Distributed Decoupling Of Multiagent Simple Temporal Problems

Jayanth Krishna Mogali

CMU-RI-TR-16-28

Submitted in partial fulfilment of the requirements for the degree of
Master of Science in Robotics

June 2016

Thesis Committee
Stephen Smith , Co-Chair
Zachary Rubinstein , Co-Chair
Soummya Kar
Matt Wytock

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

Simple temporal networks (STNs) have become a popular tool for modelling the temporal space for a set of tasks during their execution by an agent. Formally, the problem concerning STNs can be viewed as a TCSP (Temporal Constraint Satisfaction Problem) where an agent has a set of tasks to complete, subject to binary and unary constraints. The binary constraints couple pairs of time points (time points can represent start or end time for a task) in an affine manner while unary constraints associate with each time point an upper and lower bound for admissible values (representative of deadlines). STNs model the TCSP as a graph with the time points being the vertices and edges representing the constraint values. In real world settings, the value for each time point is either determined or merely observed by an agent. The effect of assignment of a value to a time point needs to be propagated through the graph for checking feasibility and updating the domains for the unassigned time points while execution. STN's owe their popularity to efficient incremental algorithms for the propagation process.

The Multi-agent STN (MaSTN) is an extension to the STN machinery where each agent owns and maintains its own STN but also shares constraints with other agents. These constraints couple time points belonging to different agents which we refer as inter-agent constraints. While executing the MaSTN, agents need to communicate with each other during the propagation process. In many multi-agent settings, communication between agents may be unreliable or may have a high cost. In such settings, one reasonable strategy would be to cooperatively constrain time points participating in inter-agent constraints prior to execution at the cost of some loss in flexibility to the overall MaSTN, a process which is referred to as decoupling in literature. A decoupled MaSTN eliminates the need for communication between agents during execution and any propagation is limited to within an agent's own STN. The challenge is to decouple the MaSTN in a distributed fashion without too much compromise of flexibility in agent schedules.

In this thesis, we propose a new distributed algorithm for decoupling the Multi-agent Simple Temporal Network (MaSTN) problem. The agents cooperatively decouple the MaSTN while simultaneously optimizing a sum of concave objectives local to each agent. Several schedule flexibility measures are applicable in this framework. We pose the MaSTN decoupling problem as a distributed convex optimization problem subject to constraints having a block angular structure; we adapt existing variants of Alternating Direction Method of Multipliers (ADMM) type methods to perform decoupling optimally. The resulting algorithm is an iterative procedure that is guaranteed to converge. Communication only takes place between agents with temporal inter-dependencies and the information exchanged between them is carried out in a privacy preserving manner. We present experimental results for the proposed method on problems of varying sizes, and demonstrate its effectiveness in terms of solving quality and computational cost. We then provide 2 extensions to our baseline algorithm and present their empirical performance. One of the algorithms aims at reducing the number of iterations for convergence while the other aims to compute decoupling quickly with bounds on the quality of the decoupling solution with respect to the optimal. We finally conclude by summarizing the contributions of this thesis and identify future directions of research.

Contents

Introduction

One basic challenge in many multiagent planning and scheduling domains is that of managing interdependencies between the planned activities of different agents over time. Effective, optimized multiagent schedules often require tight coordination among agents, but such schedules also tend to be brittle in dynamic and uncertain execution environments. To hedge against this problem, so-called "flexible times" planning and scheduling procedures (e.g., [Smith and Cheng, 1993; Laborie and Ghallab, 1995; Smith *et al.*, 2000; Policella *et al.*, 2009]) have been increasingly adopted as a basic means of retaining temporal flexibility in the plan. These procedures rely on an underlying representation of the plan/schedule as a Simple Temporal Network (STN) [Dechter *et al.*, 1991], an edge weighted graph that encodes all constraints between start and end points of activities in the plan and enables efficient determination of feasible activity execution intervals through temporal constraint propagation. In distributed planning settings, however, where each agent maintains a local STN that represents its portion of the overall multiagent plan, there is the additional complication of maintaining consistency across agent schedules as local planning constraints change during execution. If communications are reliable, then it is possible to maintain consistency by simply distributing the constraint propagation process [Smith *et al.*, 2007]. However in cases where communications are unreliable, a better solution might be to compromise somewhat on flexibility and decouple the subplans of constituent agents.

The concept of temporal decoupling was first introduced in [Hunsberger, 2002]. In brief, it entails elimination of all inter-agent temporal constraints (i.e., all constraints relating time points in the local STNs of two agents). For each such constraint, the set of feasible time point assignments admitted by this constraint is partitioned into two consecutive subintervals, and then new local constraints are substituted that allocate these subintervals respectively to both agents (The position where the interval of feasible assignments is split is sometimes referred to as the decoupling point.) Once a temporal decoupling is obtained, an individual agent is free to adjust its local schedule within the stated local bounds of its STN without disrupting the consistency of the overall joint multiagent plan. Initial centralized algorithms for computing a temporal decoupling were proposed in [Hunsberger, 2002], and in [Planken *et al.*, 2010] the tractability of the optimal problem was first established. Subsequently, [Boerkoel and Durfee, 2011, 2013] formulated temporal decoupling as a distributed multiagent problem, and proposed a distributed algorithm for decoupling an interdependent multiagent schedule (however with no guarantee of optimality). Other recent work [Wilson *et al.*, 2013; Witteveen *et al.*, 2014] has focused alternatively on developing more accurate characterizations of flexibility, which provide a stronger basis for optimization.

One advantage of a distributed approach to decoupling in several domains is its potential for sensitivity to privacy concerns. Consider a civil construction project, where different subcontractors are engaged to perform specific construction tasks (e.g., plumbing, painting). These contractors need to cooperatively construct a plan for the project subject to temporal dependencies (e.g., walls must be plastered before painted). At the same time, each subcontractor has its own private constraints (e.g., other contracting commitments and schedules), which would not normally be revealed to other contractors for business reasons. Decoupling the project plan gives individual contractors the flexibility to locally reorganize work plans to better accommodate new commitments without need for renegotiation with other contractors.

In this thesis, we describe and analyze a new distributed algorithm for temporal decoupling that explic-

itly addresses agent privacy concerns. Our approach, which adapts recently popularized Alternating Direction Method of Multiplier (ADMM) type methods (e.g., [Boyd *et al.*, 2011]), is designed to share only the minimal amount of information about inter-agent dependencies necessary to optimally decouple, and thus agent privacy is preserved to the maximal extent possible. Our approach also offers several other advantages over previously developed decoupling algorithms. First, it provides a distributed decoupling procedure that, in contrast to the heuristic procedure of [Boerkoel and Durfee, 2013], is guaranteed to produce the optimal decoupling for a given decoupling objective, provided that the objective is a concave function. Second, by supporting concave objectives, the algorithm is not bound to a single definition of flexibility but can operate with several (including those specified in [Boerkoel and Durfee, 2013; Wilson *et al.*, 2013; Witteveen *et al.*, 2014] which are all linear). Third, it is even possible to ascribe different decoupling objectives to individual agents in circumstances where agents may be more self-interested and have different goals.

The remainder of the thesis is organized as follows. First we provide a brief introduction to STN and summarize the multiagent decoupling problem in chapter 1. In chapter 2 we present our ADMM-based decoupling algorithm. We analyze the performance of our ADMM-based approach on the set of reference problems first introduced in [Boerkoel and Durfee, 2013] using their flexibility metric. We report the algorithm's empirical performance and discuss its communication requirements. In chapter 3, we present some useful extensions to the algorithms described in chapter 2, focussing on reducing iterations for quick feasible MaSTN decouplings and some discussions on division of social welfare (Flexibility) among the agents. Finally, we draw some conclusions and identify some directions for future research.

Chapter 1

Multi-Agent Simple Temporal Networks And Decoupling

We begin this chapter by giving a brief introduction to Simple Temporal Networks (STN) and introduce some basic concepts concerning it. We then introduce the Multi-agent version of the STN called MaSTN. We then introduce the concept of decoupling the MaSTN and establish the problem this thesis aims to solve.

1.1 Simple Temporal Networks

STNs [Dechter *et al.*, 1991] are concerned with finding feasible solutions to binary TCSPs. A binary TCSP is basically a collection of time points $\mathcal{T} = [z, t_1, .., t_N]$ and a set of binary constraints $\mathcal{C}$ of the form $t_i - t_j \leq c_{ij}$, where $t_i, t_j \in \mathcal{T}$ and z is a fixed arbitrary time point reference. Following the convention of [Hunsberger, 2002], we represent the binary TCSP by the pair $< \mathcal{T}, \mathcal{C} >$. A feasible solution is an assignment of values to the elements in $\mathcal{T}$ that respects the constraints in $\mathcal{C}$. For a given TCSP, their corresponding STN representation allow us to efficiently answer queries of the following types.

1. What possible values any time point can assume in the space of feasible solutions relative to reference z?
2. How are any two time points related to each other in the space of feasible solutions?

Consider the following binary TCSP example. Anna has to decide when to leave office in order to pick her children from school and groceries en-route. Historically, Anna leaves office between 3:45 PM and 4:30 PM. It takes her at least 20 minutes to get to the grocery store from the office. At the grocery store, it would normally take her between 10-20 minutes to complete her shopping. From then on, it takes her between 15 and 20 minutes to reach her children's school. However, the school breaks for the day only at 5:00 PM and she must pick up her kids by no later 5:10 PM so that she could drive her children to soccer practice on time. We represent the problem in the form of a binary TCSP. Let t_O be the time Anna decides to leave the office, t_G is the time point Anna reaches the grocery store, t_L corresponds to the event Anna completes her shopping and t_S is the time she reaches school. Let z=4.00 PM be our reference point. Then $\mathcal{C}$ contains the following constraints where the values for the constraints are in minutes.

$$\begin{aligned} -15 \leq\ & t_O - z \leq 30 \\ 20 \leq & t_G - t_O \\ 10 \leq & t_L - t_G \leq 20 \\ 15 \leq & t_S - t_L \leq 25 \\ 60 \leq\ & t_S - z \leq 70 \end{aligned}$$

Apart from finding feasible solutions to our problem, some canonical type questions that are useful from a planning perspective are the following.

1. What is the latest time by which Anna can enter the grocery store without sabotaging her chances of meeting her deadline to pick up her children from school? In other words, this question is interested in the domain of time point t_G.

2. What is the maximum time she could spend travelling and shopping from the moment she left office and reached her kid's school? In other words, this question is interested in the minimal constraint between t_O and t_S.

To answer these questions, we first convert these time points and constraints into STN form. An STN is a directed graph where the vertices are the time points and edge values are the constraints. Say for instance if we have a constraint of the form $t_i - t_j \leq c_{j,i}$ in $\mathcal{C}$, then we graphically represent this constraint using a directed arc from time point t_j to t_i with label $c_{j,i}$. Fig. 1.1 is the STN representation for our binary TCSP. We may interpret this directed graph as a distance graph where the edge weights denote path distances.

