrange multipliers, λ and μ, and is obtained by minimizing the Lagrangian over the variable x,

$$g(\lambda, \mu) = \min_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda, \mu). \tag{2.3}$$

The dual problem is the maximization problem

$$d^* = \max_{\lambda \in \mathbb{R}^\ell, \mu \in \mathbb{R}^m_\geq} \quad g(\lambda, \mu), \tag{2.4}$$

where $\mathbb{R}_\geq$ denotes the non-negative orthant. Furthermore due to the fact that the primal problem is convex, the dual function is concave and bounded from above, satisfying the inequality

$$d^* \leq p^*,$$

where equality is obtained when the primal problem is strictly convex and Slater's conditions hold (i.e., existence of a strictly feasible point) [15].

For convex optimization problems in the form (2.1), characterizations of the optimal solution are given by the first-order necessary conditions, known as the Karush-Kuhn-Tucker conditions (KKT) [15]. For primal and dual optimal values $\overline{x}, \overline{\lambda}, \overline{\mu}$, the KKT conditions can be stated as

$$\begin{array}{rcl}
f_i(\overline{x}) & \leq & 0,\ i = 1, \ldots, \ell \text{ (Primal Feasibility)} \\
h_j(\overline{x}) & = & 0,\ j = 1, \ldots, m \text{ (Primal Feasibility)} \\
\overline{\lambda} & \geq & 0,\ i = 1, \ldots, \ell \text{ (Dual Feasibility)} \\
\overline{\lambda}_i f_i(\overline{x}) & = & 0,\ i = 1, \ldots, \ell \text{ (Complementary Slackness)} \\
\nabla_x \mathcal{L}(\overline{x}, \overline{\lambda}, \overline{\mu}) & = & 0.
\end{array}$$

2.2.1 Quadratic Programming

This work will make extensive use a special class of convex programs known as the *quadratic program* (QP) [15]. The standard form of a quadratic program is given as,

$$\begin{aligned}
\min_x \quad & \tfrac{1}{2} x^T H x + c^T x \\
& A_{eq} x = b_{eq} \\
& A_{leq} x \leq b_{leq}
\end{aligned} \tag{2.5}$$

where $x \in \mathbb{R}^n$ is the optimization variable, H is a $n \times n$ positive-definite symmetric matrix, and $c \in \mathbb{R}^n$ is a vector. Quadratic programs have linear equality constraints specified by the matrices A_{eq} and b_{eq}, and inequality constraints specified by the matrices A_{leq} and b_{leq}.

An important feature of quadratic programs with only equality constraints is that the QP admits an analytic solution. For equality constrained quadratic programs, the solution can be obtained from the following linear equation,

$$\begin{bmatrix} H & A_{eq}^T \\ A_{eq} & 0 \end{bmatrix} \begin{bmatrix} \overline{x} \\ \overline{\lambda} \end{bmatrix} = \begin{bmatrix} -c \\ b_{eq} \end{bmatrix} \tag{2.6}$$

where $\overline{x}$ is the optimal vector of the quadratic program, and $\overline{\lambda}$ is the corresponding optimal Lagrange multiplier that comes from the Lagrangian function,

$$L(x, \lambda) = \frac{1}{2} x^T H x + c^T x + \lambda^T (A_{eq} x - b_{eq}).$$

When the matrix in (2.6) is invertible, the solution can be expressed as

$$\left\{ \begin{array}{rcl} \overline{x} & = & -H^{-1}c + H^{-1}A_{eq}^T \left(A_{eq}H^{-1}A_{eq}^T\right)^{-1} \left(b_{eq} + A_{eq}H^{-1}c\right) \\ \overline{\lambda} & = & -\left(A_{eq}H^{-1}A_{eq}^T\right)^{-1} \left(b_{eq} + A_{eq}H^{-1}c\right) \end{array} \right. . \tag{2.7}$$

Chapter 3

The Connectivity-Constrained Preference Agreement Problem

In this chapter, we formally introduce the preference agreement problem with connectivity constraints. We will formulate the problem as an optimal control problem, and present its solution from a centralized perspective. First, we will review the preference agreement problem with only terminal constraints, and then discuss how the connectivity constraints can be introduced. Numerical simulations will be provided to show how the resulting trajectories behave.

3.1 Preference Agreement with Terminal Constraints

We consider a group of n agents that aim to minimize individual objective functions while satisfying a terminal constraint where all agents must meet at an agreed point in space in finite time - the "agreement" constraint. Each agent is modeled as a simple discrete-time integrator,

$$x_i(t+1) = x_i(t) + u_i(t),\ x_i(0) = x_{i0}, i = 1, \dots, n, \tag{3.1}$$

where $x(t) = [x_1(t) \cdots x_n(t)]^T$ is the concatenated state vector for all n agents, and $u(t) = [u_1(t) \cdots u_n(t)]^T$ is the concatenated control. The self-interest of every agent is modeled by a quadratic objective function,

$$J_i(t_0, T, x_i, u_i) = \frac{1}{2}\left(\sum_{t=t0}^{T-1} q_i(x_i(t+1) - \xi_i)^2 + r_i u_i(t)^2\right), \tag{3.2}$$

where ξ_i is the preference state of agent i, $q_i > 0$ represents a state weight, and $r_i > 0$ represent the control weights. The terminal time constraint requires that all agents are in agreement at the final time T, meaning $x_1(T) = x_2(T) = \cdots = x_n(T)$. In this problem we assume that this final time is known by all agents and is given *a priori*. Note that the agreement constraint can be expressed compactly using the incidence matrix for the complete graph, $E(K_n)$, as

$$E(K_n)^T x(T) = 0,$$

since this requires $x_i - x_j = 0$ for all possible agent pairs. Observe, however, that the representation above includes redundant constraints - that is, the constraints are linearly dependent. We can express the terminal time constraint equivalently with a minimal number of equations as $E^T(\mathcal{T})x(T) = 0$, where $\mathcal{T}$ is any spanning tree. In fact, this equivalent representation will become important in the sequel when we consider connectivity constraints on the agents.

The optimal control problem (OCP) can now be stated as,

$$\begin{aligned} OCP(t_0, T, x_0): \quad \min_{x,u} \quad & \sum_{i=1}^{n} J_i(t_0, T, x_i, u_i) \\ s.t. \quad & x(t+1) = x(t) + u(t), x(t_0) = x_0 \\ & E^T(\mathcal{T})x(T) = 0. \end{aligned} \tag{3.3}$$

The OCP with terminal constraints can be transformed into a static quadratic program. The complete state trajectory of each agent is denoted as $x_i = [x_i(t_0 + 1) \; \cdots \; x_i(T)]^T$, and the control of each agent is denoted as $u_i = [u_i(t_0) \; \cdots \; u_i(T-1)]^T$. The complete trajectory and control of all agents can thus be combined into the single vector, $\mathbf{x} = [x_1^T \; \cdots \; x_n^T]^T$ and $\mathbf{u} = [u_1^T \; \cdots \; u_n^T]^T$. We can state the dynamic constraint as the linear equation,

$$x_i \quad = \quad \mathbb{1} x_{i0} + B_{T-t_0} u_i, \tag{3.4}$$

where $B_{T-t_0} \in \mathbb{R}^{(T-t_0)\times(T-t_0)}$ is defined as

$$[B_{T-t_0}]_{i,j} = \left\{ \begin{array}{ll} 1, & i \geq j \\ 0, & o.w. \end{array} \right. .$$

Let $z = [\mathbf{x}^T \, \mathbf{u}^T]^T$ be a new vector which holds the state and controls of all the agents for the entire horizon. Let

$$Q_x = \begin{pmatrix} q_1 I_T & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & q_n I_T \end{pmatrix}, \quad R_u = \begin{pmatrix} r_1 I_T & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & r_n I_T \end{pmatrix}, \quad H = \begin{pmatrix} Q_x & \mathbf{0} \\ \mathbf{0} & R_u \end{pmatrix},$$

and define $f \in \mathbb{R}^{2n(T-t_0)}$ as

$$f = \begin{pmatrix} -q_1\xi_1\mathbb{1}_{T-t_0} \\ \vdots \\ -q_n\xi_n\mathbb{1}_{T-t_0} \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

The equality constraints can be defined as,

$$\begin{aligned} A_{eq} &= \begin{bmatrix} E(\mathcal{T})^T \otimes e_{T,T}^T & \mathbf{0}_{(n-1)\times(T-t_0)} \\ -I_{(T-t_0)n} & I_n \otimes B_{T-t_0} \end{bmatrix} \in \mathbb{R}^{(n-1)+n(T-t_0)\times 2n(T-t_0)} \\ b_{eq} &= \begin{bmatrix} \mathbf{0}_{n-1} \\ -x_1(0)\mathbb{1}_{(T-t_0)n} \\ \vdots \\ -x_n(0)\mathbb{1}_{(T-t_0)n} \end{bmatrix} \in \mathbb{R}^{(n-1)+n(T-t_0)}, \end{aligned}$$

where $e_{i,m} \in \mathbb{R}^m$, $[e_{i,m}]_j = 1$, if $j = i$, otherwise $[e_{i,m}]_j = 0$. Note that the first block-row of A_{eq} deals with the terminal constraints, and the 2nd block-row deals with the dynamic constraints. The notation $\otimes$ denotes the matrix Kronecker product [42].

Using the above definitions, we can now express (3.3) as the quadratic program,

$$\begin{aligned} \min_{z} \quad & \frac{1}{2}z^T Hz + f^T z \\ s.t. \quad & A_{eq}z = b_{eq}. \end{aligned}$$

Now we can solve the OCP with terminal constraints defined in equation (3.3) using an analytic solution as in (2.7), or using a numerical solver (such as a `quadprog` in MATLAB). Here, we emphasize that the solution methods for solving (3.3) are computed using a *centralized* approach. In this setting, we are assuming that there is a centralized coordinator that has access to the parameters of the entire team (i.e., the cost functions and initial conditions). This coordinator then solves the OCP and provides to each agent its optimal *open-loop* control. It is important to note that in this setting there is no explicit coordination between agents. This solution, however, represents the *optimal* trajectories and will serve as a benchmark for the distributed algorithms we will provide in the sequel. In the following, we denote by $\overline{\mathbf{x}}$ and $\overline{\mathbf{u}}$ the optimal trajectory and control generated by OCP.

3.1.1 Numerical Example

This section provides a simulation example for the OCP with the terminal constraints defined in (3.3). In this example, we consider $n = 8$ agents, and terminal time of $T = 30$ seconds. While the choice of the spanning tree used to define the terminal time constraint is arbitrary, we employ the graph in Figure 3.1. The agent preferences, state and control weights, and initial conditions are given as,

$$r = \begin{bmatrix} 10 \\ 6 \\ 1 \\ 4 \\ 2 \\ 8 \\ 5 \\ 2 \end{bmatrix}, \quad q = \begin{bmatrix} 7 \\ 7 \\ 4 \\ 5 \\ 8 \\ 5 \\ 9 \\ 4 \end{bmatrix}, \quad \xi = \begin{bmatrix} -26 \\ 42 \\ 48 \\ -9 \\ -36 \\ -8 \\ -45 \\ 16 \end{bmatrix}, \quad x_0 = \begin{bmatrix} 0.4710 \\ 6.0231 \\ -20.4554 \\ 7.8192 \\ -20.7754 \\ -25.1874 \\ -13.9426 \\ -0.2915 \end{bmatrix}.$$

The optimal trajectories were found in MATLAB using the quadratic programming formulation. Figure 3.2 shows the resulting trajectories. We can see that each agent is pursuing its preference, while at the terminal time T, all of the agents reach a common meeting point ($x(T) = -8.3407\mathbb{1}$).

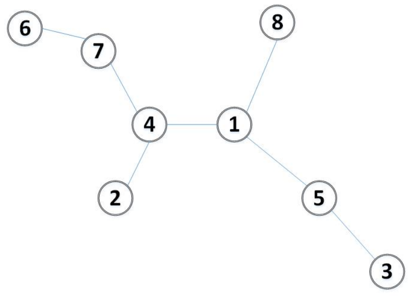
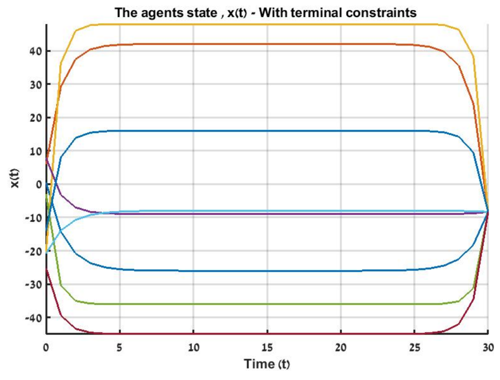
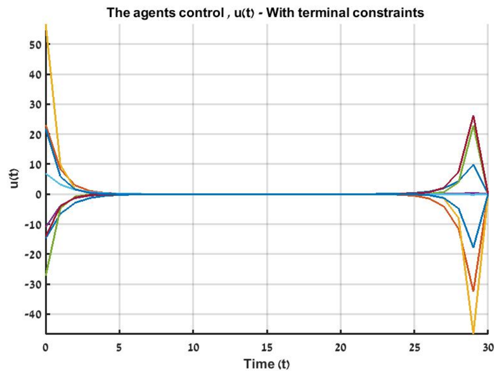


Figure 3.1: A spanning tree on 8 nodes.

In the next example, we used the same parameters as in the previous simulation, but change the final agreement time to $T = 3$. Here we observe that in order to satisfy the terminal agreement constraint the agents are not able to obtain the minimum value of their individual objective functions (i.e., reach their preference state).