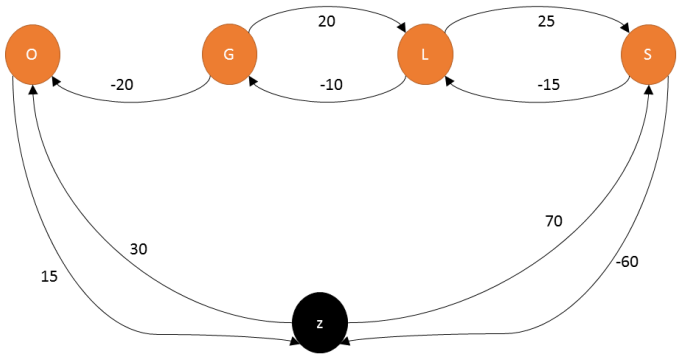

Figure 1.1: STN / Distance graph representation for our problem

To answer the queries we posted earlier, we simply apply an all pair shortest path (APSP) distance algorithm such as Floyds-Warshall on the distance graph shown in Fig 1.1. We could then simply read off the answers to our queries from the shortest path distance graph we computed in Fig 1.2. We refer to the shortest path distance graph as the minimal STN for our problem since the weight on each edge is the shortest path distance between vertices connected across each edge. If the distance graph does not contain any negative cycles, then the set of constraints $\mathcal{C}$ is feasible. A STN that is feasible is also called a consistent STN.

1.1.1 Computing a feasible solution from the minimal STN

Using our example, we illustrate the process of computing a feasible solution from a minimal STN. In the minimal STN computed in Fig 1.2, we start assigning values to the time points in any order. However, each successive assignment of time point must be checked against previous assignments for feasibility w.r.t constraints between them. For example, let us suppose we decide to assign the values to the time points in the following order t_O, t_G, t_L, t_S , then the following steps illustrates the process-:

1. t_O needs to only satisfy $t_O - z \leq 25$ and $z - t_O \leq 15$, where since z is reference its value is 0. Let us choose t_O = 5. If we denote the value on the edge connecting vertex t_O to z as $C_{t_O,z}$ and the edge connecting z to t_O as C_{z,t_O} then assigning a value is to t_O equivalent to choosing a value from the interval $[-C_{t_O,z}, C_{z,t_O}]$.

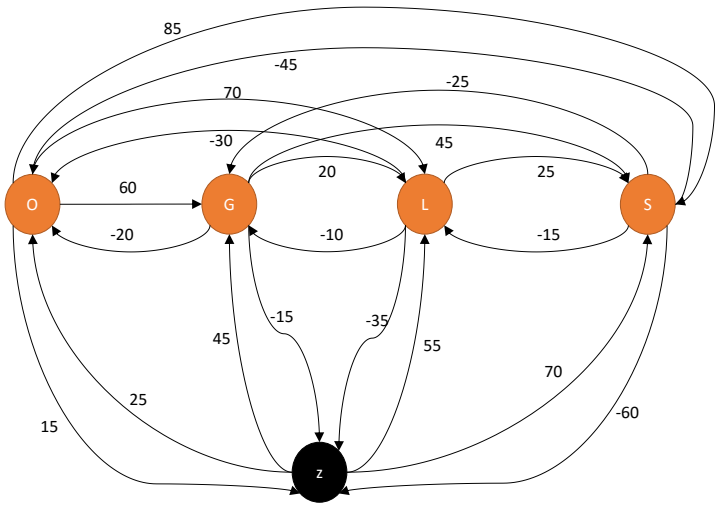

Figure 1.2: Corresponding minimal STN for constraints STN in Fig 1.1

2. t_G needs to satisfy

$$\begin{aligned} t_G - z &\leq 45 \\ z - t_G &\leq -15 \\ t_G - t_O &\leq 60 \\ t_O - t_G &\leq -20 \end{aligned}$$

Substituting t_O = 5 and z = 0, we find t_G = 40 is a feasible assignment to the constraints.

3. Similarly t_L must satisfy the following constraints

$$\begin{aligned} t_L - z &\leq 55 \\ z - t_L &\leq -35 \\ t_L - t_O &\leq 70 \\ t_O - t_L &\leq -30 \\ t_L - t_G &\leq 20 \\ t_G - t_L &\leq -10 \end{aligned}$$

Substituting t_O = 5, z = 0, t_G = 40 we find that t_L = 50 is a feasible assignment to the constraints.

4. Finally for setting the value of t_S, we apply the procedure identical to above.

$$\begin{aligned} t_S - z \leq\ 70,\ z - t_S \leq -60 \\ t_S - t_L \leq 25,\ t_L - t_S \leq -15 \\ t_S - t_G \leq 45,\ t_G - t_S \leq -25 \\ t_S - t_O \leq 85,\ t_O - t_S \leq -45 \end{aligned}$$

Substituting t_O = 5, z = 0, t_G = 40 and t_L = 50 we find that t_S = 65 is a feasible assignment to the problem. [Dechter *et al.*, 1991] prove that all feasible solutions to the binary TCSP can be generated in this manner.

In real world planning and scheduling applications, the assignment of values to time points are not entirely under an agent's control. Suppose an agent has to execute two tasks in succession, then start time of the second task is dependent on the end time of the first task. Owing to uncertainties in the real world, it will mostly be the case that the end time of the first task is an observable quantity only. The process during which the time point values are assigned\observed by an agent is called execution of the plan. STNs provide efficient algorithms to check whether a plan is feasible during execution. Also, from our example above, it should be intuitively obvious that when we assign a value to a time point it may reduce the domain for the unassigned time points. Having the reduced domain information for an unassigned time point can be very useful in planning applications from a computational efficiency point of view. The process of propagating the effects of assigning values to time points on the domain of unassigned time points is captured in a process called constraint propagation. Next, we explain this concept.

1.1.2 Constraint Propagation

Using the values we assigned in the previous example, we explain the constraint propagation process.

1. We began by setting t_O = 5. This is equivalent to setting the weight of the edge connecting vertex z to t_O equal to +5 and the edge connecting t_O to z as -5 in the minimal STN shown in Fig 1.2. Let us call the modified edges e_1 and e_2. Setting e_1 =5 and e_2 = -5, limits the domain of t_O to the singleton $\{5\}$. The new graph is shown in Fig 1.3.

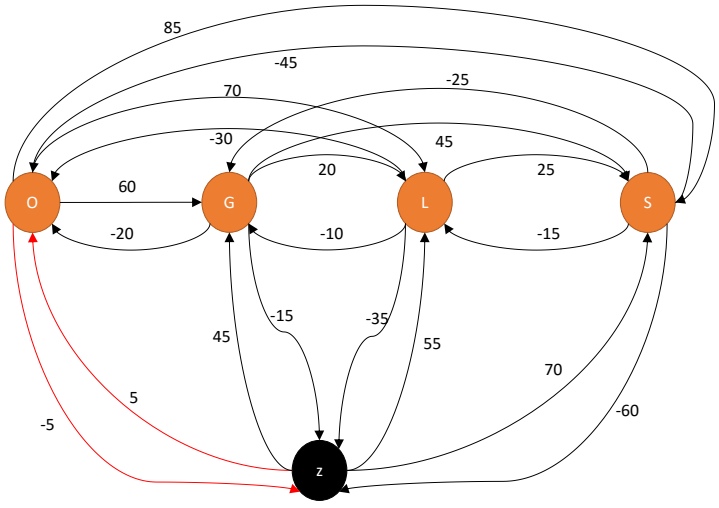

Figure 1.3: The modified edges are highlighted in red.

2. To propagate the effects of modifying e_1 and e_2, we recompute all the shortest paths for the graph shown in Fig 1.3. The new minimal STN is shown in Fig 1.4. If the new shortest path distance graph contains any negative length cycles, then STN is no longer consistent implying that any further assignment of time point values cannot be feasible to the original TCSP.

3. Next when setting t_G = 40, we modify weights of edges connecting vertices z to t_G and t_G to z to +40 and -40 respectively, similar to step 1. On this new modified graph we recompute the shortest paths just as in step 2. We repeat the process for other time points.

Notice, if one compares the edge weights on the minimal STNs before and after setting the value to a time point, one would observe the edge weights of all edges never increased but possibly decreased. This observation is in line with our observation that the domain may decrease. Another way of reasoning is through the distance

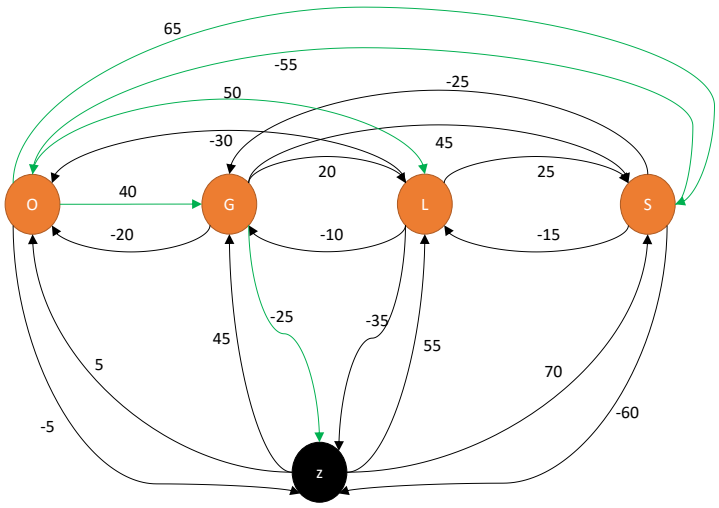

Figure 1.4: The edges that became shorter are highlighted in green.

graph interpretation of the STN. Observe when we modified e_1, e_2 in step 1, we replaced the weights on these edges with values strictly no greater than they previously were; this will always be the case as long the newly assigned time point value is feasible with previously assigned time point values. When we shorten the distances of some edges on a distance graph, every other shortest path between any pair of vertices cannot increase but may decrease. We call this phenomenon contraction of the STN. Hence assigning values to time points in a STN can only contract the STN. Later, we discuss the importance of the constraint propagation process for the Multi-agent case.

1.2 Multi-Agent Simple Temporal Network (MaSTN)

A MaSTN is a simple extension to the STN framework, where we additionally assign ownership of each time point to one of the agents. Accordingly, we associate with each $t_i \in \mathcal{T}\backslash\{z\}$ an agent as its owner $O(t_i)$, and distinguish two types of constraints within the set $\mathcal{C}$ associated with the overall MaSTN: (1) intra-agent constraints, which are local to a given agent and considered private (i.e., $O(t_i) = O(t_j)$ or one of t_i, t_j is z), and (2) inter-agent constraints, which couple the STNs of different agents (i.e., $O(t_i) \neq O(t_j)$). The owner for each time point is always specified in the problem, for simplicity we will let all agents be owners to the reference time point z, although its value is always pre-assigned and immutable. Further, it is assumed that each agent assigns values to the time points it owns, we refer to this process as an agent maintaining its own private STN. The induced sub-graph formed by the vertices owned by an agent in the MaSTN is the private STN of that agent. An MaSTN without assignment of values to any of the time points represents a joint multi-agent plan. The process during which the agents assign values to their time points is referred to as execution of the joint-plan.

Functionally, when an agent assigns\observes time point values, the assigned values must be feasible w.r.t constraints its shares with its own previously assigned time points as well as with time points assigned by other agents. From the constraint propagation process previously described, it is intuitively clear that when an agent assigns a value to a time point it can affect the domain of time points belonging to other agents; these other agents must then factor those effects before they assign values to any of their unassigned time points. Propagating these effects require the agents to communicate whenever a time point is assigned in any of the agents. A more subtle observation is that the agents cannot asynchronously assign values to their time points. In many applications, communication between agents may be unreliable or too expensive during execution of the plan. Thus maintaining the joint multi-agent plan using the STN framework may become difficult. Secondly,

owing to the synchronous update process, agents are highly interdependent upon one another for successful execution of the joint-plan. These issues naturally leads us to the concept of decoupling, where we break the interdependencies between agents in the joint plan prior to execution.

1.3 Decoupling The MaSTN

The overall goal of decoupling is to decompose the MaSTN into their component STNs agent-wise such that each STN which is private to an agent can be independently maintained by the agent with assurance that the overall MaSTN remains globally consistent i.e. any assignment of values which is feasible locally to an agent should automatically satisfy the inter-agent constraints specified in the problem. We explain the basic problem in two steps with the aid of the simple two-agent example shown in Fig.1.5. In this example, we assume the following notation and definitions:

n_A $\rightarrow$ Number of time points owned by agent A
A_i $\rightarrow$ Time point i in agent A , $i \in \{1, \ldots, n_A\}$
T_A $\rightarrow$ $\{A_i\}_{i=1}^{n_A}$
n_B $\rightarrow$ Number of time points owned by agent B
B_i $\rightarrow$ Time point j in agent B , $j \in \{1, \ldots, n_B\}$
T_B $\rightarrow$ $\{B_j\}_{j=1}^{n_B}$
z $\rightarrow$ Common reference time point for all agents,$T_A \cap T_B = \{z\}$
C_{V_i,V_j} $\rightarrow$ Problem specified constraint value for edge from time point V_i to time point V_j where $V_i, V_j \in T_A \cup T_B$, and $V_i \neq V_j$.

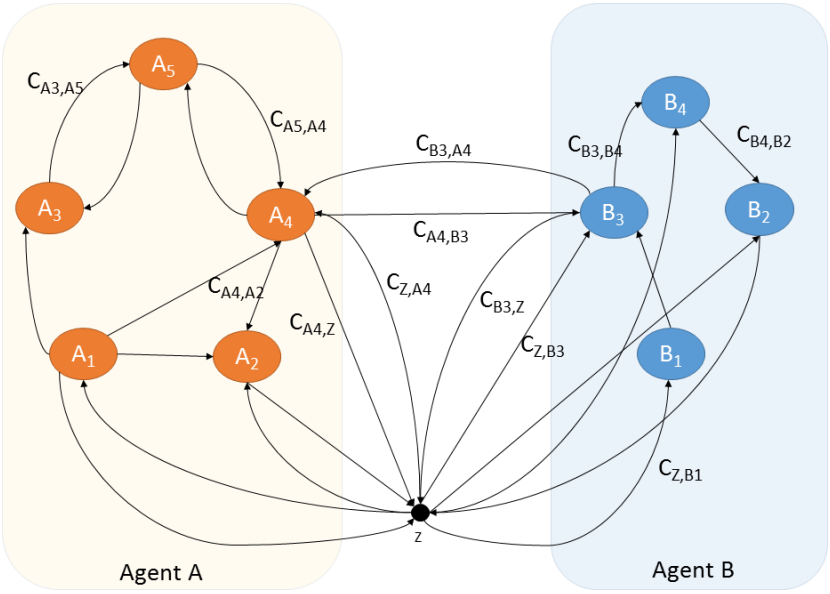

Figure 1.5: Decoupling a 2 agent MaSTN.

We partition the constraint set $\mathcal{C}$ into inter and intra-agent constraints. The constraint linking time points A_3 to A_5 is an example of an intra-agent constraint and represented as a directed edge whose edge weight is given by C_{A_3,A_5}. Similarly, the edge connecting time points A_4 and B_3, is an example of an inter-agent constraint. The constraint set $\mathcal{C}$ is specified in the problem and is equivalently represented in the STN framework. For example, if the problem specifies the following constraints:

$$\begin{aligned} A_4 - A_2 &\leq 120 \\ A_2 - A_4 &\leq 60 \end{aligned}$$

then, we assign the values $C_{A_2,A_4} = 120$ and $C_{A_4,A_2} = 60$.

Using just the intra-agent constraints specified for each agent, agents A and B can separately compute STNs that are minimal with respect to intra-agent constraints [Dechter *et al.*, 1991]. All pairs of distinct vertices within the computed STNs are now connected through a directed edge. We denote the edge connecting vertices i and j for agent A's minimal STN with respect to intra-agent constraints as w_{A_i,A_j}. Intuitively the interval $[-w_{z,A_i}, w_{A_i,z}]$, represents the bounds on the value that time point A_i can be assigned in any feasible schedule subject to just the intra agent constraints. We denote a vector containing all the edges in the just computed minimal STN of agent A as $\mathbf{W_A} = \left[\{w_{A_i,A_j}\}\right], \forall i,j \in \{1,\ldots,n_A\}, i \neq j$, i.e. $\mathbf{W_A} = [w_{A_1,A_2}, w_{A_1,A_3}, \ldots, w_{A_1,A_{n_A}}, w_{A_2,A_1}, w_{A_2,A_3}, \ldots, w_{A_2,A_{n_A}}, \ldots, w_{A_{n_A},A_{n_A-1}}]^T$.
We are now ready to formulate our two agent decoupling problem.

1.3.1 Constraints

The central idea behind decoupling is to tighten (shorten) the bounds on time points that are involved in inter-agent constraints such that any assignment of values to those time points within those bounds can be decided independently by an agent without need for coordination with other agents participating in inter-agent constraints, while also ensuring such an assignment is feasible with respect to an agent's own intra-agent constraints. In our discussion in section 1.1.2 on constraint propagation, we showed how assignment of a value to a time point can affect the domain of other unassigned time points. However, it is intuitively clear that merely tightening bounds on time points involved in inter-agent constraints (i.e. red edges in Fig 1.6 for example in Fig 1.5) can also affect all other edge weights in the minimal STN. Hence, any procedure for tightening must factor this into the propagation process. The approach we adopt is to start from the minimal STNs computed using just intra-agent constraints (i.e., $\mathbf{W_A}$ for each agent $\mathbf{A}$), and then let the agents cooperatively further tighten their corresponding STN edges to decouple the MaSTN while also ensuring that each agent's private STN remains consistent. This ensures that the intra-agent constraints during decoupling is always automatically satisfied. We denote the vector containing edges in the further tightened minimal STN of agent A as $\mathbf{P_A} = [\{p_{A_i,A_j}\}], \forall i,j \in \{1,\ldots,n_A\}, i \neq j$ (refer to the definition of $\mathbf{W_A}$ above for expansion). In our discussions to follow, we let the edge weights on each agent's STN (i.e. $\mathbf{P_A}, \mathbf{P_B}$) be the variables. The overall aim is compute the edge weights on the minimal STN for each agent that decouples the agent STNs while simultaneously respecting intra-agent and inter-agent constraints specified in the original problem. We present necessary constraints on $\mathbf{P_A}, \mathbf{P_B}$ that the agents need to satisfy to achieve this objective.

Since time points A_4 and B_3 for our problem in Fig.1.6 participate in an inter-agent constraint, decoupling the time points for that constraint yields the following decoupling equations:

$$(DEC)$$

$$p_{A_4,z} + p_{z,B_3} \leq C_{A_4,B_3} \tag{1.1}$$

$$p_{z,A_4} + p_{B_3,z} \leq C_{B_3,A_4} \tag{1.2}$$

We show why Eqns (1.1) and (1.2) decouple agents A and B. Any feasible assignment of values to t_{A_4} and t_{B_3} must satisfy

$$-p_{A_4,z} \leq t_{A_4} \leq p_{z,A_4} \tag{1.3}$$

$$-p_{B_3,z} \leq t_{B_3} \leq p_{z,B_3} \tag{1.4}$$

The original problem requires the following the constraint to hold

$$-C_{A_4,B_3} \leq t_{A_4} - t_{B_3} \leq C_{B_3,A_4} \tag{1.5}$$

It is easy to see that satisfying Eqns (1.3) , (1.4) , (1.1) and (1.2) automatically satisfy Eqn (1.5). Also, when the conditions in Eqns (1.1) and (1.2) are satisfied, agents A and B can independently assign values to t_{A_4} and t_{B_3} respectively according to Eqn (1.3) and (1.4), implying the dependency between agents have been decoupled for execution. From Eqns (1.1) and (1.2) it is easy to see that each inter-agent constraint requires 2 constraints

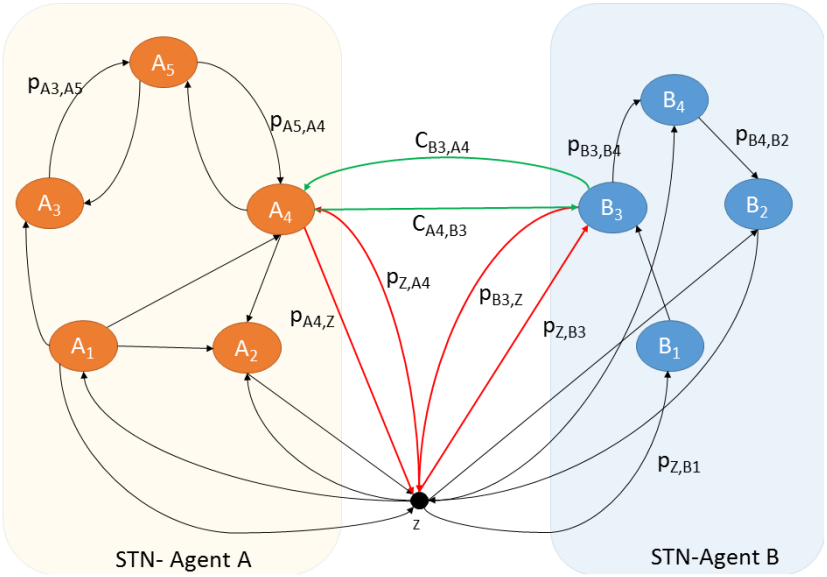

Figure 1.6: Corresponding minimal STN for problem in Fig 1.5. Green edges are inter-agent coupling edges. Red edges are tightened to dominate green edges.

for decoupling. Hence the number of DEC constraints is twice the number of inter-agent constraints.