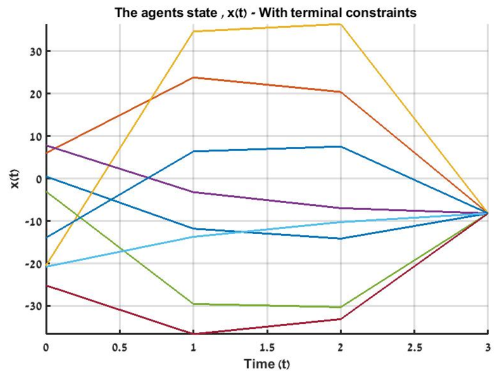


(a) Trajectories of $\overline{\mathbf{x}}$ for OCP.

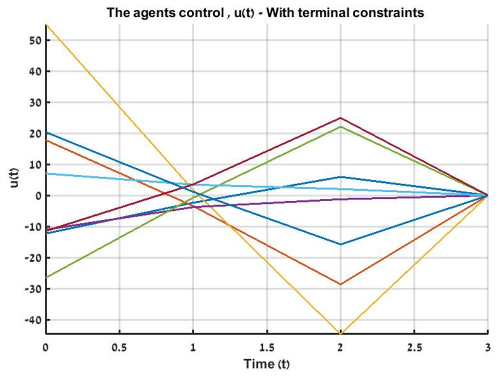


(b) Control $\overline{\mathbf{u}}$ for OCP.

Figure 3.2: Optimal trajectories for the optimal control problem (3.3).



(a) Trajectories of $\overline{\mathbf{x}}$ for OCP with short T.



(b) Control $\overline{\mathbf{u}}$ for OCP with short T.

Figure 3.3: Optimal trajectories for the optimal control problem (3.3) with short T.

At Figure 3.3a we can see that each agent is trying to reach its preference. When comparing the Figures 3.2a and 3.3a, we can see that when the terminal time is large enough the agents get to their preference value. While, in the case of short terminal time the agents may not get to their preference. Hence, the parameters we choose have a big influence on the trajectory.

3.2 Preference Agreement with Terminal and Connectivity Constraints

We now propose the main problem we aim to examine in this work. We recall here the primary motivation for this work is for a team of agents to simultaneously attempt to minimize their individual cost functions while satisfying a terminal time agreement constraint, along with a connectivity constraint between each other. Before we proceed, we elaborate on this last point.

The solution to the OCP in (3.3) was found in a centralized manner. As discussed in the previous works [3], we would actually like to solve this problem distributedly and in real-time. This will in fact require communication between each agent. Thus, we assume that we have in place a fixed communication graph defined by a spanning tree. This graph represents the network that agents are permitted to communicate with each other in order to cooperatively solve the control problem at hand. As seen in the simulations examples from Chapter 3.1, the optimal solution might cause agents to be too far away from each other to maintain communication (i.e., if their distance exceeds the communication radius between them). Thus, we impose an additional constraint on the optimization problem that ensures the agents do not violate a prescribed distance constraint - this is what we call the *connectivity constraint* for the network. This constraint in fact becomes crucial when we examine distributed and real-time algorithms in Chapter 4. We also recall now that the agreement constraint can be expressed using any spanning tree - we therefore chose the same spanning tree that describes the connectivity to also describe the agreement constraint.

In this direction, we are now ready to present the preference agreement problem with terminal and connectivity constraints. The associated optimal control problem can be stated as

$$\begin{aligned} OCP(t_0, T, x_0): \quad \min_{x,u} \quad & \sum_{i=1}^{n} J_i(t_0, T, x_i, u_i) \\ s.t. \quad & x(t+1) = x(t) + u(t), \; x(t_0) = x_0 \\ & -R\mathbb{1} \leq E^T x(t) \leq R\mathbb{1}, t = t_0, \dots, T \\ & E^T x(T) = 0, \end{aligned} \tag{3.5}$$

Where $R > 0$ is the communication radius of each agent, and E is the incidence matrix of a given and fixed spanning tree. Note that the notation $x \leq y$ for vectors $x, y \in \mathbb{R}^n$ means the inequality should be satisfied by each component of the vector, i.e., $x_i \leq y_i$ for $i = 1, \dots, n$.

As in Chapter 3.1, (3.5) can be expressed as a quadratic program with equality and inequality constraints. The QP representation has the same cost function expression and equality constraints described in (3.3). The inequality constraints can be expressed as:

$$\begin{aligned} A_{ieq} &= \begin{bmatrix} E^T \otimes I_{T-t_0} & \mathbf{0}_{(n-1)(T-t_0)\times n(T-t_0)} \\ -E^T \otimes I_{T-t_0} & \mathbf{0}_{(n-1)(T-t_0)\times n(T-t_0)} \end{bmatrix} \in \mathbb{R}^{2(n-1)(T-t_0)\times 2n(T-t_0)} \\ b_{ieq} &= R\mathbb{1}_{2(n-1)(T-t_0)}. \end{aligned}$$

Using the above definitions, we can now express (3.5) as the quadratic program,

$$\begin{aligned} \min_z \quad & \frac{1}{2} z^T H z + f^T z \\ s.t. \quad & A_{eq} z = b_{eq} \\ & A_{ieq} z \leq b_{ieq}. \end{aligned}$$

Now we can solve the OCP with terminal and distance constraints defined in (3.5) using a solver (such as a `quadprog` in Matlab). Unlike the OCP with terminal constraints, we can not use an analytical solution since the OCP with terminal and distance constraints holds inequality constraints. As before, this problem is solved *centrally* and is meant to serve as a performance benchmark for when we provide distributed algorithms to solve this problem. In the following, we denote by $\overline{\mathbf{x}}$ and $\overline{\mathbf{u}}$ the optimal trajectory and control generated by OCP.

3.2.1 The Dual Problem

We now examine the dual problem associated with (3.5). The dual formulation will be required when we consider distributed solutions for (3.5), discussed in Chapter 4. We define the *partial* Lagrangian function associated with (3.5) as

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \mathbf{u}, \delta, \zeta, \mu) &= \sum_{i=1}^{n} J_i(t_0, T, x_i, u_i) + \mu^T E^T (I_n \otimes e_{T,T})^T \mathbf{x} + \\ & \quad \delta^T \left(\overline{E}^T \mathbf{x} - R\mathbb{1} \right) + \zeta^T \left(-\overline{E}^T \mathbf{x} - R\mathbb{1} \right), \end{aligned} \tag{3.6}$$

Here, μ is the Lagrange multiplier associated with the terminal constraint, and δ and ζ are the Lagrange multipliers associated with the edges distance constraints. The vector $e_{T,T} \in \mathbb{R}^T$ has value 1 in the T position, and 0 elsewhere. The matrix

$\overline{E} = E \otimes I_{T-t_0} \in \mathbb{R}^{n(T-t_0)\times(n-1)(T-t_0)}$ is an inflated version of the incidence matrix. The dual function is obtained by minimizing the partial Lagrangian subject to the dynamical constraints,

$$g(\mu,\zeta,\delta) = \min_{\mathbf{x},\mathbf{u}} \quad \mathcal{L}(\mathbf{x},\mathbf{u},\delta,\zeta,\mu) \tag{3.7}$$

$$\text{s.t.} \quad x(t+1) = x(t) + u(t), x(t_0) = x_0. \tag{3.8}$$

The dual problem is obtained by maximizing the dual function,

$$\max_{\substack{\mu\in\mathbb{R}^{n-1} \\ \zeta\in\mathbb{R}_{\geq}^{(n-1)(T-t_0)} \\ \delta\in\mathbb{R}_{\geq}^{(n-1)(T-t_0)}}} \quad g(\mu,\zeta,\delta). \tag{3.9}$$

Here, $\mathbb{R}_{\geq}$, represents the non-negative orthant. The optimal solution of the primal and dual problems are denoted by $(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{\mu},\bar{\delta},\bar{\zeta})$. Since OCP($t_0,T,x_0$) is a convex problem with linear constraints, we have strong duality which implies

$$g(\bar{\mu},\bar{\zeta},\bar{\delta}) = \sum_{i=1}^{n} J_i(t_0,T,x_i,u_i). \tag{3.10}$$

This work will be based on the understanding of how the OCP behaves once the initial conditions are changed, and how that will affect the rest of the trajectory. For a given initial time t and initial condition $x(t)$, we denote the optimal primal and dual solutions associated with the $OCP(t,T,x(t))$ by

$$(\bar{x}^{(t,x(t))},\bar{u}^{(t,x(t))},\bar{\mu}^{(t,x(t))},\bar{\zeta}^{(t,x(t))},\bar{\delta}^{(t,x(t))}). \tag{3.11}$$

Thus, for example, $\bar{x}^{(t,x(t))}$ denotes the optimal state trajectory beginning at time t with initial condition $x(t)$. Using this notation we will state a principle of optimality result for dynamic programming in the context of OCP to study the relation between the optimal trajectories of different instances of OCP.

Lemma 1. *(Principle of optimality) . The optimal trajectories generated by the $OCP(t,T,z)$ and $OCP(t+1,T,w)$ with $w = z + \bar{u}^{(t,z)}(t)$ satisfy*
$(\bar{x}^{(t,z)}(\tau),\bar{u}^{(t,z)}(\tau),\bar{\mu}^{(t,z)}(\tau),\bar{\zeta}^{(t,z)}(\tau),\bar{\delta}^{(t,z)}(\tau)) =$
$(\bar{x}^{(t+1,w)}(\tau),\bar{u}^{(t+1,w)}(\tau),\bar{\mu}^{(t+1,w)}(\tau),\bar{\zeta}^{(t+1,w)}(\tau),\bar{\delta}^{(t+1,w)}(\tau))\,,\ \tau = t+1\ldots T.$

Proof. Concerning the primal solution, the initial condition for $OCP(t+1,T,w)$ corresponds to the point $\bar{x}^{(t,z)}(t+1)$. The remaining statement is a direct application of the principle of optimality for dynamic programming [5], and its uniqueness is due to the strict convexity of the problem statement. The statement concerning the dual solution is a direct consequence of the first statement, in particular, we have $\bar{x}^{(t,z)}(T) = \bar{x}^{(t+1,w)}(T)$. A necessary condition for optimality (the KKT conditions) is

$$0 = \frac{\partial}{\partial \bar{x}(T)}\mathcal{L}(\mathbf{x}, \mathbf{u}, \delta, \zeta, \mu) \quad = \quad Q(\bar{x}(T) - \xi) + E\overline{\mu} + \overline{E}\delta - \overline{E}\zeta.$$

This system of n equations admits a unique solution since the incidence matrix E is full column-rank (the communication graph is a tree). Similarly,

$$\begin{aligned} 0 = \frac{\partial}{\partial \bar{x}(t)}\mathcal{L}(\mathbf{x}, \mathbf{u}, \delta, \zeta, \mu) \quad &= \quad Q(\bar{x}(t) - \xi) + (I_n \otimes e_{t,T})^T\overline{E}\delta + \\ &\quad -(I_n \otimes e_{t,T})^T\overline{E}\zeta, \ t = t_0, \ldots, T-1, \end{aligned}$$

which also admits a unique solution.

□

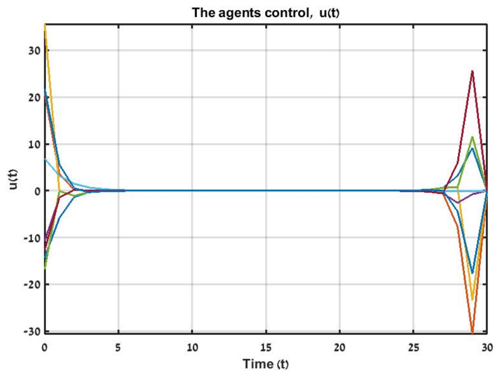
3.2.2 Numerical Example

This section provides a simulation example for the OCP with the distance and terminal constraints defined in (3.5). In this example, we consider $n = 8$ agents and a terminal time of $T = 30$ seconds. The radius defined is $R = 35$ with communication graph shown in Figure 3.1. For easy comparison we ran the OCP with distance constraints with the same parameters and initial states as in the OCP with terminal constraints numerical examples.

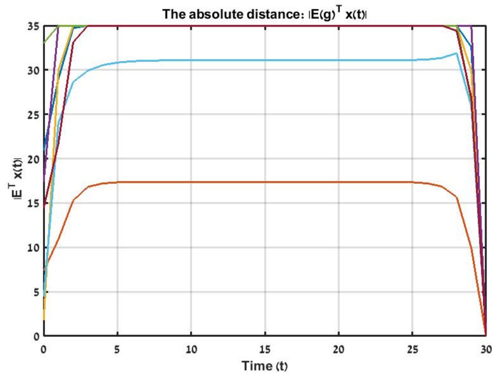
The optimal trajectories were found in MATLAB using the quadratic programming formulation. Figures 3.4a and 3.5 shows the resulting trajectories. We can see that each agent pursuing its preference, while at the terminal time T, all of the agents go to a common meeting point ($x(T) = -8.3407\mathbb{1}$). In Figure 3.5 we can see that each agent is trying to pursue its preference value while they are satisfying the distance constraint between adjacent agents for the entire trajectory.

As we can see in Figure 3.4b, there are 6 edges where the connectivity constraint is active. In this setting, each agent was not able to obtain the minimum of its individual objective in order to ensure the feasibility of the constraints.

In Figure 3.5 we can see the different trajectories applied to the agents in the case of terminal constraints alone (the solid lines) and compared to the optimal trajectories of the agents in the case of terminal and connectivity constraints (the dashed lines). We can see that in the case of terminal constraints, the agents are



(a) Trajectories of $\overline{\mathbf{u}}$ for OCP (3.5).



(b) A plot of $|E^T\mathbf{x}(t)|$ for OCP (3.5).