For the inequalities that follow in this section we present them for only agent A, it is analogous for agent B. We also remind the reader that a minimal STN can be interpreted as a shortest path distance graph. Hence, all edges in the minimal STN for each agent must obey triangle inequality.

$$(TRI)$$
$$p_{A_i,A_j} - p_{A_i,A_k} - p_{A_k,A_j} \leq 0,\ \forall i,j,k \in \{1,\ldots,n_A\} \tag{1.6}$$

where $i \neq j \neq k$. The number of TRI constraints for agent A is $\mathcal{O}(n_A^3)$. While Eqn.(1.6) ensures that every closed path of length 3 on the STN is consistent, the following inequalities ensure that every path of length 2 is also consistent.

$$(CON)$$
$$-p_{A_i,A_j} - p_{A_j,A_i} \leq 0, \forall i,j \in \{1,\ldots,n_A\}, \& \, i > j \tag{1.7}$$

The number of CON constraints for agent A is $\mathcal{O}(n_A^2)$. The upper bound value for each edge p_{A_i,A_j} is derived by noting that the decoupling process only shortens distances (edges) on the distance graph. For any valid p_{A_i,A_j}, we have

$$(UB)$$
$$p_{A_i,A_j} \leq w_{A_i,A_j} \tag{1.8}$$

The specification of all aforementioned constraints was previously developed in [Planken *et al.*, 2010]. To this formulation we introduce a way for computing the lower bounds for p_{A_i,A_j}. The sum of the edge weights along any closed path of lengths 2 and 3 must necessarily be ≥ 0 on any shortest path distance graph. For any valid

p_{A_i,A_j}, we have:

$$
\begin{aligned}
& & (LB) & \\
p_{A_i,A_j} & \geq max(-p_{A_j,A_i}, -p_{A_i,A_k} - p_{A_k,A_j}) & & \\
& \forall k \in \{1,\ldots,n_A\}\backslash\{i,j\} & & \\
& \geq max(-w_{A_j,A_i}, -w_{A_i,A_k} - w_{A_k,A_j}), & & (1.9) \\
& \forall k \in \{1,\ldots,n_A\}\backslash\{i,j\} & &
\end{aligned}
$$

where Eqn.(1.9) is obtained from Eqn.(1.8). The LB constraints proposed are not necessary but can significantly improve the numerical efficiency of the solving procedure to be described in chapter 2. From Eqns (1.8) and (1.9), it is easy to see that the number of LB and UB constraints are $\mathcal{O}(n_A^2)$ for agent A. In the following chapters, it will be shown all DEC, TRI, CON inequalities will be converted into equality constraints through the addition of slack variables. Hence the number of LB and UB constraints for agent A will increase to $\mathcal{O}(n_A^3) + \#$ inter-agent constraints agent A participates in, to accommodate slack variables for inequalities transformed to equality constraints.

1.3.2 Optimal Decoupling

As the constraints defined in the previous section are all affine, a number of decoupling solutions can exist for a given MaSTN, and our objective in this paper is to find the optimal decoupling. We assume a proper concave function ($Flex$) for defining the flexibility of a STN and parametrize it by the STN's edge weights. As the decoupling process decomposes the MaSTN into individual STNs, one for each agent, we define the social welfare as the sum of the flexibilities of each agent's STN. By optimal decoupling, we aim to maximize the social welfare of the decoupled STNs. Previously developed flexibility measures (e.g. [Boerkoel and Durfee, 2013; Wilson *et al.*, 2013]) are all applicable in this framework. The resulting objective has a separable form with respect to the agents and hence is particularly suitable in applications that require minimal communication between agents and/or where certain privacy requirements exist.

1.3.3 Privacy Considerations

As indicated earlier, the time points and constraints in an agent's STN can be partitioned into private and shared variables. In Fig.1.6, A_4 is a shared time point while elements in the set $\{T_A - \{A_4\}\}$ are private to agent A. For the problem in Fig.1.6, Eqns.(1.1) and (1.2) suggest that the minimal information that agent B requires from agent A are the edges $p_{A_4,z}$ and p_{z,A_4}. Similarly, agent A's inference over the time points and constraints in agent B should be restricted to $p_{B_3,z}$ and p_{z,B_3}. Let us consider a case where we add a third agent C to the problem in Fig.(1.6) and include an inter-agent constraint between time points A_3 and C_2. Although, A_4 and A_3 are both shared variables, since agent C is coupled with agent A only via time point A_3, the decoupling algorithm should be able to restrict $C's$ inference over agent $A's$ variables to just $p_{A_3,z}$ and p_{z,A_3}. This minimal amount of information sharing between agents is the privacy specification we consider in this work. Such a specification has the additional advantage of minimizing the communication overhead between agents. In comparison, the method of [Boerkoel and Durfee, 2011] entails the establishment of a chordal graph between shared time points of all participating agents. Depending on the graph's topology, agents that were not constrained with each other through explicit specification of temporal constraints between their respective time points, may be required to communicate with each other to correctly establish a minimal MaSTN.

In the next chapter, we propose distributed algorithms that produces optimal decoupling of a MaSTN under privacy considerations mentioned in section 1.3.3. The extent to which we preserve privacy will become clear in the algorithm description.

Chapter 2

Distributed Decoupling

We represent the N agent decoupling problem in a convex optimization framework. Following the convention introduced previously, we denote $\mathbf{P_i}$ as the vector of all edges in the STN of agent i and the problem is to find an optimal assignment of values to $\mathbf{P_i}, i \in \mathcal{V} = \{1, \ldots, N\}$ subject to the DEC, TRI, CON , LB and UB constraints. For the rest of the thesis, we use numerical values for naming agents. The decoupling problem in matrix notation is represented as:

Primal

$$\max_{\mathbf{P_1},\ldots,\mathbf{P_N}} \sum_{i=1}^{N} Flex_i(\mathbf{P_i})$$

s.t.

$$\begin{bmatrix} E_1 & E_2 & . & . & E_N \\ D_1 & & & & \\ & D_2 & & & \\ & & . & & \\ & & & . & \\ & & & & D_N \end{bmatrix} \begin{bmatrix} \mathbf{P_1} \\ \mathbf{P_2} \\ . \\ . \\ \mathbf{P_N} \end{bmatrix} \leq \begin{bmatrix} b \\ c_1 \\ c_2 \\ . \\ c_N \end{bmatrix}$$

$$L_1 \leq \mathbf{P_1} \leq U_1, \ \ldots, \ L_N \leq \mathbf{P_N} \leq U_N$$

where,

$\mathbf{P_i} \rightarrow$ Vector of edge weights in agent $i's$ STN

$|\mathbf{P_i}| \rightarrow$ Length of the vector $\mathbf{P_i}$

$Flex_i(\cdot) \rightarrow$ Proper concave function outputting flexibility of agent i's STN

$\sum_{i=1}^{N} E_i\mathbf{P_i} \leq b \rightarrow$ DEC constraints of all agents

$D_i\mathbf{P_i} \leq c_i \rightarrow$ TRI and CON constraints for agent i

$L_i \leq \mathbf{P_i} \leq U_i \rightarrow$ LB and UB constraints for $\mathbf{P_i}$

$[E] \rightarrow [E_1\ E_2\ \ldots\ E_N] \in \mathbb{R}^{W \times \sum_{i=1}^{N} |\mathbf{P_i}|}$

Notice that the primal formulation is a maximization problem defined over a block diagonal constraint set. Matrices D_i and c_i for agent i is obtained by analogously representing all constraints in Eqns (1.6) and (1.7) for agent A in matrix notation. All DEC constraints in the multi agent problem are collectively represented as $\sum_{i=1}^{N} E_i\mathbf{P_i} \leq b$, let the number of DEC constraints be W. We denote $E_{i,r}$ as a row vector and define it to be the entries in row r corresponding to agent $i's$ matrix E_i, since each row in E represents a DEC constraint

between exactly 2 agents, entries in $E_{i,r}$ corresponding to non-interacting agents of row r are all set to 0. We first show how the primal problem is generically solved and later present the necessary modifications to meet the privacy requirements in our problem. For the reminder of our work, we convert DEC constraints into equality constraints through the addition of slack variables.[1]

$$\text{Dual}$$
$$\min_{\lambda} \sum_{i=1}^{N} \underbrace{\sup_{\substack{D_i\mathbf{P_i}\leq c_i \\ L_i\leq \mathbf{P_i}\leq U_i}} Flex_i(\mathbf{P_i}) - \lambda^T (E_i\mathbf{P_i} - b_i)}_{G_i(\lambda)}, \text{ where } \sum_{i=1}^{N} b_i = b \tag{2.1}$$

To respect privacy, we set $b_i^r = \frac{b^r}{2}$ if $E_{i,r}$ is a non-zero vector, else $b_i^r = 0$, where b_i^r represents r^{th} row entry in vector b_i. Many distributed algorithms solve the primal problem by transforming to its dual. Every consistent MaSTN has a valid decoupling [Hunsberger, 2002] i.e. $\exists$ solution satisfying constraints in the primal; combining this fact with our assumption on the flexibility metric ensures that the primal and dual optimal objective values are equal by strong duality. Moreover, for consistent MaSTNs our flexibility metric outputs a finite optimal objective, implying that dual is solvable and the primal optimal solution is attainable from dual optimal solution [Boyd and Vandenberghe, 2004]. Hence we can equivalently work with the dual form since Eqn (2.1) suggests that computations can be carried out in parallel by each agent. Solution strategies for the dual typically consists of an inner loop where each agent independently solves the minimization problem for the current estimate of the shared vector $\lambda \in \mathbb{R}^W$, and an outer loop where solutions from all agents are accumulated to update λ.

Recently, ADMM [Boyd *et al.*, 2011] type methods have been popularized as having improved convergence properties over prior art in terms of convergence with respect to primal variables $\mathbf{P_i}$ in Eqn (2.1). The original ADMM algorithm takes a sequential approach to solving Eqn (2.1) where each agent take turns in updating its variables. This limitation has been addressed in Consensus ADMM type methods such as the Dual Consensus ADMM (DC-ADMM) method [Chang *et al.*, 2015]. DC-ADMM is attractive in our context since it helps improve agent utilization. The philosophy of DC-ADMM is to have each agent maintain a private copy of λ (possibly with different values at the start of the algorithm), and drive their private estimates of λ to consensus through neighbour wise consensus constraints such as Eqn (2.2).

$$\text{DC-ADMM Primal}$$
$$\min_{\substack{\lambda_1,\ldots,\lambda_N \\ \{m_{.,.}\}}} \sum_{k=1}^{N} G_k(\lambda_k)$$
$$\text{s.t.}$$
$$\lambda_i = m_{i,j},\ \lambda_j = m_{i,j}, \text{ where } j \in Neigh(i),\ \forall (i,j) \in \mathcal{H}, \tag{2.2}$$

where, $\lambda_i \in \mathbb{R}^W$ is agent $i's$ estimate, $Neigh(i)$ is the set of agents that are neighbours to agent i with all of whom i exchanges its estimate of the lagrange dual vector λ_i. $m_{i,j}$ is a consensus slack variable $\in \mathbb{R}^{W\times 1}$ indicating agent j is a neighbour of agent i. The set $\mathcal{H}$ contains all agent neighbour pairs where the set member (g,h) with agents g and h is considered different from the set member (h,g). Mathematically we represent it as $\mathcal{H} = \{\{i,j\}_{j\in Neigh(i)}\}_{i=1}^{N}$; the notation $\{m_{.,.}\}$ denotes the set of all valid consensus variables. G_i is the lagrange dual function for agent i, refer Eqn 2.1 for its definition. Notice that DC-ADMM Primal and the Dual problem in Eqn (2.1) are equivalent. The DC-ADMM Primal problem is solved using the standard ADMM framework; we refer our readers to [Chang *et al.*, 2015] for details.

Algorithmically, at each iteration of DC-ADMM, agents exchange their estimate of the dual vector λ with neighbours and use it to update their own estimate (cf Algorithm 3 in [Chang *et al.*, 2015]). In our problem, if

[1] Since exactly two agents participate in each row/constraint of matrix E, we arbitrarily assign ownership of the slack variable to one of the participating agents (say k) and append it to $\mathbf{P_k}$. We represent the new DEC constraints as $\sum_{i=1}^{N} E_i\mathbf{P_i} = b$.

agent i does not participate in row r, then $E_{i,r} = 0$ and $b_i^r = 0$. This implies that r^{th} row entry of the dual vector is inconsequential to agent i. However, for DC-ADMM to work correctly, agent i is required to transmit values to its neighbours that are inconsequential to itself but may be required for its neighbours. In a certain sense, transmitting these inconsequential values via agent i can be considered as a privacy violation; it is susceptible to manipulation; and is also inefficient to store and transmit.

2.1 Proposed Algorithm

To avoid communication of non-essential and potentially private information, we adjust the DC-ADMM procedure to ensure that an agent creates Lagrange dual variables $\in \mathbb{R}$ only for rows in matrix E that an agent participates in. This also ensures correspondingly fewer consensus constraints in our formulation. We define $R_E(i)$ as a function that outputs a set containing indices of all rows in which agent i is involved in matrix E. The function $Neigh_E(i, r)$ outputs j if agent i shares the constraint with agent j in row r of matrix E. A lagrange dual variable $y_{i,r} \in \mathbb{R}$ is allocated by agent i iff $r \in R_E(i)$, likewise $y_{j,r}$ is allocated by agent j iff $r \in R_E(j)$. We denote $\mathcal{E}$ as the set of all valid agent-row pairs, i.e. $\mathcal{E} = \{\{i, r\}_{r \in R_E(i)}\}_{i=1}^{N}$. Hence the modified DC-ADMM method for the decoupling problem can be represented as:

Modified DC-ADMM Primal

$$\min_{\substack{\{y_{\cdot,\cdot}\}\\ \{t_{\cdot,\cdot}\}}} \sum_{i=1}^{N} \sup_{\substack{D_i\mathbf{P_i} \leq c_i\\ L_i \leq \mathbf{P_i} \leq U_i}} Flex_i(\mathbf{P_i}) - \sum_{r \in R_E(i)} y_{i,r} \left(E_{i,r}\mathbf{P_i} - \frac{b^r}{2} \right)$$

s.t.

$$y_{i,r} = t_{i,r}, \;\; y_{j,r} = t_{i,r}, \text{ where } r \in R_E(i) \text{ and } j = Neigh_E(i, r), \forall (i, r) \in \mathcal{E}$$

where $t_{i,r}$ is a consensus slack variable $\in \mathbb{R}$ and $\{y_{\cdot,\cdot}\}, \{t_{\cdot,\cdot}\}$ denote the set of all lagrange and consensus variables allocated in the above problem.

In Algorithm 1, we present the most generic form of the algorithms that we will be considering in this thesis to solve the Modified DC-ADMM primal problem. Later, we present some empirical results for some special cases of Algorithm 1. We applied Sion's Minimax theorem and standard ADMM updates to the Modified DC-ADMM primal problem for obtaining the expressions in Algorithm 1. The steps taken to derive are identical to those taken in [Chang, 2014] and listed in Appendix A.

We step you through the process in Algorithm 1 below. The algorithm accepts as input the penalty parameter for each agent (i.e. ρ_i for agent i), a threshold PEN_THRESH_ITER $\in \mathbb{Z}^+$, global penalty parameter ρ and initial values for $\{y_{\cdot,\cdot}\}$ and $\{t_{\cdot,\cdot}\}$. At each iteration until termination,

1. All agents independently solve a maximization problem in polynomial time. The solution obtained ($\mathbf{P}_{\cdot}^{(\cdot)}$) will be feasible with respect to intra-agent constraints, with some infeasibility across DEC constraints. $\mathbf{P_i^{(k)}}$ corresponds to assignment of values to agent $i's$ STN at the k^{th} iteration.

2. The variables ($\{y_{\cdot,\cdot}\}$) are updated using edge weights of the agent's STN computed in step 1 and relevant variable values from neighbours communicated during the previous iteration.

3. The updated variables are then communicated to neighbours selectively. For example, if agents i and j share the constraint in the r^{th} row of matrix E, then agent $i's$ estimate of $y_{i,r}$ at the k^{th} iteration (i.e. $y_{i,r}^{(k)}$) is computed using $y_{j,r}^{(k-1)}$ and communicated only to agent j. Similarly, agent j updates and communicates $y_{j,r}^{(k)}$ only to agent i. Recall $R_E(i)$ outputs all relevant rows in matrix E for agent i and $Neigh_E(i, r)$ outputs the neighbour of agent i corresponding to row r iff $r \in R_E(i)$.

Algorithm 1 Modified DC-ADMM For Decoupling

Given: $\rho_i^{(0)}$**, for each agent** $i \in \mathcal{V} = \{1 \dots N\}, y_{i,r}^{(0)}, t_{i,r}^{(0)}, v_{i,r}^{(0)}, w_{i,r}^{(0)} \, \forall \, (i,r) \in \mathcal{E}, k = 1, \rho$, PEN_THRESH_ITER

repeat

for $i \in \mathcal{V}$ in parallel **do**

$$\mathbf{P_i^{(k)}} = \underset{\substack{D_i\mathbf{P_i} \leq c_i \\ L_i \leq \mathbf{P_i} \leq U_i}}{\arg\sup} \; Flex_i(\mathbf{P_i}) \; +$$

1: $$\sum_{r \in R_E(i)} \frac{-1}{2(\rho_i^{(k-1)} + \rho_j^{(k-1)})} \left(E_{i,r}\mathbf{P_i} - v_{i,r}^{(k-1)} - w_{j,r}^{(k-1)} - b_{i,r} + \rho_i^{(k-1)} t_{i,r}^{(k-1)} + \rho_j^{(k-1)} t_{j,r}^{(k-1)} \right)^2$$

, where $j = Neigh_E(i,r)$

2: **for** $r \in R_E(i)$ **do**

$$j = Neigh_E(i,r)$$

$$y_{i,r}^{(k)} = \frac{E_{i,r}\mathbf{P_i}^{(k)} - b_{i,r} - v_{i,r}^{(k-1)} - w_{j,r}^{(k-1)}}{\rho_i^{(k-1)} + \rho_j^{(k-1)}} + \frac{\rho_i^{(k-1)} t_{i,r}^{(k-1)} + \rho_j^{(k-1)} t_{j,r}^{(k-1)}}{\rho_i^{(k-1)} + \rho_j^{(k-1)}}$$

end for

3: Communicate updated y variables to neighbors selectively

4: **for** $r \in R_E(i)$ **do**

$$j = Neigh_E(i,r)$$

$$t_{i,r}^{(k)} = \frac{v_{i,r}^{(k-1)} + w_{i,r}^{(k-1)}}{2\rho_i^{(k-1)}} + \frac{y_{i,r}^{(k)} + y_{j,r}^{(k)}}{2}$$

end for

5: Communicate updated t variables to neighbors selectively

6: **for** $r \in R_E(i)$ **do**

$$j = Neigh_E(i,r)$$

$$v_{i,r}^{(k)} = v_{i,r}^{(k-1)} + \rho_i^{(k-1)}(y_{i,r}^{(k)} - t_{i,r}^{(k)})$$

$$w_{i,r}^{(k)} = w_{i,r}^{(k-1)} + \rho_i^{(k-1)}(y_{j,r}^{(k)} - t_{i,r}^{(k)})$$

end for

7: **if** k $\geq$ PEN_THRESH_ITER

$$\rho_i^{(k)} = \rho$$

else

$\rho_i^{(k)}$ = Update($\rho_i^{(k-1)}, \{y_{i,\cdot}^{(k)}\}, \{t_{i,\cdot}^{(k)}\}$)

end if

8: Communicate updated ρ values to all neighbors

end for

k = k + 1

until Termination Criterion

4. The consensus variables ($\{t_{\cdot,\cdot}\}$) are similarly updated.

5. Consensus variables are communicated to neighbours selectively like in step 3.

6. The dual variables $\{v_{\cdot,\cdot}\}, \{w_{\cdot,\cdot}\}$ are updated.

7. If the current iteration number is less than the threshold PEN_THRESH_ITER, then agents update their local penalty parameter value (ρ_i for agent i), else all agents uniformly set the local penalty parameter to a predefined global value ρ. Notice after PEN_THRESH_ITER iterations, the algorithm behaves like standard ADMM.

8. Each agent communicates its updated penalty parameter value to all its neighbours.

2.1.1 Termination Criterion

We chose to use the extent of infeasibility of the DEC constraints as our termination criterion. If agents i and j share a DEC constraint at row r, using the updates shown in Algorithm 1 we can easily arrive at the following relation

$$\begin{aligned} E_{i,r}\mathbf{P_i^{(k)}} + E_{j,r}\mathbf{P_j^{(k)}} - b^r = &(\rho_i^{(k-1)} + \rho_j^{(k-1)}) \left(y_{i,r}^{(k)} + y_{j,r}^{(k)} - y_{i,r}^{(k-1)} - y_{j,r}^{(k-1)} \right) + \\ &\left(v_{i,r}^{(k-1)} + w_{i,r}^{(k-1)} - \frac{\rho_i^{(k-1)}}{\rho_i^{(k-2)}} (v_{i,r}^{(k-2)} + w_{i,r}^{(k-2)}) \right) + \\ &\left(v_{j,r}^{(k-1)} + w_{j,r}^{(k-1)} - \frac{\rho_j^{(k-1)}}{\rho_j^{(k-2)}} (v_{j,r}^{(k-2)} + w_{j,r}^{(k-2)}) \right) \end{aligned}$$

where k is the iteration number. The term on the right side limits to zero as $k \to \infty$, owing to the convergence characteristics of ADMM. In our set up, the agents determine the magnitude of infeasibility for the DEC constraints at each iteration by simply examining the exchanged variable values as the expression above suggests. When all DEC constraints are satisfied within an acceptable tolerance ϵ_{Inf}, Algorithm 1 terminates.