Figure 3.4: Optimal trajectories for the optimal control problem (3.5).

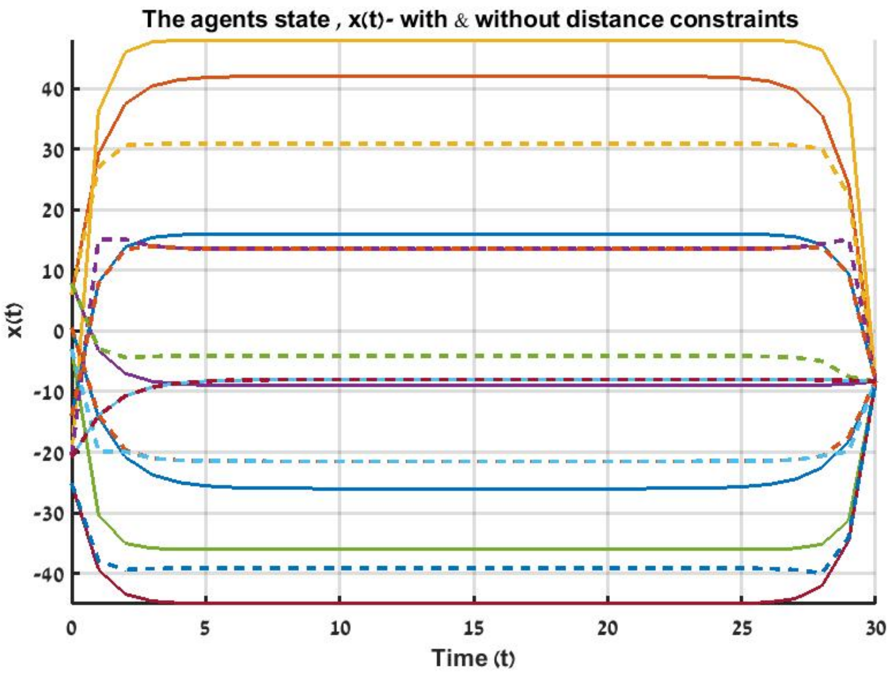


Figure 3.5: The OCP with terminal and distance constraints (3.5) vs the OCP with terminal constraints (3.3); a plot of $\bar{x}$.

moving towards their preference. In the case of the distance constraints, we see that the agents trajectories are bounded according to the distance constraints.

Chapter 4

Distributed and Finite-time Algorithms

The previous chapter presented the preference agreement problem as an optimal control problem. The optimal open-loop controls were then computed in a *centralized* manner by a centralized coordinator. The main goal of this work, however, is to develop a *distributed* and *real-time* solution to this problem. That is, agents should communicate with each other to determine their optimal trajectories. Furthermore, we are assuming that the agreement time T is a *hard deadline*, and therefore the agents are not able to wait for the optimal control to be computed, but rather at each time step should already begin to move along a trajectory it considers optimal at that time. Our solution approach is based on the *shrinking horizon preference agreement problem* (SHPA) introduced in [3] which considers a modification of the dual sub-gradient algorithm for distributedly solving problems of the form (3.3). In this section, we first review the dual decomposition sub-gradient algorithm for distributedly solving (3.5), and then present the main result of this work which is an extension to the SHPA algorithm in order to handle the distance constraints imposed by the problem.

4.1 Dual Decomposition Sub-Gradient Algorithm

A decomposition method for solving (3.5) is an algorithm that attempts to divide the main problem into smaller sub-problems that coordinate with each other to solve the original problem. In this section, we review a dual decomposition sub-gradient algorithm [5] for distributedly solving (3.5).

As described in Chapter 3, the OCP (3.5) can be solved using centralized methods. The challenge in finding a distributed solution that can be solved by each agent is that the constraints couple neighboring agents together. Note however, that the objective function is already in a separable form - that is,

$$J(t_0, T, x, u) = \sum_{i=1}^{n} J_i(t_0, T, x_i, u_i),$$

is the sum of each agent's individual objective. The challenging part of OCP (3.5) is the connectivity constraint and the terminal time agreement constraint which couples agents together.

Observe that the Lagrangian function associated with (3.5) is also *not* separable,

$$\begin{aligned}\mathcal{L}(\mathbf{x}, \mathbf{u}, \delta, \zeta, \mu) &= \sum_{i=1}^{n} J_i(t_0, T, x_i, u_i) + \mu^T E^T (I_n \otimes e_{T,T})^T \mathbf{x} + \\ &\quad \delta^T \left(\overline{E}^T \mathbf{x} - R\mathbb{1}\right) + \zeta^T \left(-\overline{E}^T \mathbf{x} - R\mathbb{1}\right).\end{aligned}$$

In particular, the multipliers μ, δ, ζ are associated with the *edges* in the graph. However, exploiting the properties of the incidence matrix, we define a new variable that can be associated with the nodes in the graph. In particular, let

$$\left\{\begin{array}{lcl} \gamma & := & E\mu \\ \lambda & := & \overline{E}\delta \\ \beta & := & \overline{E}\zeta \end{array}\right. . \tag{4.1}$$

This transformation is unique, since E (and also $\overline{E}$) has full column-rank. Consequently, it is also possible to transform the node variables back to edges with the following expression,

$$\mu = (E^T E)^{-1} E^T \gamma. \tag{4.2}$$

Similar transformations can be used for δ and ζ.

After this variable transformation, we can now write the partial Lagrangian in a separable form,

$$\begin{aligned}\mathcal{L}(\mathbf{x}, \mathbf{u}, \delta, \zeta, \mu) &= \sum_{i=1}^{n} \tilde{\mathcal{L}}_i(x_i, u_i, \lambda_i, \beta_i, \gamma_i) - R(\delta + \zeta)^T \mathbb{1}, \qquad (4.3)\\ \tilde{\mathcal{L}}_i(x_i, u_i, \lambda_i, \beta_i, \gamma_i) &= J_i(t_0, T, x_i, u_i) + \gamma_i e_{T,T}^T x_i + (\lambda_i - \beta_i)^T x_i.\end{aligned}$$

The dual function is found by minimizing the Lagrangian subject to the dynamic constraints,

$$\begin{aligned} g(\mu,\zeta,\delta) &= \min_{\mathbf{x},\mathbf{u}} \mathcal{L}(\mathbf{x},\mathbf{u},\delta,\zeta,\mu) \\ &\quad s.t.\ x(t+1) = x(t) + u(t),\ x(t_0) = x_0 \\ &= \left(\min_{x_i,u_i} \sum_{i=1}^{n} \tilde{\mathcal{L}}_i(x_i,u_i,\lambda_i,\beta_i,\gamma_i) \right) - R(\delta+\zeta)^T \mathbb{1} \qquad (4.4) \\ &\quad s.t.\ \begin{bmatrix} I & -B_{T-t_0} \end{bmatrix} \begin{bmatrix} x_i \\ u_i \end{bmatrix} = \mathbb{1} x_{i0},\ i = 1,\ldots,n. \end{aligned}$$

Observe that

$$\arg\min_{\mathbf{x},\mathbf{u}} \mathcal{L}(\mathbf{x},\mathbf{u},\delta,\zeta,\mu) = \arg\min_{x_i,u_i} \sum_{i=1}^{n} \tilde{\mathcal{L}}_i(x_i,u_i,\lambda_i,\beta_i,\gamma_i). \qquad (4.5)$$

since the constant term $-R(\delta+\zeta)^T \mathbb{1}$ does not affect the minimization (it is constant). Therefore, when deriving the sub-gradient algorithm, we may only consider minimization over $\sum_{i=1}^{n} \tilde{\mathcal{L}}_i(x_i,u_i,\lambda_i,\beta_i,\gamma_i)$, which is separable across each agent.

We are now prepared to describe the dual-decomposition sub-gradient algorithm. At each iteration of the algorithm, each agent computes the dual function for a fixed value of the multiplier. So for iteration k, agent i solves the problem

$$\begin{aligned} \begin{bmatrix} \overline{x}_i^{[k+1]} & \overline{u}_i^{[k+1]} \end{bmatrix} &= \arg\min_{x_i,u_i} \tilde{\mathcal{L}}_i(x_i,u_i,\lambda_i^{[k]},\beta_i^{[k]},\gamma_i^{[k]}) \\ &\quad s.t.\ \begin{bmatrix} I & -B_{T-t_0} \end{bmatrix} \begin{bmatrix} x_i \\ u_i \end{bmatrix} = \mathbb{1} x_{i0} \\ &= \arg\min_{x_i,u_i} \left(J_i(t_0,T,x_i,u_i) + \gamma_i^{[k]} e_{T,T}^T x_i + (\lambda_i^{[k]} - \beta_i^{[k]})^T x_i \right) \qquad (4.6) \\ &\quad s.t.\ \begin{bmatrix} I & -B_{T-t_0} \end{bmatrix} \begin{bmatrix} x_i \\ u_i \end{bmatrix} = \mathbb{1} x_{i0}; \end{aligned}$$

The minimization is only over the functions $\tilde{\mathcal{L}}_i$ due to (4.5). Note that the minimization above is an equality constrained quadratic program. We refer to this sub-problem solved by each agent as QP_i. The notation $\overline{x}_i^{[k+1]}$ indicates the optimal solution computed in the kth iteration.

The next step in the algorithm is to propagate the multipliers in the direction of the positive gradient of the Lagrangian function with respect to each multiplier.

Thus,

$$\begin{aligned}\mu^{[k+1]} &= \mu^{[k]} + \alpha_{\mu}^{[k]} \nabla_{\mu} \mathcal{L}(\mathbf{x}, \mathbf{u}, \delta, \zeta, \mu) \\ &= \mu^{[k]} + \alpha_{\mu}^{[k]} E^T \begin{bmatrix} e_{T,T}^T \overline{x}_i^{[k+1]} \\ \vdots \\ e_{T,T}^T \overline{x}_n^{[k+1]} \end{bmatrix} = \mu^{[k]} + \alpha_{\mu}^{[k]} E^T \overline{\mathbf{x}}^{[k+1]}(T), \quad (4.7)\end{aligned}$$

where we slightly abuse notation above. From the transformation in (4.1), the multiplier update can be expressed equivalently for the nodes as

$$E\mu^{[k+1]} = \gamma^{[k+1]} = \gamma^{[k]} + \alpha_{\mu}^{[k]} E E^T \overline{\mathbf{x}}^{[k+1]}(T). \quad (4.8)$$

Observe that EE^T is the graph Laplacian matrix [1], and the multiplier update can be achieved distributedly by exchanging the values $e_{T,T}\overline{x}_i^{[k+1]}$ to neighbours over the network.

The multiplier update for δ and ζ are similar, but because of the non-negativity constraint we use a projected update,

$$\begin{aligned}\delta^{[k+1]} &= \max\left\{\delta^{[k]} + \alpha_{\delta}^{[k]} \nabla_{\delta} \mathcal{L}(\mathbf{x}, \mathbf{u}, \delta, \zeta, \mu), 0\right\} \\ &= \max\left\{\delta^{[k]} + \alpha_{\delta}^{[k]} \left(\overline{E}^T \overline{\mathbf{x}}^{[k+1]} - R\mathbb{1}\right), 0\right\} \quad (4.9) \\ \zeta^{[k+1]} &= \max\left\{\zeta^{[k]} + \alpha_{\zeta}^{[k]} \nabla_{\delta} \mathcal{L}(\mathbf{x}, \mathbf{u}, \delta, \zeta, \mu), 0\right\} \\ &= \max\left\{\zeta^{[k]} + \alpha_{\zeta}^{[k]} \left(-\overline{E}^T \overline{\mathbf{x}}^{[k+1]} - R\mathbb{1}\right), 0\right\} \quad (4.10)\end{aligned}$$

For these updates, multiplication by $\overline{E}$ does not immediately lead to a distributed computation because of the projection operator. However, the update is still only based on relative quantities between each agent and their neighbours, so a distributed implementation can still be possible (by, for example, having each agent keep track of the edge multipliers for all edges it is incident to). After the edge multiplier update, it can be converted to nodes using $\overline{E}$,

$$\lambda^{[k+1]} = \overline{E}\delta^{[k+1]}, \quad \beta^{[k+1]} = \overline{E}\zeta^{[k+1]},$$

to be used in the next step of the algorithm. Observe that $\lambda^{[k+1]}$ and $\beta^{[k+1]}$ do not need to be non-negative.

Note the choice of the step-sizes for the updates, $\alpha_{\mu}^{[k]}, \alpha_{\delta}^{[k]}, \alpha_{\zeta}^{[k]}$ are also important for the convergence of the algorithm. The choice of step-size is beyond the scope of this work, but can be chosen using standard rules [5]. For this work, we assume constant step-sizes.