2.1.2 Discussion Of Algorithm 1

From a communications perspective, each agent in Algorithm 1 communicates trice ($\{y_{i,\cdot}\}, \{t_{i,\cdot}\}$ and ρ_i for agent i) at every iteration with its neighbours. In total all agents collectively transmit $4W + \mathcal{O}(N^2)$ number of variable values at each iteration where each variable takes a value in $\mathbb{R}$. W is the total number of DEC constraints generated in the problem; since each each of the 2 agents participating in an DEC constraint needs to transmit 2 variable values i.e. y and t corresponding to that constraint between themselves, we obtain the multiplier 4. Also until PEN_THRESH_ITER iterations agents transmit their penalty multiplier values to their neighbors, this leads to the $\mathcal{O}(N^2)$ term. In terms of convergence, after PEN_THRESH_ITER number of iterations, Algorithm 1 behaves like standard ADMM since all agents use the same value for the penalty parameter; hence Algorithm 1 is guaranteed to converge.

2.2 Algorithm 2, Special case of Algorithm 1

By instantiating the variables in Algorithm 1 in the following manner, we present a very simple algorithm that requires lesser communication per iteration than Algorithm 1 [Mogali *et al.*, 2016].

1. Set $\rho_i^{(0)} = \rho$, $\forall i \in \{1, \ldots, N\}$ and PEN_THRESH_ITER = 1.
2. Introduce variables $\{p_{\cdot,\cdot}\} = \{p_{i,r}\}_{(i,r)\in\mathcal{E}}$ s.t. $p_{i,r} = v_{i,r} + w_{j,r}$, where $j = Neigh_E(i,r)$. Initialize $p_{i,r}^{(0)} = v_{i,r}^{(0)} + w_{j,r}^{(0)} = 0$, for e.g. $v_{i,r}^{(0)} = w_{j,r}^{(0)} = 0$.

With the above instantiations, Algorithm 1 has been simplified to obtain Algorithm 2. The proof is provided in Appendix B. Notice in Algorithm 2, the consensus variables $\{t_{\cdot,\cdot}\}$ have been completely eliminated. By setting PEN_THREASH_ITER = 1 we have avoided providing a definition for Update($\cdot$) since the definition becomes inconsequential. Moreover, we have replaced $\{v_{\cdot,\cdot}\}, \{w_{\cdot,\cdot}\}$ by $\{p_{\cdot,\cdot}\}$ such that $|\{p_{\cdot,\cdot}\}| = |\{v_{\cdot,\cdot}\}| + |\{w_{\cdot,\cdot}\}|$, where the operator $|\cdot|$ denotes the number of elements in the set. In terms of communication requirements, agents communicate only once per iteration and all agents collectively transmit $2W$ messages per iteration.

Algorithm 2

Given: $\rho, k = 1, y_{i,r}^{(0)} \forall\, (i,r) \in \mathcal{E}, p_{ir}^{(0)} = 0\ \forall\, (i,r) \in \mathcal{E}$

repeat

for $i \in \mathcal{V}$ in parallel **do**

$$\mathbf{P_i^{(k)}} = \underset{\substack{D_i\mathbf{P_i} \leq c_i \\ L_i \leq \mathbf{P_i} \leq U_i}}{\arg\sup}\ Flex_i(\mathbf{P_i}) +$$

1:
$$\sum_{r \in R_E(i)} \frac{-\rho}{4} \left(\frac{E_{ir}\mathbf{P_i} - p_{i,r}^{(k-1)} - b_{i,r}}{\rho} + y_{i,r}^{(k-1)} + y_{j,r}^{(k-1)} \right)^2$$

where, $j = Neigh_E(i,r)$

2: **for** $r \in R_E(i)$ **do**

$j = Neigh_E(i,r)$

$$y_{i,r}^{(k)} = \frac{E_{i,r}\mathbf{P_i^{(k)}} - b_{i,r} - p_{i,r}^{(k-1)} + \rho(y_{i,r}^{(k-1)} + y_{j,r}^{(k-1)})}{2\rho}$$

end for

3: Communicate updated y variables to neighbors selectively

4: **for** $r \in R_E(i)$ **do**

$j = Neigh_E(i,r)$

$p_{i,r}^{(k)} = p_{i,r}^{(k-1)} + \rho(y_{i,r}^{(k)} - y_{j,r}^{(k)})$

end for

end for

k = k + 1

until Termination Criterion

2.2.1 Empirical Analysis Of Algorithm 2

Experiment Design

To evaluate the effectiveness of our approach, we tested our algorithm on a set of reference problems called the *BDH_Problem_Instances*[2], due to [Boerkoel and Durfee, 2013]. This dataset contains problems with varying numbers of agents, specifically there are 50 MaSTN instances each for 2, 4, 8, 12 , 16 , and 20 agent problems. In each problem, each agent has 10 tasks to complete, which implies each agent STN consists of 20 time points. The number of inter-agent constraints specified for each problem instance is $50 \times (N-1)$, where N is the number of agents in the problem.

The objective we optimize is the sum of the flexibilities of all agents in our N agent decoupling problem. The flexibility measure specified to all agents was the one in [Boerkoel and Durfee, 2013] due to [Hunsberger, 2002], given that this was the assumed objective when this problem set was generated (We would expect to obtain comparable results to those presented below if the flexibility measure of [Wilson *et al.*, 2013] were substituted). To define flexibility, we adopt the convention used in section 1.3.1. Mathematically, the flexibility between 2 time points A_i and A_j belonging to agent A is defined equal to $p_{A_i,A_j} + p_{A_j,A_i}$. The flexibility for agent A's STN is given by

$$Flex_A(\mathbf{P_A}) = \sum_{\forall i,j \in \{1,\ldots,n_A\}, i>j} p_{A_i,A_j} + p_{A_j,A_i}$$

The penalty multiplier ρ in Algorithm 2 was set to 1. The feasibility tolerance ϵ_{Inf} was set to 0.1 based on the judgement that a maximum infeasibility with regard to any single DEC constraint of $\leq$ one tenth of a time tick would be insignificant from a practical perspective.

Our decoupling algorithm was implemented in C++ with Message Passing Interface (MPI) for inter-agent communication. For performing optimization in Algorithm 2, we used an Interior Point Solver IPOPT [Wächter and Biegler, 2006] with warm start. The primal version of each problem instance was also solved centrally using

[2]https://data.3tu.nl/repository/uuid:ce3ad00b-d905-4be3-8785-228ae19e371a

Number Agents	Average Iterations	Average % Dev	Average Time (in sec)
2	209	1.59×10^{-3}	43
4	408.1	4.59×10^{-3}	122.6
8	595.9	5.98×10^{-4}	324.9
12	666.7	4.17×10^{-4}	591.3
16	658.2	3.105×10^{-4}	749
20	663.5	3.497×10^{-4}	940.5

Table 2.1: Convergence performance of proposed algorithm

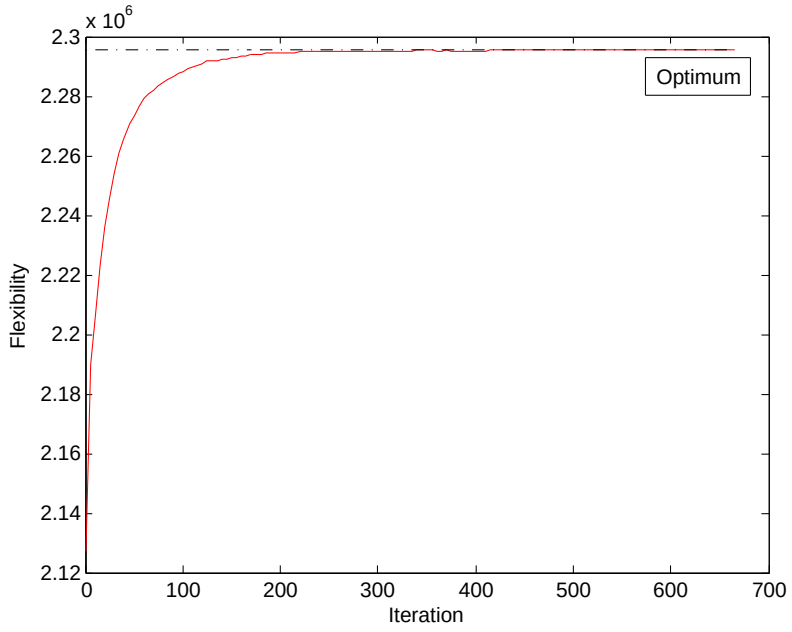

Figure 2.1: Solving progress on a 20 agent problem instance. Termination condition is satisfied at iteration 671.

IPOPT to provide optimal solutions for benchmarking. All experiments were carried out on an Intel 4 core i7-4790 processor at 3.6 GHz.

Results

The experimental results are summarized in Table 2.1. For each problem set size, we show

- the average number of iterations until the feasibility tolerance threshold is met,
- the average % deviation of the solution found from the optimal solution, where % deviation is defined as:

$$\%\, Dev = \frac{OptimalFlex - ComputedFlex}{OptimalFlex} \times 100$$

- average total computation time required.

As can be seen, optimizing performance is quite good; the percentage deviation from the optimal ranging from .0016 to .0006 of 1% depending on the size of the problem. On feasibility, the termination condition ensures that the maximum infeasibility with regard to any one DEC constraint is 0.1 time tick. However, we have also examined the Root Mean Squared (RMS) error of infeasibility of constraints (indicative of the magnitude of individual DEC constraint violations), defined as

$$RMS_{Inf} = \left(\sum_{l=1}^{L} \frac{(l^{th}\, DEC\, Violation\, Magnitude)^2}{L} \right)^{\frac{1}{2}}$$

where L is the number of DEC constraints. When the values obtained for instances of certain problem size are averaged, the resulting average RMS values range from 0.01 to 0.06 depending on problem size. Hence the average infeasibility associated with a particular DEC constraint is closer to one hundredth of a time tick. Since [Boerkoel and Durfee, 2013] have not to our knowledge published the solutions they obtained on these problems, it is not possbile to perform a direct performance comparison with their approach. However, we can note that their approach makes no claim of finding optimal solutions. The computation times shown in Table 2.1 were achieved using a single 4-core machine, which, for the 20-agents test problems, required multiple agents to share cores. In practice, where we expect agents to operate independently and have devoted cores, the computation time will significantly reduce. Moveover, as shown in the plot of a representative run in Fig.2.1, the algorithm quickly drops to the vicinity of the optimal and then oscillates around this neighborhood until the acceptable tolerance is achieved. If a compromise can be made on optimality such that a larger tolerance is acceptable, then the computation time will be substantially shorter.

Chapter 3

Some Useful Extensions

In this chapter, we first discuss ways for reducing the number of iterations for convergence. On the topic of reducing iterations for convergence, we explore 2 broad ideas that are commonly discussed in literature. The first idea augments terms to the objective function and optimizes this new sum instead. This approach provides a trade-off between time and quality of the decoupled solution. The second idea is to adapt the penalty parameter value at each iteration without compromising on optimality of the decoupled solution. The main challenge is in adapting existing methods mostly focused on centralized settings to decentralized settings. We conclude this chapter with a discussion on some Flexibility division ideas.