The entire dual-decomposition sub-gradient algorithm is summarized in Algorithm 1. Note that this algorithm is an *asymptotic* algorithm, and with the correct choice of step-size it converges to the optimal solution of (3.5),

$$\lim_{k\to\infty}(\overline{\mathbf{x}}^{[k]}, \overline{\mathbf{u}}^{[k]}, \gamma^{[k]}, \beta^{[k]}, \lambda^{[k]}) = (\bar{\mathbf{x}}^{(t_0,x_0)}, \bar{\mathbf{u}}^{(t_0,x_0)}, E\bar{\mu}^{(t_0,x_0)}, \overline{E}\bar{\zeta}^{(t_0,x_0)}, \overline{E}\bar{\delta}^{(t_0,x_0)}).$$

Algorithm 1: Dual Sub-Gradient Method

```
Data: Initial conditions : $x_i(t_0) = x_i(0)$, $\mu^{[0]} = \mu_0$, $\zeta^{[0]} = \zeta_0$ and $\delta^{[0]} = \delta_0$,
      Parameters $\alpha_\delta, \alpha_\zeta, \alpha_\mu, R$,given.
begin
    for $k := 0$ to $\infty$ do
        $\gamma^{[k]} = E\mu^{[k]} \in \mathbb{R}^n$
        $\beta^{[k]} = \overline{E}\zeta^{[k]} \in \mathbb{R}^{nT}$
        $\lambda^{[k]} = \overline{E}\delta^{[k]} \in \mathbb{R}^{nT}$
        for $i := 1$ to $n$ do
            Each agent solves its own $QP_i$
            $[\overline{x}_i^{[k+1]}\ \overline{u}_i^{[k+1]}] = \arg\min_{x_i,u_i}(J_i(t_0,T,x_i,u_i) + \gamma_i^{[k]} e_{T,T}^T x_i + (\lambda_i^{[k]} - \beta_i^{[k]})^T x_i)$
                s.t $x_i = \mathbb{1}_T x_i(0) + B_{T-t_0} u_i$
        After solving the $QP_i(t)$ the multipliers will be updated :
        $\mu^{[k+1]} = \mu^{[k]} + \alpha_\mu E(\mathcal{G})^T (I_n \otimes e_{T,T}^T)\overline{\mathbf{x}}^{[k+1]}$
        $\delta^{[k+1]} = \max\left(\delta^{[k]} + \alpha_\delta \left[\overline{E}^T \overline{\mathbf{x}}^{[k+1]} - R\mathbb{1}_{(n-1)T}\right], 0\right)$
        $\zeta^{[k+1]} = \max\left(\zeta^{[k]} + \alpha_\zeta \left[-\overline{E}^T \overline{\mathbf{x}}^{[k+1]} - R\mathbb{1}_{(n-1)T}\right], 0\right)$
```

As seen above, the sub-gradient method has significant advantages due to the fact that it can run distributedly among the agents. Its disadvantage is that it is an *asymptotic* algorithm. If we wish to use the the dual sub-gradient algorithm, we need to run the algorithm infinity iterations to get the optimal trajectory of each agent.

Here we also reinforce the relationship between distrubuted solution methods for problem (3.5) and the connectivity graph that defines the constraints of the problem. The Algorithm 1 requires that agents exchange information with eachother. The connectivity and terminal time constraint are described using an incidence matrix that comes from the communication network between agents. As seen in the derivation of the algorithm, it is precisely this network that the agents need to exchange information on the multiplier updates with eachother.

4.1.1 Numerical Example

This section provides a numerical simulation example demonstrating Algorithm 1. In this example, we consider $n = 8$ agents, a terminal time of $T = 30$ seconds, $\alpha_\mu, \alpha_\zeta, \alpha_\delta = 1.2$ and we run the algorithm for 60 iterations. The radius defined is $R = 35$ with communication graph shown in Figure 3.1. The agent preferences, state and control weights, and initial conditions are given as the previous simulation of the OCPs (terminal and distance and terminal constraints) from Section 3.2.

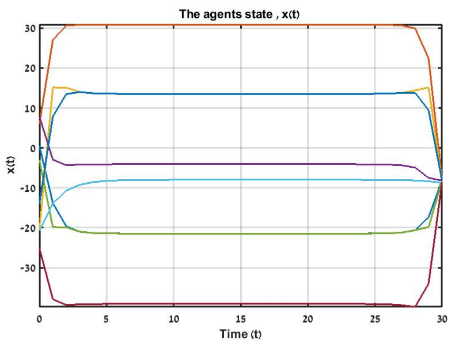
In Figures 4.1a and 4.1c we can see the control and trajectories of the agents. Notice that the agents tries to reach agreement at the surroundings of -8.3407 at the terminal time T while maintaining the distance constraints throughout the entire trajectory. Due to the fact that the sub-gradient algorithm is asymptotic, we didn't get the optimal result, (although the agents tried to reach an agreement) meaning $E^T x(T) \neq \mathbf{0}$ as seen in Figure 4.1b. We can see that there is a distance between the agents at the terminal time. In this simulation, the algorithm was running for 60 iterations to obtain a good solution. As shown in Figure 4.3, as the number of iterations in the algorithm are increased, we approach the optimal solution.

4.2 A Finite-Time Distributed Algorithm

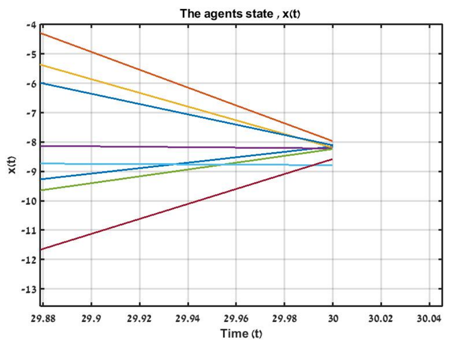
As we explored the dual decomposition sub-gradient algorithm, we saw that it can achieve good results. However, the algorithm must be run "off-line" before the agents are released to do their tasks. If we want good accuracy for the trajectories, we need to run the algorithm for potentially many iterations.

The primary goal of this thesis is to find an on-line finite-time algorithm for solving (3.5). We also want to have the iteration step of the algorithm correspond to real-time. So after each step in the algorithm, the agents should already propagate their physical state forward in a direction they think is optimal. After T steps the algorithm will terminate.

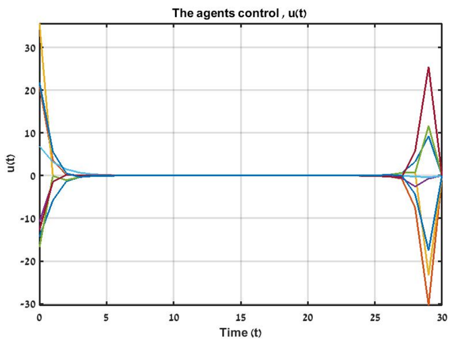
The algorithm we develop here is based on the *shrinking horizon preference agreement* (SHPA) algorithm originally developed in [3]. In this section we present



(a) The Sub-gradient (Algorithm 1) agnets trajectory ; a plot of $\mathbf{x}(t)^{[60]}$.



(b) The Sub-gradient (Algorithm 1) agents trajectory at terminal time; a plot of zooming on $\mathbf{x}(T)^{[60]}$.



(c) The Sub-gradient (Algorithm 1) agents control; a plot of $\mathbf{u}(t)^{[60]}$.

Figure 4.1: The Sub-gradient (Algorithm 1); a plot of $\mathbf{u}(t)^{[60]}$ and $\mathbf{x}(t)^{[60]}$.

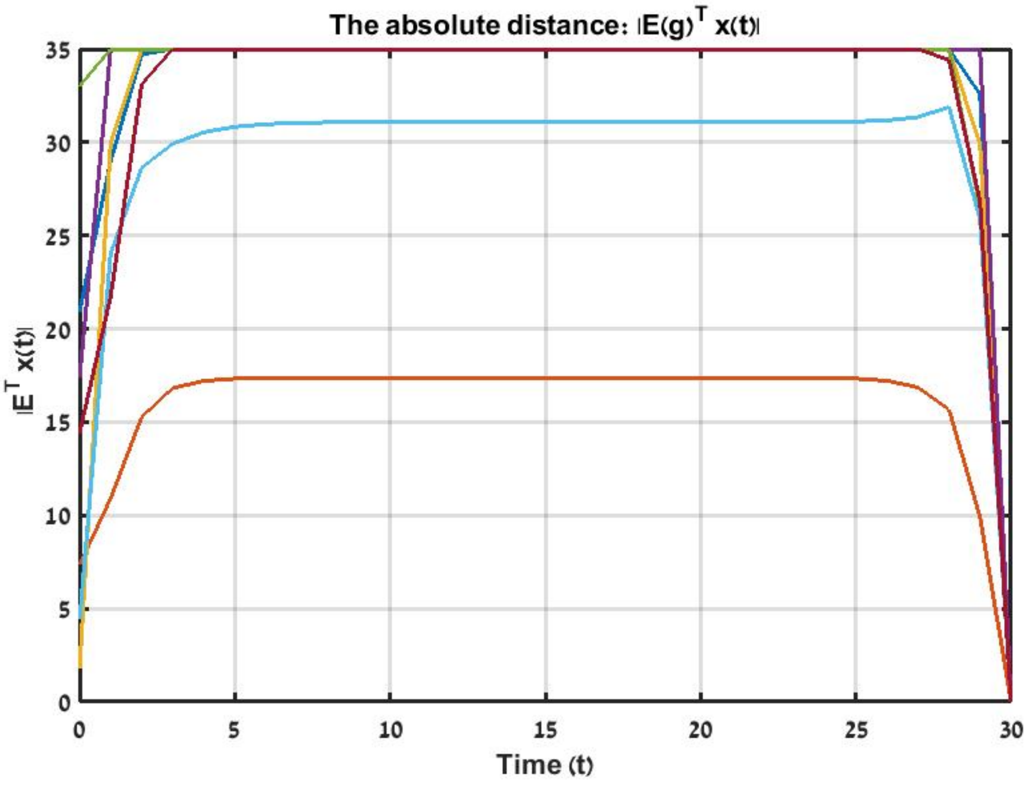


Figure 4.2: The Sub-gradient (Algorithm 1) satisfies the distance constraints; a plot of $|E^T\mathbf{x}(t)^{[60]}|$.

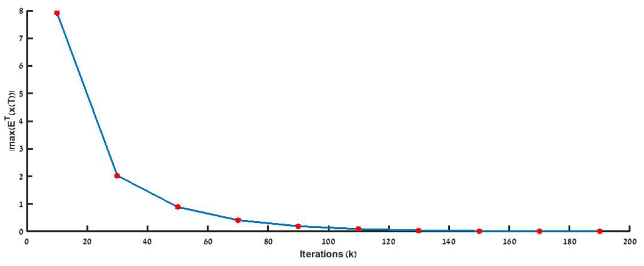


Figure 4.3: The Sub-Gradient (Algorithm 1) absolute maximal distance as a function of the number of iterations; a plot of $\max(|E^T\mathbf{x}(t)^{[K_{limit}]}|)$ at different K_{limit} values.

a modification of the SHPA algorithm to incorporate the distance constraints of (3.5). We solve this problem for the 2-agent case.

4.2.1 The SHPA Algorithm

In this subsection we will introduce the SHPA algorithm as described in [3], with the addition of the distance constraints. The algorithm is stated in Algorithm 2. The main differences between this algorithm and Algorithm 1 is it is a finite time algorithm, and at each step of the algorithm, agent i solves a quadratic program corresponding to an optimal control problem where the initial time changes - thus motivating the term *shrinking horizon*. In this way, each agent has its physical state, $x_i(t)$, and the multiplier values $\mu(t), \delta(t)$, and $\zeta(t)$ at time t. Thus, the QP_i is

$$\begin{aligned}\left[\begin{array}{cc}\overline{x}_i^{[t+1]} & \overline{u}_i^{[t+1]}\end{array}\right] &= \arg\min_{x_i,u_i} J_i(t,T,x_i,u_i) + \gamma_i(t)e_{\tau,\tau}^T x_i + (\lambda_i(t)-\beta_i(t))^T x_i \quad (4.11)\\ s.t.\ \left[\begin{array}{cc} I & -B_\tau\end{array}\right]\left[\begin{array}{c} x_i \\ u_i\end{array}\right] &= \mathbb{1}x_i(t);\end{aligned}$$

we refer to this problem as $QP_i(t)$ to emphasize that the problem parameters explicitly depend on the real-time t. The horizon is defined as τ, where $\tau = T - t$. Note that at each iteration, the horizon effectively “shrinks” (the length of the trajectory) and the initial condition (the physical state $x_i(t)$) corresponds to the propagated state from the previous iteration. The solution of the $QP_i(t)$ is then used to physically propagate that agent forward using the optimal control,

$$x_i(t+1) = x_i(t) + e_{1,\tau}^T \overline{u}_i^{[t+1]},$$

and the multipliers are updated according to the sub-gradient as in the dual - decomposition algorithm.

We now examine the performance of Algorithm 2 compared to the dual-decomposition sub-gradient algorithm and the optimal solution of (3.5).

Theorem 2. *Let $\overline{\mu}, \overline{\delta}, \overline{\zeta}$ denote the optimal Lagrange multipliers associated with* (3.5)*. If Algorithm 2 is initialized with these multipliers, then the trajectories generated by the algorithm are the optimal trajectories of* (3.5)*.*

Proof. According to Lemma 1, once we use the optimal initial conditions we will cause our trajectory to act like the optimal solution. In our case, we are using the optimal multipliers, which will cause the $\mathcal{L}(\mathbf{x},\mathbf{u},\overline{\delta},\overline{\zeta},\overline{\mu})$ to be optimal. Hence,

Algorithm 2: SHPA Algorithm

```
Data: Initial conditions : $x_i(t_0) = x_{i0}, \mu(t_0) = \mu_0, \zeta(t_0) = \zeta_0$ and
      $\delta(t_0) = \delta_0$, Parameters $\alpha_\delta, \alpha_\zeta, \alpha_\mu, R$ given.
begin
    for $t := t_0$ to $T-1$ do
        $\tau = T - t$
        $\gamma(t) = E(\mathcal{G})\mu(t) \in \mathbb{R}^n$
        $\beta(t) = \overline{E}\zeta(t) \in \mathbb{R}^{n\tau}$
        $\lambda(t) = \overline{E}\delta(t) \in \mathbb{R}^{n\tau}$

        for $i := 1$ to $n$ do
            Each agent solves its own $QP_i(t)$
            $[\overline{x}_i \; \overline{u}_i] = \arg\min_{x_i,u_i}(J_i(t,T,x_i,u_i) + \gamma_i(t)e_{\tau,\tau}^T x_i + (\lambda_i(t) - \beta_i(t))^T x_i$
                s.t $x_i(k+1) = x_i(k) + u_i(k),\; k = t, \dots, T-1$
        After solving the $QP_i(t)$ the physical state of each agent is
        propagated as
            $x_i(t+1) = x_i(t) + e_{1,\tau}^T \overline{u}_i,\; i = 1, \dots, n$
        and the multipliers will be updated as
            $\mu(t+1) = \mu(t) + \alpha_\mu E(\mathcal{G})^T (I_n \otimes e_{\tau,\tau})^T \overline{\mathbf{x}}$
            $\delta(t+1) = \max\left(\delta(t) + \alpha_\delta\left[\overline{E}^T\overline{\mathbf{x}} - R\mathbb{1}_{(n-1)\tau}\right], 0\right)$
            $\zeta(t+1) = \max\left(\zeta(t) + \alpha_\zeta\left[-\overline{E}^T\overline{\mathbf{x}} - R\mathbb{1}_{(n-1)\tau}\right], 0\right)$
```

the trajectory will be optimal as well. As mentioned in Lemma 1 : $OCP(t,T,z)$ and $OCP(t+1,T,w)$ with $w = z + \bar{u}^{(t,z)}(t)$ satisfy
$(\bar{x}^{(t,z)}(\tau), \bar{u}^{(t,z)}(\tau), \bar{\mu}^{(t,z)}(\tau), \bar{\zeta}^{(t,z)}(\tau), \bar{\delta}^{(t,z)}(\tau)) =$
$(\bar{x}^{(t+1,w)}(\tau), \bar{u}^{(t+1,w)}(\tau), \bar{\mu}^{(t+1,w)}(\tau), \bar{\zeta}^{(t+1,w)}(\tau), \bar{\delta}^{(t+1,w)}(\tau))\,, \tau = t+1 \dots T.$ $\square$

We now demonstrate this algorithm with a numerical example. In this example, we consider $n = 8$ agents, and a terminal time of $T = 30$ seconds. The connectivity graph used is given in Figure 3.1, and the agent preferences, weights, and initial conditions are the same as those from Section 3.1.1. We chose $R = 35$ for the connectivity radius, and the step sizes are $\alpha_\mu, \alpha_\zeta, \alpha_\delta = 1.2$. Figure

We first demonstrate the results of Theorem 2 by initializing the algorithm with the optimal multiplier values associated with OCP (3.5). Figure 4.4 shows the trajectory generated by the algorithm (solid line) against the optimal trajectory computed using a centralized algorithm (dashed line) as in Section 3.2. Figure 4.5 shows that the distance constraints are not violated during the algorithm.

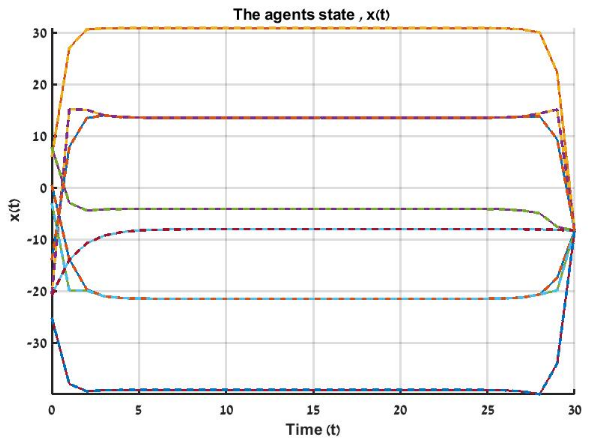


Figure 4.4: The SHPA (Algorithm 2) agents trajectory; a plot of $\mathbf{x}(t)$ when $\delta(0) = \bar{\delta}$,$\zeta(0) = \bar{\zeta}, \mu(0) = \bar{\mu}$.

We now compare this against the algorithm when initialized with 0 values for the multipliers. Figure 4.6 shows the trajectory of each agent generated by the algorithm (solid). The optimal trajectories of OCP (3.5) are shown in dashed lines. Clearly we can see the generated trajectories are not the same. Furthermore, Figure 4.7 shows that the distance constraints are violated by the algorithm.

We emphasize here that the simulation verifying the results of Theorem 2 require the optimal multiplier values from OCP (3.5). This, however, effectively requires the optimal solution to the problem, and it is not assumed that this is available. At the same time, when the algorithm is initialized with non-optimal multiplier values, we can not expect that the constraints of the problem are satisfied. This then motivates the study of initializing the multipliers to ensure that the constraints are not violated. The next result shows that indeed, it is possible to initialize the multipliers to ensure that the first step of the algorithm does not violate the distance constraints.

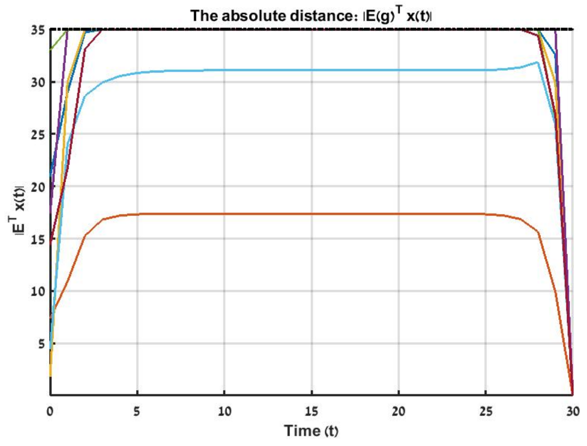


Figure 4.5: The SHPA method (Algorithm 2) satisfies the distance constraints. A plot of $\max(|E(\mathcal{G})^T\mathbf{x}(\mathbf{t})|)$ when the multipliers initialized by the optimal value from the centralized method.

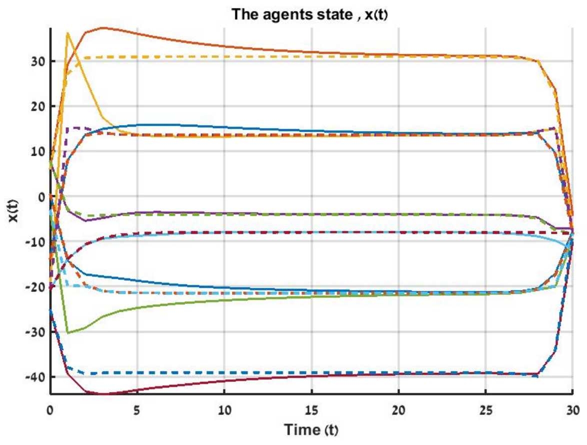


Figure 4.6: Agent trajectories $\mathbf{x}(\mathbf{t})$ generated by Algorithm 2 (solid line) when the multipliers are initialized with 0. The dashed line are the optimal trajectories of OCP (3.5).

Before presenting the result, we first express $QP_i(t)$ shown in (4.11) in standard form,

$$\begin{aligned} \min_{z_i} \quad & \frac{1}{2} z_i^T H_i z_i + f_i^T z_i \\ s.t. \quad & A_{eq} z_i = b_{ieq}. \end{aligned}$$

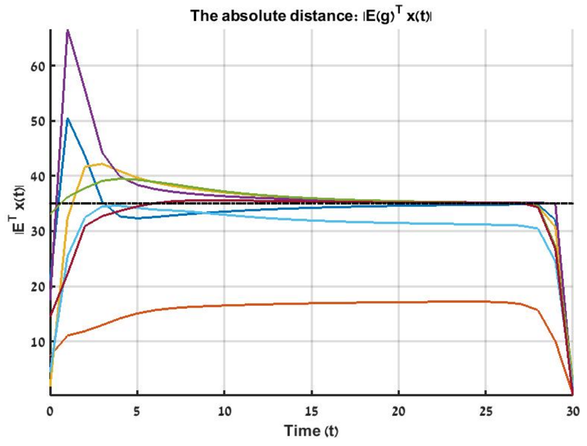


Figure 4.7: The distance constraint for Algorithm 2 when the multipliers are initiated with zeroes.

As before, z_i represents the concatenated state and control variable for the horizon from t to T for agent i, and $\tau = T - t$. The equality constraints represent the agent dynamics, and can be expressed as

$$[I \ \ -B_{T-t_0}]z_i \ = \ \mathbb{1}x_{i0}.$$

The cost function parameters are

$$H_i = \begin{bmatrix} Q_i & \mathbf{0} \\ \mathbf{0} & R_i \end{bmatrix} = \begin{bmatrix} q_i I_\tau & \mathbf{0} \\ \mathbf{0} & r_i I_\tau \end{bmatrix}$$

$$f_i = \underbrace{\begin{bmatrix} -q_i \xi_i \mathbb{1}_\tau + e_{\tau,\tau}\gamma_i \\ \mathbf{0} \end{bmatrix}}_{f_{1,i}} + \underbrace{\begin{bmatrix} \lambda_i - \beta_i \\ \mathbf{0} \end{bmatrix}}_{f_{2,i}}.$$

Theorem 3. *Consider Algorithm 2 for $n = 2$ agents and assume that the initial conditions $x_i(0)$ of each agent ($i = 1, 2$) satisfies $|E^T x(0)| \leq R\mathbb{1}$. Then there exists initial values for the multipliers $\lambda(0), \beta(0)$ and $\gamma(0)$ that guarantees that Algorithm 2 generates trajectories that satisfy $|E^T x(1)| \leq R\mathbb{1}$.*

Proof. In the first iteration of the SHPA algorithm, corresponding to $t = t_0 = 0$, each agent solves the quadratic program as mentioned in (4.11).

After transferring the QP into standard form, we can solve the $QP_i(0)$ analytically, due to the fact that it has equality constraints,

$$\bar{z}_i = -H_i^{-1} f_i + H_i^{-1} A_{eq}^T (A_{eq} H_i^{-1} A_{eq}^T)^{-1} (A_{eq} H_i^{-1} f_i + b_{ieq}).$$

The optimal control used to propagate the physical state at $t = 0$ is $e_{1,\tau}^T \overline{u}_i = e_{T+1,2T}^T \bar{z}_i$, or in expanded form as

$$e_{1,\tau}^T \overline{u}_i = \underbrace{-e_{T+1,2T}^T H_i^{-1} f_i}_{=0} + e_{T+1,2T}^T H_i^{-1} A_{eq}^T (A_{eq} H_i^{-1} A_{eq}^T)^{-1} (A_{eq} H_i^{-1} f_i + b_{ieq}).$$

To simplify the expression, define

$$\begin{aligned} C_i &= e_{T+1,2T}^T H_i^{-1} A_{eq}^T (A_{eq} H_i^{-1} A_{eq}^T)^{-1} (A_{eq} H_i^{-1} f_{1,i} + b_{ieq}) \\ v_i^T &= e_{T+1,2T}^T H_i^{-1} A_{eq}^T (A_{eq} H_i^{-1} A_{eq}^T)^{-1} A_{eq} H_i^{-1}, \end{aligned}$$

leading to the expression

$$e_{1,\tau}^T \overline{u}_i = v_i^T f_{2,i} + C_i.$$

Note that by fixing $\gamma_i(0)$, the term C_i is a known constant that can be computed locally by agent i. It remains to then chose initial conditions for $\lambda_i(0)$ and $\beta_i(0)$ (the terms inside $f_{2,i}$).

The constraint that we wish to satisfy between the agents at $t = 0$ can be expressed as

$$\begin{aligned} -R \leq \big(x_1(0) + e_{1,\tau}^T \overline{u}_1\big) - \big(x_2(0) + e_{1,\tau}^T \overline{u}_2\big) \leq R \\ -R \leq x_1(0) + C_1 + v_1^T f_{2,1} - x_2(0) - v_2^T f_{2,2} - C_2 \leq R. \end{aligned} \tag{4.12}$$

By defining $\tilde{C}_i = x_i(0) + C_i$, we obtain the following inequality,

$$-R + \tilde{C}_2 - \tilde{C}_1 \leq v_1^T f_{2,1} - v_2^T f_{2,2} \quad \leq \quad R + \tilde{C}_2 - \tilde{C}_1 \tag{4.13}$$

Observe also that the multipliers, due to (4.1), must satisfy $\mathbb{1}^T \lambda = \mathbb{1}^T \beta = 0$, leading to

$$\mathbb{1}^T (f_{2,1} + f_{2,2}) = 0. \tag{4.14}$$

The multipliers λ_i and β_i are vectors with values associated to each time in the trajectory horizon. However, as we are concerned with ensuring that the first time step does not violate the distance constraint, we will assume the form $\lambda_i = \overline{\lambda}_i \mathbb{1}$ and $\beta_i = \overline{\beta}_i \mathbb{1}$ for the multipliers. This then allows us to determine a single scalar value $S_i = \overline{\lambda}_i - \overline{\beta}_i$ to satisfy the desired condition rather then τ values. The required inequality then simplifies to

$$-R + \tilde{C}_2 - \tilde{C}_1 \leq \underbrace{v_1^T \begin{bmatrix} \mathbb{1} \\ \mathbf{0} \end{bmatrix}}_{M_1} S_1 - \underbrace{v_2^T \begin{bmatrix} \mathbb{1} \\ \mathbf{0} \end{bmatrix}}_{M_2} S_2 \quad \leq \quad R + \tilde{C}_2 - \tilde{C}_1.$$

Now we can define a set Ω containing all pairs (S_1, S_2) that satisfy the inequalities above,

$$\Omega = \left\{ \begin{array}{cc} (S_1, S_2) \in \mathbb{R}^2 | & S_1 + S_2 = 0, \\ & -R + \tilde{C}_2 - \tilde{C}_1 \leq M_1 S_1 - M_2 S_2 \leq R + \tilde{C}_2 - \tilde{C}_1 \end{array} \right\} \tag{4.15}$$

Therefore, for any initial condition $\gamma_i(0)$ with $f_{2,i}$ of the form

$$\begin{bmatrix} S_i \mathbb{1} \\ \mathbf{0} \end{bmatrix}$$

with $(S_1, S_2) \in \Omega$ ensures that the distance constraint at $t = 1$ is not violated. Any λ_i and β_i satisfying $\lambda_i - \beta_i = S_i \mathbb{1}$ can be used as the initial condition. □

Theorem 3 characterizes the set of multipliers that ensure the constraint is satisfied. For purposes of the algorithm, we must also propose a method for choosing a particular value to use. Note that the set Ω defines a line segment in $\mathbb{R}^2$, visualized in Figure 4.8. It is possible, therefore, to simply calculate the endpoints of the line segment to use as the initial condition for the multipliers. We denote these points as (S_1^{High}, S_2^{Low}) and (S_1^{Low}, S_2^{High}) (also labeled in Figure 4.8). These points can be found by solving the linear equations

$$\begin{aligned} \begin{bmatrix} M_1 & -M_2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} &= \begin{bmatrix} R + \tilde{C}_1 - \tilde{C}_2 \\ 0 \end{bmatrix} \\ \begin{bmatrix} M_1 & -M_2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} &= \begin{bmatrix} -R + \tilde{C}_1 - \tilde{C}_2 \\ 0 \end{bmatrix}, \end{aligned}$$

and we are free to chose either pair of points.