3.1 Augmented Objective Approach

In this section, we begin by briefly pointing out relevant literature that analyzes how the rate of convergence of ADMM is affected by the objective term in the optimization problem. Taking cue from those studies, we modify the objective in the primal problem defined in chapter 2 by augmenting terms to obtain a new problem whose optimum can be computed more efficiently in theory. Since only the objective is modified to obtain the modified problem, the optimal solution to the modified problem is automatically a feasible solution to the original primal problem. Hence, this section is dedicated to settings where computing a quick feasible decoupling is sufficient. As an added advantage from the choice of our augmenting (regularization) term, we will point out the existence of a relationship that bounds the difference in flexibility we obtain by solving the modified problem from the optimum to the original primal problem.

The convergence rate of ADMM is currently known as $\mathcal{O}(1/k)$, where k is the iteration number. In practice however, the algorithm has been observed to perform better. When the objective is strongly convex many accelerated schemes have been proposed that improves this rate to $\mathcal{O}(1/k^2)$. In [Goldstein *et al.*, 2014], the authors achieve this rate by combining ADMM with a predictor-corrector type acceleration step. For weakly convex problems their algorithm does not guarantee this rate. Recently [Suzuki, 2013; Ouyang *et al.*, 2013] proposed replacing the gradient descent step in ADMM with a stochastic step. Despite the decrease in computational load per iteration these methods offer in comparison to standard ADMM, they too suffer from the drawback of requiring a central agent. Most relevant to our work is that of [Shi *et al.*, 2014]. For consensus minimization problems having strongly convex objective, they show linear rate of convergence ($\mathcal{O}(1/c^k)$, $c > 1$) when optimized using standard ADMM with a specialized initialization of variables. Motivated by the approach of [Shi *et al.*, 2014], we modify our problem to satisfy the strong convexity condition of the objective. Notice in the Modified DC-ADMM primal problem that we defined in the previous chapter, it was to the dual formulation of the primal problem (also defined in previous chapter) that we applied ADMM. Hence if we were to obtain the linear rate of convergence as developed in [Shi *et al.*, 2014], we would need to make the objective in our dual formulation strongly convex. Next we run you through the steps in our attempt to make the objective in the dual formulation strongly convex.

We begin by modifying the objective in our primal formulation by adding logarithmic barrier as regularization terms. The logarithmic barrier terms are obtained from the LB and UB constraints specified. Skipping LB and UB constraints, we convert all other inequality constraints to equality constraints by adding slack variables into the equations[1]

Augmented Primal

$$\max_{\mathbf{P_1},\ldots,\mathbf{P_N}} \sum_{i=1}^{N} Flex_i(\mathbf{P_i}) + s\,\phi(\mathbf{P_i}, L_i, U_i)$$

s.t.

$$\sum_{i=1}^{N} E_i\mathbf{P_i} = b$$

$$D_i\mathbf{P_i} = c_i,\ i \in \mathcal{V}$$

where $s \in \mathbb{R}^+$ and $\phi(\cdot,\cdot,\cdot)$ is defined as follows. If $X = [x1,\ldots,x_n]^T, L = [l_1,\ldots,l_n]^T, U = [u_1,\ldots,u_n]^T$ where $X, L, U \in \mathbb{R}^{n\times 1}$ and $L \leq X \leq U$, then

$$\phi(X, L, U) = \sum_{i=1}^{n} log\left((u_i - x_i)(x_i - l_i)\right)$$

Notice that if we compare the Primal (defined in the previous chapter) and the Augmented primal problem, we see that LB and UB constraints have been used to generate the regularization function($\phi(\cdot,\cdot,\cdot)$) and effectively removed from the constraint set. We define the dual to Augmented primal problem next.

Dual To Augmented Primal

$$\min_{\mathbf{y}} \sum_{i=1}^{N} \underbrace{\sup_{D_i\mathbf{P_i}=c_i} Flex_i(\mathbf{P_i}) + s\,\phi(\mathbf{P_i}, L_i, U_i) - y^T\left(E_i\mathbf{P_i} - b_i\right)}_{G_i(y)}$$

Let $G(y) = \sum_{i=1}^{N} G_i(y)$, then $G(y)$ is the function we are attempting to make strongly convex to obtain the linear rate of convergence. Let $D = blkdiag(D_1,\ldots,D_N)$, where $blkdiag(\cdot)$ is the block diagonal operator and $H(\mathbf{P}) = \nabla^2_{\mathbf{P}}\left(\sum_{i=1}^{N} Flex_i(\mathbf{P_i}) + s\,\phi(\mathbf{P_i}, L_i, U_i)\right)$, then by Lemma 3.4 in [Necoara and Suykens, 2009] we have,

$$\nabla^2 G(y) = -E\left(H(\mathbf{P}^*(y))^{-1} - H(\mathbf{P}^*(y))^{-1}D^T(DH(\mathbf{P}^*(y))^{-1}D^T)^{-1}DH(\mathbf{P}^*(\mathbf{y}))^{-1}\right)E^T$$

$$\text{where } \mathbf{P}^*(y) = \underset{\substack{D_i\mathbf{P_i}=c_i \\ i\in\{1,\ldots,N\}}}{\arg\sup} \sum_{i=1}^{N} Flex_i(\mathbf{P_i}) + s\,\phi(\mathbf{P_i}, L_i, U_i) - y^T\left(E_i\mathbf{P_i} - b_i\right) \tag{3.1}$$

Since we assume E to be full row rank, $\nabla^2 G(y)$ is guaranteed to positive definite ($\succ 0$) iff

$$H(\mathbf{P}^*(y))^{-1} \prec H(\mathbf{P}^*(y))^{-1}D^T(DH(\mathbf{P}^*(y))^{-1}D^T)^{-1}DH(\mathbf{P}^*(\mathbf{y}))^{-1} \tag{3.2}$$

In general we are not guaranteed that Eqn (3.2) will hold, however $\nabla^2 G(y) \succeq 0$ is trivially true since G is convex. We also have the following-:

1. $\nabla G(y) = -\left(E\,\mathbf{P}^*(y) - b\right)$, where $\mathbf{P}^*(y)$ is unique since in Eqn (3.1) the objective is strictly concave.

[1]The slack variables themselves have been appended into the optimization variable vector. Computing the upper bounds for the slack variables for our decoupling problem is straightforward and is appended into the objective as a logarithmic regularization term.

2. Let $\bar{\mathbf{P}}_\mathbf{s}$ be the optimal solution to the Augmented primal problem and $\bar{\mathbf{P}}$ be the optimal solution to the primal problem (refer chapter 2). If the flexibility function is linear or concave quadratic for all agents, then according to Theorem 4.1 in [Necoara and Suykens, 2009] we have the following relation

$$0 \leq \left(\sum_{i\in\mathcal{V}} Flex_i(\bar{\mathbf{P}})\right) - \left(\sum_{i\in\mathcal{V}} Flex_i(\bar{\mathbf{P}}_\mathbf{s})\right) \leq s \times n, \text{ where } \bar{\mathbf{P}}, \bar{\mathbf{P}}_\mathbf{s} \in \mathbb{R}^{n\times 1} \tag{3.3}$$

The first observation from the above is that the dual function $G(y)$ is differentiable. Secondly for flexibility metrics of interest in this thesis, for any value $s > 0$ we set in the Augmented primal problem and solve, we can bound the maximum difference between the flexibility of the solution we obtain and the true optimal of the primal problem before actually computing either of the two.

3.1.1 Algorithm 3

Based on the discussions we had with augmenting barrier regression terms, we present Algorithm 3 which is inspired from the Interior point literature. Procedurally, at iteration k we set $s = s^{(k)} = \tau\, s^{(k-1)}$ (where $0 < \tau < 1$) in the Augmented primal problem objective and solve it distributively using Algorithm 1. We use the optimal solution we obtained in the previous iteration i.e $\{\bar{y}_{\cdot,\cdot}^{(k-1)}\}, \{\bar{t}_{\cdot,\cdot}^{(k-1)}\}, \{\bar{v}_{\cdot,\cdot}^{(k-1)}\}, \{\bar{w}_{\cdot,\cdot}^{(k-1)}\}$, PEN_THREASH_ITER =1 as the initialization to the variables in Algorithm 1 before solving in step 3. Note, Algorithm 3 requires as input the value of n, where n is the number of variables in the Augmented primal problem i.e. $n = \sum_{i=1}^{N} |\mathbf{P_i}|$, where $|\mathbf{P_i}|$ denotes the length of the vector $\mathbf{P_i}$. This implies, for Algorithm 3 to check for termination, it requires all agents to reveal their total number of edges in their minimal STNs. The output of the algorithm is the STN edge weights ($\mathbf{P_i}$ for agent i) for all agents in the decoupled MaSTN and the associated dual variables of the constraints. Algorithm 3 is expected to be useful in applications where quickly finding a feasible decoupling and having some sense of how it compares with the optimal is beneficial.

Algorithm 3

Given: $n, s_{\text{start}}, \rho, \epsilon_{\text{Inf}}, \tau, s = s_{\text{start}}, \{y_{\cdot,\cdot}^{(0)}\} = 0, \{t_{\cdot,\cdot}^{(0)}\} = 0, \{v_{\cdot,\cdot}^{(0)}\} = 0, \{w_{\cdot,\cdot}^{(0)}\} = 0, \epsilon_{\text{acc}}, k = 1, k_{\max}$ satisfying $\epsilon_{\text{acc}} < n\, s_{\text{start}}$, $0 < \tau < 1$, $s_{\text{start}} > 0$

Step 1. If $\epsilon_{\text{acc}} > n\, s^{(k-1)}$ or $k > k_{\max}$ go to Step 5.

Step 2. Initialize Algorithm.1 with $\rho_i = \rho, \forall i \in \mathcal{V}, \{y_{\cdot,\cdot}^{(k)}\} = \{\bar{y}_{\cdot,\cdot}^{(k-1)}\}, \{t_{\cdot,\cdot}^{(k)}\} = \{\bar{t}_{\cdot,\cdot}^{(k-1)}\}, \{v_{\cdot,\cdot}^{(k)}\} = \{\bar{v}_{\cdot,\cdot}^{(k-1)}\}, \{w_{\cdot,\cdot}^{(k)}\} = \{\bar{w}_{\cdot,\cdot}^{(k-1)}\}$, PEN_THRESH_ITER = 1. Set $s^{(k)} = \tau\, s^{(k-1)}$.

Step 3. Solve iteration k using Algorithm 1 to obtain optimal solution to the augmented primal problem with $s^{(k)}$ as the multiplier to the regularization function in the objective. **(Note the objective contains the regularization term $\phi(\cdot)$ in addition to the flexibility terms.)**

Step 4. Set $k = k + 1$. Go to Step 1.

Step 5. Output $\bar{\mathbf{P}}_\mathbf{i}^{(k-1)}$ where $i \in \mathcal{V}, \{\bar{y}_{\cdot,\cdot}^{(k-1)}\}, \{\bar{t}_{\cdot,\cdot}^{(k-1)}\}, \{\bar{v}_{\cdot,\cdot}^{(k-1)}\}, \{\bar{w}_{\cdot,\cdot}^{(k-1)}\}$.