It now remains to chose λ_i and β_i. We also chose a heuristic as follows,

$$\beta_1 = \begin{cases} -S_1, & S_1 > 0 \\ S_2, & S_1 < 0 \\ 0, & \text{o.w} \end{cases} \tag{4.16}$$

$$\beta_2 = -\beta_1. \tag{4.17}$$

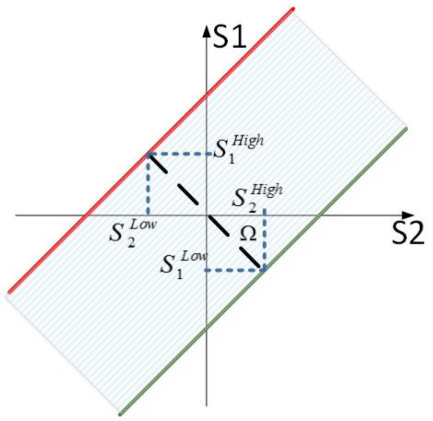


Figure 4.8: The S_1 and S_2 domain, the diagonal green and red lines are the inequalities, the dashed black line represent the set Ω.

$$\lambda_1^{[0]} = \begin{cases} S_1, & S_1 < 0 \\ -S_2, & S_1 > 0 \\ 0, & \text{o.w} \end{cases} \tag{4.18}$$

$$\lambda_2 = -\lambda_1. \tag{4.19}$$

Equations (4.16) , (4.17), (4.18) and (4.19) ensure that the edges multipliers will be non-negative (due to the inequality constraints). Once we found the multipliers value, we can use the left-inverse of the incidence matrix to transfer the multipliers from nodes to edges.

$$\delta = (\bar{E}^T\bar{E})^{-1}\bar{E}^T(\lambda \otimes \mathbb{1}) \tag{4.20}$$

$$\zeta = (\bar{E}^T\bar{E})^{-1}\bar{E}^T(\beta) \otimes \mathbb{1}). \tag{4.21}$$

To demonstrate the results of Theorem 3, we run a numerical example with two agents. The parameters of the agents are given as

$$r = \begin{bmatrix} 9 \\ 10 \end{bmatrix}, \quad q = \begin{bmatrix} 5 \\ 5 \end{bmatrix}, \quad \xi = \begin{bmatrix} 25 \\ -59 \end{bmatrix}, \; x_0 = \begin{bmatrix} -5.5628 \\ -14.5777 \end{bmatrix}.$$

The radius is given as $R = 35$, $T = 50$, and the multipliers step values as $\alpha_\mu, \alpha_\zeta, \alpha_\delta = 1$. The optimal trajectories of OCP (3.5) were found in MATLAB using the quadratic programming formulation. Figures 4.9a and 4.9b shows the resulting trajectories generated by Algorithm 2 using the result of Theorem 3 (solid line) compared with the optimal trajectories of OCP (3.5) (dashed lines). Observe that the output generated by the SHPA algorithm does not satisfy the terminal time constraint,

$$x(T) = \begin{bmatrix} -16.74 \\ -17.72 \end{bmatrix},$$

which is not optimal since the agents are not in agreement.

Based on Theorem 3, we chose the values $S_1(0) = 59.1904,\ S_2(0) = -59.1904$, which translate to $\delta(0) = 59.1904\mathbb{1}$ and $\zeta(0) = \mathbf{0}$.The set Ω is shown in Figure 4.10.

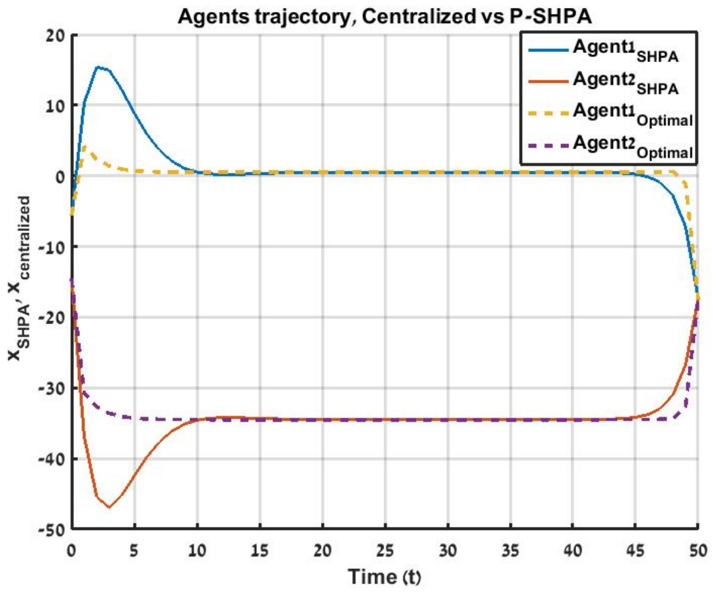
Figure 4.11 shows the distance between the two agents during the trajectory generated by the algorithm. Notice that in the first time step, the distance constraint is indeed preserved, validating the results of Theorem 3. However, we also observe that in the next step, this constraint still becomes violated. This then motivates the modification to the algorithm. At each time-step we can perform a similar projection of the multipliers to ensure the distance constraint is not violated - we term this new algorithm the *projected SHPA* algorithm. This result is presented in the next section.

4.2.2 The Projected SHPA Algorithm

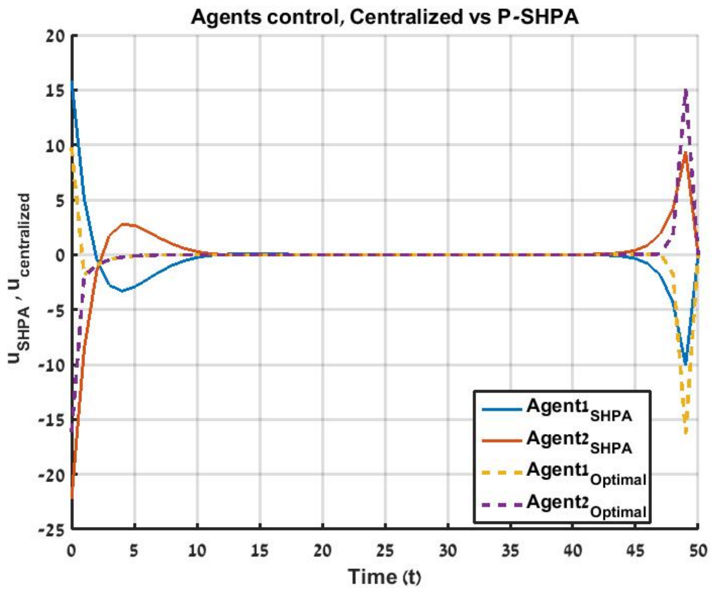
The SHPA algorithm in Algorithm 2 is based on the dual-decomposition sub-gradient algorithm. The update of the multipliers corresponding to the distance constraints had to be projected onto the non-negative orthant. In Theorem 3, we saw that in the first step, we could determine a good set of multipliers by picking values from a special set. The main idea of the Projected SHPA algorithm (P-SHPA) is in the multiplier update phase, we project the multipliers difference onto this special set at each iteration. The P-SHPA algorithm is given in Algorithm 3.

The algorithm works as follows. At each step of the algorithm, the set Ω, as described in Theorem 3, should be calculated. Note that now we explicitly express this set as a function of time, since the parameters will change at each step of the algorithm. We then project the current value of the multipliers $\lambda(t)$ and $\beta(t)$ onto this set and use this value to solve each agent's $QP_i(t)$ - the solution can be obtained analytically using (2.7). The algorithm then transforms these multipliers back to the edges, and performs the normal sub-gradient propagation procedure.

We now present a result showing that this algorithm indeed ensures that the distance constraints are not violated.



(a) Plot of $\mathbf{x(t)}$.



(b) Plot of $\mathbf{u(t)}$.

Figure 4.9: Demonstration of Theorem 3.

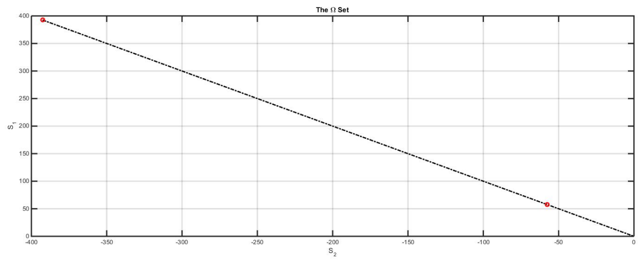


Figure 4.10: A plot of the set Ω from Theorem 3. The dashed line is the set $S_1 + S_2 = 0$ and the red circles are the endpoints of the line segment in Ω.

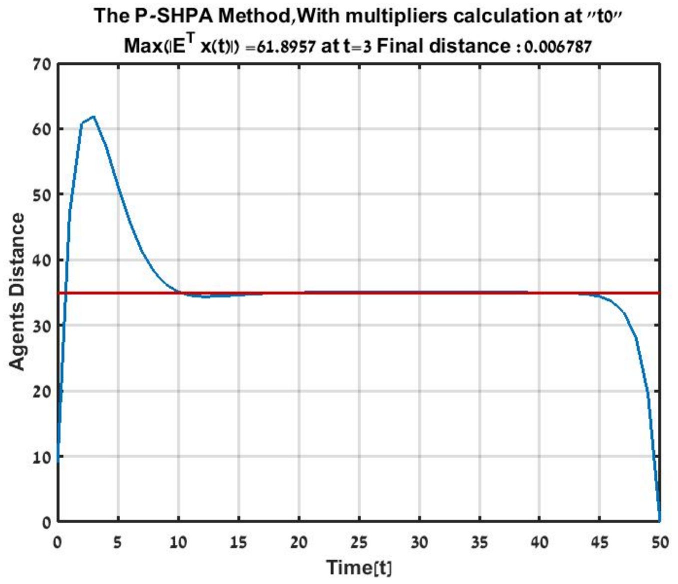


Figure 4.11: The distance constraint is satisfied in the first time step, but violated afterwards.

Theorem 4. *Assume that the initial conditions of the agents satisfy $|E^T x(t_0)| \leq R$. Then $|E^T x(t)| \leq R$, $\forall t = t_0 \ldots T-1$ when x(t) are the trajectories generated by the P-SHPA algorithm for any initiated values of the multipliers.*

Proof. After running and demonstrating Theorem 3, we saw that it is possible to use the multipliers value to influence and control the distance between the agents distance. At Theorem 3 we proved it for $t = t_0$, but we can run the same procedure each time step to calculate the control $u_i(t)$ so the distance at $t = t + 1$ will not diverge from the given radius.

Algorithm 3: Projected SHPA Algorithm

```
Data: Initial conditions : $x_i(t_0) = x_i(0)$, $\mu(t_0) = \mu_0$, $t_0 = 0$, $\zeta(t_0) = \zeta_0$ and
      $\delta(t_0) = \delta_0$, Parameters $\alpha_\delta, \alpha_\zeta, \alpha_\mu, R, T$ given.
begin
    for $t := t_0$ to $T$ do
        $\tau = T - t$
        $\gamma(t) = E(\mathcal{G})\mu(t) \in \mathbb{R}^n,\ \beta(t) = \overline{E}\zeta(t) \in \mathbb{R}^{n\tau},\ \lambda(t) = \overline{E}\delta(t) \in \mathbb{R}^{n\tau}$
        for $i = 1, 2$
        $\begin{cases} \tilde{C}_i(t) &= e^T_{\tau+1,2\tau} H_i^{-1} A_{eq}^T (A_{eq} H_i^{-1} A_{eq}^T)^{-1} (A_{eq} H_i^{-1} f_{1,i} + b_{ieq}) + x_i(t) \\ v_i^T(t) &= e^T_{\tau+1,2\tau} H_i^{-1} A_{eq}^T (A_{eq} H_i^{-1} A_{eq}^T)^{-1} A_{eq} H_i^{-1} \\ M_i(t) &= v_i^T(t) \begin{bmatrix} \mathbb{1}_\tau \\ \mathbf{0} \end{bmatrix} \end{cases}$
        $\Omega(t) = \{(S_1 \mathbb{1}_\tau, S_2 \mathbb{1}_\tau) \in \mathbb{R}^{2\tau} | S_1, S_2 \in \mathbb{R}, S_1 + S_2 = 0,$
                   $-R + \tilde{C}_2 - \tilde{C}_1 \leq M_1(t)S_1 - M_2(t)S_2 \leq R + \tilde{C}_2 - \tilde{C}_1\}$
        Project $\lambda(t) - \beta(t)$ onto $\Omega : P_\Omega(\lambda(t) - \beta(t)) = (\overline{S}_1 \mathbb{1}_\tau, \overline{S}_2 \mathbb{1}_\tau)$
        Each agent solves its own $QP_i(t)$,
        $[\overline{x}_i \quad \overline{u}_i] = \arg\min_{x_i, u_i} \ (J_i(t, T, x_i, u_i) + \gamma_i(t) e^T_{\tau,\tau} x_i + \overline{S}_i \mathbb{1}_\tau^T x_i$
                       s.t $\quad x_i(t+1) = x_i(t) + e^T_{1,\tau} \overline{u}_i$
        If $\lambda(t) - \beta(t) \neq P_\Omega(\lambda(t) - \beta(t))$, determine $\lambda_i(t), \beta_i(t)$ using
        heuristic in (4.17), (4.19) and calculate the edge multipliers
        $\zeta(t) = (\overline{E}^T \overline{E})^{-1} \overline{E}^T \beta(t)$.
        $\delta(t) = (\overline{E}^T \overline{E})^{-1} \overline{E}^T \lambda(t)$
        Propagate the multipliers:
            $\mu(t+1) = \mu(t) + \alpha_\mu E(\mathcal{G})^T x(T)$
            $\delta(t+1) = \max\left(\delta(t) + \alpha_\delta \left[\overline{E}^T \overline{x} - R\mathbb{1}_{(n-1)\tau}\right], 0\right)$
            $\zeta(t+1) = \max\left(\zeta(t) + \alpha_\zeta \left[-\overline{E}^T \overline{x} - R\mathbb{1}_{(n-1)\tau}\right], 0\right)$
```

□

This means that the *P-SHPA* algorithm will ensure that the distance constraints are not violated. We now examine the performance of Algorithm 3 compared to centralized solution, which is the optimal solution of (3.5). In this example we consider the following parameters,

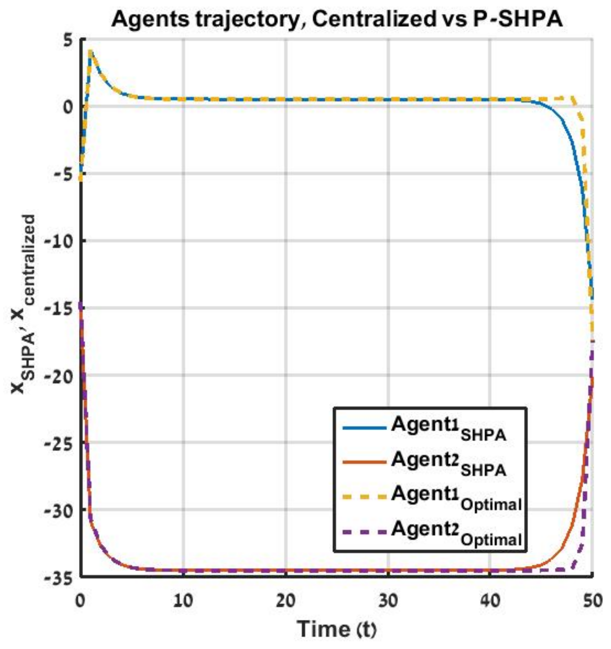
$$r = \begin{bmatrix} 9 \\ 10 \end{bmatrix}, \quad q = \begin{bmatrix} 5 \\ 5 \end{bmatrix}, \quad \xi = \begin{bmatrix} 25 \\ -59 \end{bmatrix}, \; x_0 = \begin{bmatrix} -5.5628 \\ -14.5777 \end{bmatrix}.$$

The radius is given as $R = 35$, the final time $T = 50$, and the multipliers step values , $\alpha_\mu, \alpha_\zeta, \alpha_\delta = 1$. As shown in Figures 4.12a , 4.12b and 4.13, the trajectories generated by Algorithm 3 do not violate the distance constraint. Furthermore, for these parameters, the trajectories do a good job approximating the optimal solution. Figure 4.14 plots the distance constraint multipliers and the terminal time multiplier.

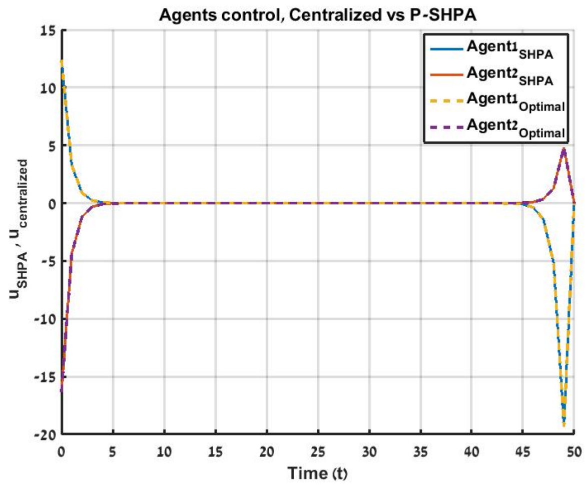
We now show a simulation example for two agents with preferences that are less than the distance radius apart, that is $|\xi_1 - \xi_2| < R$. The parameters are given as

$$r = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \quad q = \begin{bmatrix} 2 \\ 8 \end{bmatrix}, \quad \xi = \begin{bmatrix} 29 \\ -4 \end{bmatrix}, \; x_0 = \begin{bmatrix} 12.126 \\ 18.2983 \end{bmatrix}.$$

The radius is defined as $R = 35$, terminal time as $T = 50$, and the multipliers step values as $\alpha_\mu, \alpha_\zeta, \alpha_\delta = 1$. We can see in Figures 4.15a , 4.15b and 4.16, that for this case the algorithm performs very well and does not lead to undesirable trajectories that, for example, might cause the agents to deviate from their preference. As expected, this algorithm is still sub-optimal and we can not expect that the terminal time constraint will be satisfied.



(a) Plot of $\mathbf{x}(\mathbf{t})$.



(b) Plot of $\mathbf{u}(\mathbf{t})$.

Figure 4.12: Trajectories generated by the P-SHPA algorithm (solid) and the optimal trajectories of OCP (3.5).

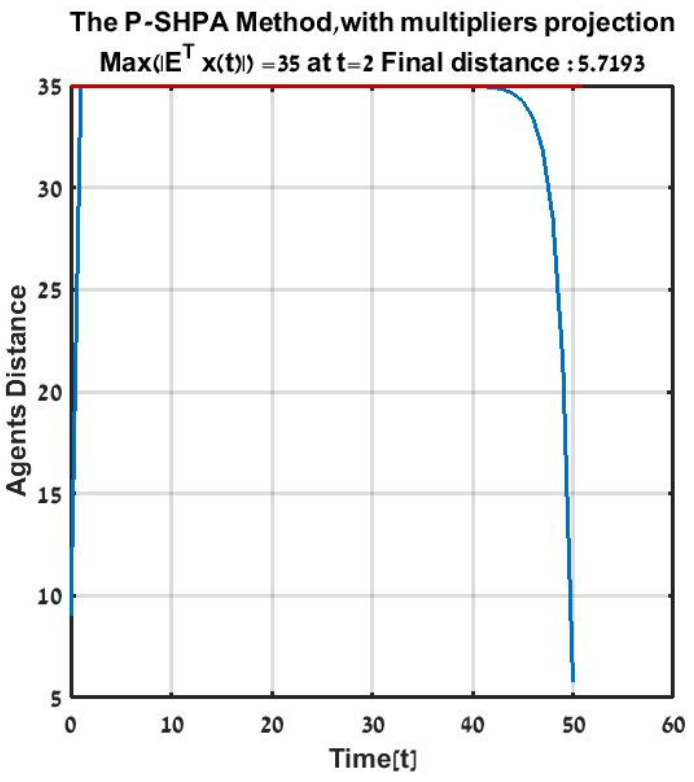
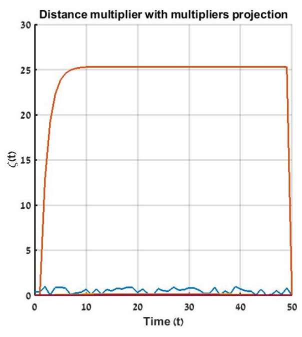
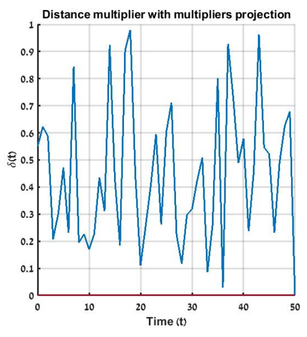


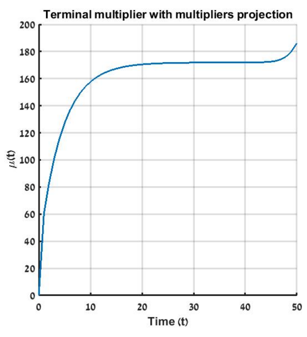
Figure 4.13: The distance constraint is satisfied for the entire horizon using the P-SHPA algorithm.



(a) The multiplier ξ.

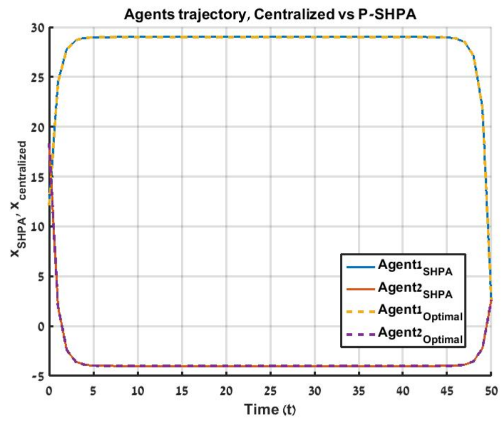


(b) The multiplier δ.

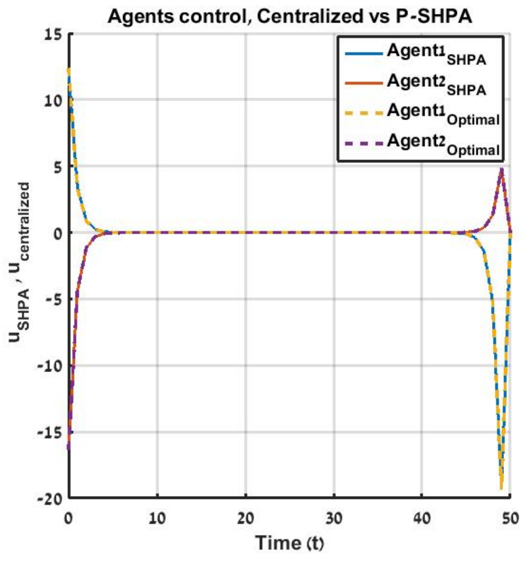


(c) The multiplier μ.

Figure 4.14: A plot of the multipliers generated by the P-SHPA algorithm.



(a) Plot of $\mathbf{x}(\mathbf{t})$.



(b) Plot of $\mathbf{u}(\mathbf{t})$.

Figure 4.15: Trajectories generated by the P-SHPA algorithm (solid) and the optimal trajectories of OCP (3.5) when $|E^T\xi| \leq R\mathbb{1}$.

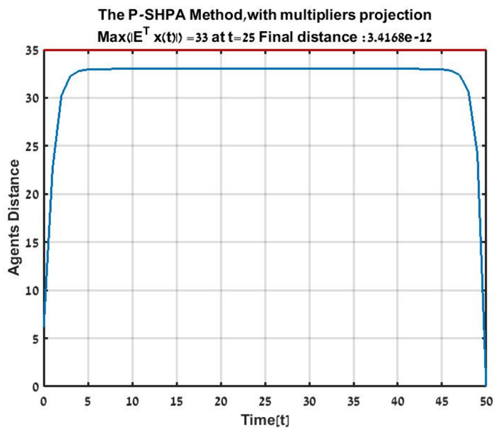


Figure 4.16: The distance constraint is satisfied for the entire horizon using the P-SHPA algorithm when $|E^T\xi| \leq R\mathbb{1}$.

Chapter 5

Conclusions

There is no doubt that the field of multi-agent system will be studied in more depth in the coming years. This research can provide a steadier environment for the study; due to the radius keeping constraint we can know with certainty that no agent will disappear from the connectivity graph. Additionally, we know that no agent will lose its communication with the adjacent agents. This important property means that the connectivity graph staying the same throughout the trajectory is primarily an assumption, but this research can cause it to happen. From the manager of the multi-agent system point of view, it has significance. No lost agents, no graph damage, means a stable multi-agents network. For example, if we use a multi-agent system, with the connection topology of a star, every agent is connected to the central agent. There may be a possibility that the central agent will lose its connectivity to the rest of the group. This will have an immediate impact on the rest of the agents. Most of the agents will probably not know what to do once they become lost. In this specific case, if there is no reconnecting algorithm to join the lost agents, they are lost. The primary achievement of this research is that it will not harm the connectivity graph of the multi-agent system. The secondary achievement is that it is all done on-line, distributed, at finite time, meaning that every agent can work as an individual and still be connected to the rest of the group.

Future work

This research was done and demonstrated by simulation on a two agents case; it can be further investigated into a full swarm of n agents. When dealing with an agent's trajectory, there are many parameters to test. One thing we can test is the function of obstacle avoidance, meaning that each agent in its own trajectory needs to have the ability to overcome obstacles it may encounter when pursuing its objective while maintaining the defined distance and terminal constraints. This is only one example, there are many more. When dealing with the agent's dynamics, we can try various types of dynamics. This work addresses linear dynamics: simple integrator which can be expanded into non-linear dynamics or perhaps mixed dynamics. The field of the multi-agent systems can be widely explored with respect to every field. This specific work can be explored through the field of dealing with n agents, meaning calculating all the multipliers boundaries (as done for two agent case) and projecting the multipliers to the correct set so that there will be no diversion in the distance between the connected agents.

Bibliography

[1] M. Mesbahi and M. Egerstedt *Graph theoretic methods in multiagent networks*, in Princeton University Press, 2010.

[2] L. R. Foulds *Graph theory with applications*, in Springer Science and Business Media, 2012.

[3] D. Zelazo, M. Bürger and F. Allgöwer, *A Finite-Time Dual Method for Negotiation between Dynamical Systems*, in SIAM Journal on Control and Optimization 51.1 (2013): 172-194.

[4] B. Liu, T. Chu, L. Wang and G. Xie, *Controllability of a leader–follower dynamic network with switching topology*. IEEE Transactions on Automatic Control, 53.4 (2008): 1009-1013.

[5] A. Ruszczynski , *Nonlinear Optimization*. Princeton University Press, Princeton, NJ, 2006.

[6] R. Olfati-Saber, *Flocking for multi-agent dynamic systems: Algorithms and theory*. IEEE Transactions on Automatic Control, 51.3 (2006): 401-420.

[7] D. P. Spanos and R. M. Murray, *Motion planning with wireless network constraints*, in Proceedings of the American Control Conference. Vol. 1. 2005.

[8] S. P. Bhat and D. S. Bernstein, *Finite-time stability of continuous autonomous systems*. SIAM Journal on Control and Optimization 38.3 (2000): 751-766.

[9] T. Li, M. Fu, L. Xie and J. F. Zhang, *Distributed consensus with limited communication data rate*. IEEE Transactions on Automatic Control, 56.2 (2011): 279-292.

[10] M. Franceschelli, M. Egerstedt,A. Giua and C. Mahulea, *Constrained invariant motions for networked multi-agent systems*. American Control Conference, 2009. ACC'09. IEEE, 2009.

[11] R. Olfati-Saber,J. A. Fax and R. M. Murray, *Consensus and cooperation in networked multi-agent systems*,in Proceedings of the IEEE 95.1 (2007): 215-233.

[12] W. Wang and M. Zhao, *Joint effects of radio channels and node mobility on link dynamics in wireless networks* , in INFOCOM 2008. The 27th Conference on Computer Communications. IEEE (pp. 933-941).

[13] C. D. Godsil and G. Royle , *Algebraic graph theory*, Springer , New York, 2001.

[14] R. A. Horn and C. R. Johnson, *Topics in matrix analysis*. Cambridge university press, 1991.

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

[16] R. Olfati-Saber, J. A. Fax, and R. M. Murray, *Consensus and Cooperation in Networked Multi-Agent Systems*. Proceedings of the IEEE, 2007 volume 95.

[17] W. Ren, R. W. Beard, and E. M. Atkins, *A survey of consensus problems in multi-agent coordination*. -,Proceedings of the 2005, American Control Conference.

[18] W. W. Chai, *Agreement and consensus problems in groups of autonomous agents with linear dynamics*. -,2005 IEEE International Symposium on Circuits and Systems.

[19] L. Wang and F. Xiao, *Finite-Time Consensus Problems for Networks of Dynamic Agents*. -, 2008.

[20] R. Saber and R. M. Murray *Consensus Protocols for Networks of Dynamic Agents*. Control and Dynamical Systems California Institute of Technology Pasadena, 2003.

[21] D. P Bertsekas and J. N. Tsitsiklis *Parallel and distributed computation*. Prentice Hall Inc, 1989.

[22] J. N. Tsitsiklis *Problems in Decentralized Decision Making and Computation*. Laboratory for Information and Decision Systems, MIT, 1984.

[23] T. D. Tran and A. Y. Kibangou *Distributed Design of Finite-time Average Consensus Protocols*. Gipsa-Lab, CNRS, University Joseph Fourier, 2013.

[24] Y. Zhu , X. Guan1 and X. Luo *Finite-time Consensus for Multi-agent Systems via Nonlinear Control Protocols*. Journal of Automation and Computing, 2013.

[25] S. E. Parsegov , A. E. Polyakov and P. S. Shcherbakov *Finite-time Consensus for Multi-agent Systems via Nonlinear Control Protocols*. 4th IFAC Workshop on Distributed Estimation and Control in Networked Systems , 2013.

[26] S. Kar and J. M. F. Moura *Consensus Algorithms in Sensor Networks With Imperfect Communication: Link Failures and Channel Noise*.IEEE Transactions on signal processing, VOL. 57, NO. 1, 2009

[27] Z. Xue, , Z. Liu , C. Feng and J. Zeng *Target Tracking for Multi-agent Systems with Leader-Following Strategy*. Guangzhou, P. R. China, 29-31,July 2010, pp. 240-243

[28] G. Shi, Y. Hong, and K. H. Johansson *Connectivity and Set Tracking of Multi-Agent Systems Guided by Multiple Moving Leaders*.IEEE Transactions on autimatic control , Vol. 57, no. 3, March 2012

[29] G. Yang, Q. Yang, V. Kapila, D. Palmer and R. Vaidyanathan *Fuel optimal manoeuvres for multiple spacecraft formation reconfiguration using multi-agent optimization*.DOI: 10.1002/rnc.684 , 12 FEB 2002

[30] Y. Kim, M. Mesbahi, F. Y. Hadaegh *Multiple-Spacecraft Reconfiguration Through Collision Avoidance, Bouncing, and Stalemate*. Journal of Optimization Theory and Applications, August 2004, Volume 122, Issue 2, pp 323-343

[31] A. Nedic, and A. Ozdaglar, *Distributed Subgradient Methods for Multi-Agent Optimization*.IEEE Transactions on Automatic Control 2009, Volume 54.

[32] B. Johansson, M. Rabi, and M. Johansson, *A Randomized Incremental Subgradient Method for Distributed Optimization in Networked Systems*.SIAM Journal on Optimization 2009, Volume 20.

[33] M. Zhu, and S. Martinez, *On distributed convex optimization under inequality and equality constraints*. IEEE Transactions on Automatic Control 2012, Volume 57.

[34] A. Nedic and A. Ozdaglar and P. A. Parrilo, *Constrained Consensus and Optimization in Multi-Agent Networks*. IEEE Transactions on Automatic Control 2010, Volume 55.

[35] T. H. Chang and A. Nedic and A. Scaglione, *Distributed Constrained Optimization by Consensus-Based Primal-Dual Perturbation Method*. IEEE Transactions on Automatic Control 2014, Volume 59.

[36] G. Notarstefano and F. Bullo, *Distributed Abstract Optimization via Constraints Consensus: Theory and Applications*. IEEE Transactions on Automatic Control 2011, Volume 56.

[37] B. Johansson, A. Speranzon, M. Johansson and K. H. Johansson, *On decentralized negotiation of optimal consensus*. Automatica, 2008, Volume 44.

[38] A. Rantzer, *Using Game Theory for Distributed Control Engineering*. Department of Automatic Control, Lund University, Sweden, 2008.

[39] M. Bürger, G. S. Schmidt and F. Allgöwer, *Preference Based Group Agreement in Cooperative Control*. Proc. of the 8th IFAC Symposium on Nonlinear Control Systems, 2010.

[40] M. Bürger, D, Zelazo and F. Allgöwer, *A Distributed Real-Time Algorithm for Preference-Based Agreement*. Proc. 18th IFAC World Congress, 2011.

[41] W. Wang and M. Zhao, *Joint Effects of Radio Channels and Node Mobility on Link Dynamics in Wireless Networks*.27th IEEE Conference on Computer Communications, 2008.

[42] S. H. Willi and H. Yorick, *Matrix Calculus and Kronecker Product* .World Scientific publishing, 2011.

[43] M. Bürger, D, Zelazo and F. Allgöwer, *Dynamic Negotiation Under Switching Communication*. Mathematical System Theory – Festschrift in Honor of Uwe Helmke on the Occasion of his Sixtieth Birthday, 2013.

תקציר

נחילים ומערכות ריבוי-סוכנים נהיו רעיון אטרקטיבי של מחקר בשנים האחרונות. ישנם אתגרים שונים כשעובדים עם נחילים : שליטה על כל סוכן בנפרד ,שליטה על קבוצת סוכנית בעת ובעונה אחת , הימנעות ממכשולים ופגיעות בתוך הנחילים וכו. קיימות מספר טופולוגיות לעודה עם נחילים. אחת הטופולוגיות היא בעזרת יחידת שליטה מרכזית. כאשר הנחיל עובד עם יחידת בקרה מרכזית, כלומר כל הסוכנים נשלטים על ידי אותה יחידה, אין לסוכנים יכולת השפעה על הנעשה, הם פשוט מקבלים את הפקודות לביצוע מיחידת הבקרה המרכזית. לשיטה זו ייתרנות רבים אך יש לה לא מעט חסרונות. המצב היום שמנסים להעניק לסוכנים יכולת אוטונומית. כאשר מעניקים לסוכן יכולת כזו, יש בעיה להסתמך ולהתחבר ליחידת בקרה מרכזית. לכן הנטייה היא לתת יותר כח לסוכן בשטח כך שכל סוכן יוכל לפעול באופן עצמאי אך עדיין בשיטוף מסוים עם מספר סוכנים אחרים באותו הנחיל. לדוגמא : נחיל של סוכנים אשר אמור לטפל באזור מוכה אסון , כמו רעידת אדמה. יהיו מספר סוכנים אשר יפוזרו על פני שטח , חלקם יהיו אוויריים, יבשתיים ואולי גם תת קרקעיים. לאחר שיסרקו את השטח לניצולים או הערכת נזק ראשוני. לאחר הסריקה יאלצו להיפגש לבצע הורדה של המידע לצורך הערכת מצב הנפגעים בשטח.

בהמשך לדוגמא זו, מכיוון שאנו יודעים כי כל ערוץ תקשורת בעל רדיוס שידור יעיל, נשתמש באפיון זה להגדיר את המרחק המקסימאלי בין הסוכנים שלנו. מחקר זה יציע אלגוריתם לפתרון בזמן אמת עבור בעיית הסכם במערכות מרובות סוכנים. תנועת הסוכן מתאורת ע"י דינמיקה של סוכם פשוט ולכל סוכן יש את פונקציית מטרה שלו אותה הוא מבקש למזער. כלומר, לכל סוכן יש העדפה אשר אליה הוא חפץ להגיע ומטלה שעליו לבצע ותוך כדי כך על כל הסוכנים להגיע להסכם לגבי נקודת המפגש מבלי להתייחס למצב הסוכנים האחרים. הסוכנים חייבים לתאם ולהגיע להסכם על נקודת המפגש בזמן סופי. כל סוכן מסוגל לתקשר עם הסוכנים האחרים (שכנים בלבד) בהתאם למבנה הקישוריות. הטיפול באילוץ במחקר זה הוא להבטיח כי כל סוכן ישמור על יכולת תקשורת בתוך טווח התקשורת המוגדר עם שכניו. כלומר, לא יהיה נזק למבנה הקישוריות. וזאת כי האלגוריתם בא למנוע מרחק גדול בין הסוכנים מרדיוס התקשורת המוגדר ובכך לצמצם את מקרי "איבוד" קשר עם סוכן או אפילו נפילות תקשורת רגעיות שיכולות לסכן קבוצת סוכנים גדולה. בניגוד למחקרים נוספים שבוצעו בעבר, אשר לקחו בחשבון נקודות מפגש ממוצעות או ממוצע גיאומטרי, האלגוריתם המוצג במחקר זה מבצע תהליך החלטה דינאמי בהתאם לנתוני הסוכנים בשטח תוך אילוצי זמן סופי. כלומר, כל סוכן בעל נתונים אישיים (עדיפות, יעד , גודל בקר, וכו) אשר ממודלים לתוך פונקציית ה"עלות" של כל אחד מהסוכנים. כך שכל סוכן דואג למטרה האישית שלו אך בו בזמן עליו לדאוג גם למטרה המשותפת שהיא הסכמה על נקודת מפגש באופן דינאמי, תוך כדי שמירה על אילוצי המרחק. המטרה של כל אחד מהסוכנים היא למזער את פונקציית ה"עלות" האישית שלו ובכך למזער את פונקציית ה"עלות" הכוללת. במחקר זה נעשו מספר הנחות : גרף הקישוריות הינו מסוג Spanning Tree , והוא קבוע מתחילת הפעילות המשותפת של הסוכנים ועד סופה. הנחה נוספת היא שהמרחק ההתחלתי בין הסוכנים יהיה קטן מרדיוס התקשורת שהוגדר.

התרומה העיקרית של עבודה זו הצגת אלגוריתם מבוזר לבעית ההחלטה תוך כדי עמידה באלוצי המרחק ונקודת המפגש הסופית. עבודה זו תציע אלגוריתם מבוזר תוך כדי מתן דוגמאות , פיתוח החלקים המתמטיים והצגת סימולציות למצב של 2 סוכנים.

המחקר בוצע בהנחייתו של פרופסור דניאל זלזו מהפקולטה לאווירודינמיקה וחלל.

תהליך החלטה בין פלאטפורומות דינאמיות בזמן סופי עם אילוצי תקשורת

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת תואר מגיסטר למדעים
במערכות אוטונומיות ורובוטיקה

יניב בן שושן

הוגש לסנט הטכניון - מכון טכנולוגי לישראל

תאריך עברי : אב , חיפה , אוגוסט 2016

תהליך החלטה בין פלאטפורומות דינאמיות בזמן סופי עם אילוצי תקשורת

יניב בן שושן