Empirical Performance Of Algorithm 3

We evaluated the efficacy of Algorithm 3 for just finding quick feasible solutions on problems with varying sizes. We initialized $s_{\text{start}} = 1, k_{\max} = 1, \rho = 1, \epsilon_{\text{Inf}} = 0.1$ in Algorithm 3, hence $\tau, \epsilon_{\text{acc}}$ became inconsequential since only 1 iteration in Algorithm 3 is allowed to run. The motivation to test in this manner was simply to verify that improving the curvature of the objective function results in fewer number of iterations for convergence by ADMM. The average number of iterations in Step 3 of Algorithm 3 and its relative decrease when compared

against Algorithm 2 on the dataset introduced in chapter 2 is provided in Table 3.1. At this point, I wish to remind the reader that since Algorithms 2 and 3 have different objectives, comparison in number of iterations does not indicate their relative convergence performance.

Number Agents	Average Iterations	%Decrease In Iterations
4	193.33	50.77
8	303.5	49.70
12	332.8	49.00
16	334.5	50.73
20	356.9	46.74

Table 3.1: Performance Of Algorithm 3

In another experiment we varied s_{start} to observe how it affects the number of iterations for convergence. We tested them on the 8-agent problems with s_{start} = 1, 10 and 100. The trend observed from Table 3.2 is that higher the value of s_{start}, fewer is the number of iterations for convergence. This suggests that increasing the curvature of the function results in fewer the number of iterations for convergence.

	s_{start} = 1	s_{start} = 10	s_{start} = 100
Average Iterations	303.5	237.5	140.9
% Decrease in iterations for solving aug primal over primal	49.70	60.25	76.55

Table 3.2: Number of iterations for convergence with varying s_{start}

Usage Guidelines For Algorithm 3

We recommend using Algorithm 3 only when finding a feasible decoupling is more important than finding the optimal decoupling i.e. solution to primal problem defined in chapter 2 where the objective is devoid of regularization terms. Using a high value for s_{start} empirically gave us the fastest results on the dataset described in chapter 2. Having obtained a feasible solution, by examining its flexibility value and maximum deviation from the optimal (Eqn (3.3)), we can progressively run more number of iterations in Algorithm 3 by increasing $k_{\max}$ until we have obtained a satisfactory solution.

3.2 Variable Penalty Method

Many schemes have been proposed that accelerate ADMM by altering the penalty parameter value at each iteration. To explain their general philosophy, consider the following optimization problem and its ADMM updates.

$$\begin{aligned} \min_{x,z} \quad & f(x) + g(z) \\ \text{s.t.} \quad & Ax + Bz = c \end{aligned}$$

$$
\begin{aligned}
x^{(k)} &= \arg\min_{x} f(x) + x^T A^T y^{(k-1)} + \frac{\rho^{(k)}}{2}||Ax + Bz^{(k-1)} - c||_2^2 \\
z^{(k)} &= \arg\min_{z} g(z) + z^T B^T y^{(k-1)} + \frac{\rho^{(k)}}{2}||Ax^{(k)} + Bz - c||_2^2 \\
y^{(k)} &= y^{(k-1)} + \rho^{(k)}(Ax^{(k)} + Bz^{(k)} - c)
\end{aligned}
$$

The primal residue and dual residuals at iteration k are defined as follows.

$$
\begin{aligned}
r^{(k)} &= Ax^{(k)} + Bz^{(k)} - c && (3.4)\\
d^{(k)} &= \rho^{(k)} A^T B(z^{(k)} - z^{(k-1)}) && (3.5)
\end{aligned}
$$

The philosophy behind variable penalty approaches is to examine the relative ratios of the primal and dual residues and adjust the penalty parameter accordingly. A widely cited scheme in the literature is the following, where η is a tunable parameter.

$$
\rho^{(k+1)} = \begin{cases} 2\rho^{(k)}, & \text{if } ||r^{(k)}||_2 > \eta||d^{(k)}||_2. \\ \frac{\rho^{(k)}}{2}, & \text{if } ||d^{(k)}||_2 > \eta||r^{(k)}||_2 \\ \rho^{(k)}, & \text{otherwise} \end{cases} \tag{3.6}
$$

The problem in applying this scheme to distributed settings is that the computation of residuals require us to know the global structure of the constraints. More importantly a coordinating\central node that accumulates the residuals from all agents is required to determine the primal dual residue ratio. The authors in [Song *et al.*, 2015] proposed an adaptation of the previous scheme to distributed settings called ADMM-VP method. They suggested each agent maintains its own penalty parameter (ρ_i for agent i) and computes the primal and dual residuals by examining its variables and those of its neighbours. To explain their definitions for the residuals we use the following the consensus optimization problem (equivalent to the Modified DC-ADMM primal problem defined in chapter 2).

$$
\begin{aligned}
&\min_{\substack{\theta_1,\ldots,\theta_N \\ \{t_{\cdot,\cdot}\}}} \sum_{i=1}^{N} f(\theta_i) \\
&\text{s.t.} \qquad \theta_i = t_{ij},\ \theta_j = t_{ij}, \text{ where } j \in Neigh(i),\ i \in \mathcal{V}
\end{aligned}
$$

If one were to apply the standard definitions of the residuals as in Eqns (3.4), (3.5), we would obtain the following quantities,

$$
\begin{aligned}
||r^{(k)}||_2^2 &= \sum_{i=1}^{N} ||\theta_i^{(k)} - \bar{\theta}^{(k)}||_2^2, \text{ where } \bar{\theta}^{(k)} = \sum_{j=1}^{N} \frac{\theta_j^{(k)}}{N} && (3.7)\\
||d^{(k)}||_2^2 &= \rho^2 ||\bar{\theta}^{(k)} - \bar{\theta}^{(k-1)}||_2^2 && (3.8)
\end{aligned}
$$

The distributed adaptation of the definition for the residuals as proposed by [Song *et al.*, 2015] for agent i is as follows-:

$$
\begin{aligned}
||r_i^{(k)}||_2^2 &= ||\theta_i^{(k)} - \bar{\theta_i}^{(k)}||_2^2, \text{ where } \bar{\theta_i}^{(k)} = \sum_{j \in Neigh(i)} \frac{\theta_j^{(k)}}{|Neigh(i)|} && (3.9)\\
||d_i^{(k)}||_2^2 &= \rho_i^2 ||\bar{\theta_i}^{(k)} - \bar{\theta_i}^{(k-1)}||_2^2 && (3.10)
\end{aligned}
$$

According to [Song *et al.*, 2015], each agent would then simply look at its primal and dual residuals and accordingly update its penalty parameter similar to the scheme we showed in 3.6. In our decoupling problem however, we do not maintain consensus over all variables due to the privacy requirement, hence applying the

same definition for residuals as in Eqns (3.9) and (3.10) is not feasible. If we were to apply the standard definitions of the primal and dual residuals as defined in Eqns (3.4) and (3.5) to the Modified DC-ADMM Primal problem (defined in chapter 2), then we would have the following expressions for the residuals.

$$||r^{(k)}||_2^2 = \sum_{i=1}^{N} \sum_{r \in R_E(i)} (y_{i,r}^{(k)} - t_{i,r}^{(k)})^2 + (y_{j,r}^{(k)} - t_{i,r}^{(k)})^2, \text{ where } j = Neigh_E(i,r)$$

$$||d^{(k)}||_2^2 = \sum_{i=1}^{N} \sum_{r \in R_E(i)} (\rho_i^{(k)} t_{i,r}^{(k)} + \rho_j^{(k)} t_{j,r}^{(k)} - \rho_i^{(k)} t_{i,r}^{(k-1)} - \rho_j^{(k)} t_{j,r}^{(k-1)})^2$$

Intuitively from the method of [Song *et al.*, 2015], it seems that ADMM-VP is similar in spirit to the method of splitting the residual terms in Eqn (3.4) and (3.5) according to each agent's contribution to it. Taking cue from [Song *et al.*, 2015], we propose the following definition for the residuals for agent i.

$$||r_i^{(k)}||_2^2 = \sum_{r \in R_E(i)} (y_{i,r}^{(k)} - t_{i,r}^{(k)})^2 + (y_{j,r}^{(k)} - t_{i,r}^{(k)})^2, \text{ where } j = Neigh_E(i,r) \tag{3.11}$$

$$||d_i^{(k)}||_2^2 = (\rho_i^{(k)})^2 \sum_{r \in R_E(i)} (t_{i,r}^{(k)} - t_{i,r}^{(k-1)})^2, \text{ where } j = Neigh_E(i,r) \tag{3.12}$$

In defining the dual residual, we simply skipped A^T term from the standard definition of dual residual in Eqn (3.5). Including the A^T term would have resulted in a different heuristic having slightly greater communication requirements per iteration, whose convergence behavior has not been explored in this thesis. Using the primal and dual residual definitions in Eqns (3.11) and (3.12), we analyse Algorithm 1 for solving the Modified DC-ADMM Primal problem with the following definition for the Update$(\cdot,\cdot,\cdot)$ function (used in Algorithm 1 step 7).

$$\rho_i^{(k)} = \text{Update}(\rho_i^{(k-1)}, \{y_{i,\cdot}^{(k)}\}, \{t_{i,\cdot}^{(k-1)}\}) = \begin{cases} 2\rho_i^{(k-1)}, & \text{if } ||r_i^{(k)}||_2 > \eta ||d_i^{(k)}||_2. \\ \frac{\rho_i^{(k-1)}}{2}, & \text{if } ||d_i^{(k)}||_2 > \eta ||r_i^{(k)}||_2 \\ \rho_i^{(k-1)}, & \text{otherwise} \end{cases} \tag{3.13}$$

For convenience, we call Algorithm 1 with the above Update routine as Algorithm 4.

3.2.1 Empirical Performance Of Algorithm 4

We set PEN_THRESH_ITER = 100, $\rho_i^{(0)} = 1, i \in \mathcal{V}, \eta = 3$ and $\rho = 1$. The empirical performance of Algorithm 4 when measured relative to Algorithm 2 on the dataset introduced in chapter 2 is provided below. As the optimal we obtain from Algorithm 4 would be identical to that of Algorithm 2, comparison between number of iterations for convergence is an exact comparison between their convergence performance.

Number Agents	Average Iterations	%Decrease In Iterations
4	362.83	9.835
8	542.1	11.26
12	589.6	9.84
16	619.7	10.06
20	646.82	3.68

Table 3.3: Performance Of Algorithm 3

Our experience with Algorithm 4 is that the number of iterations decreases by 10% on average, however this varies significantly with η. We tried η=10, 5 and 3 and found that η=3 gave the better results for our dataset. However, we remark that the reduction in number of iterations for convergence is not very significant.

3.3 Flexibility Division

In our discussions thus far, the social welfare function we maximized was the sum of the flexibilities of each agent's STN in the decoupled MaSTN. However from a social welfare division perspective, this may not lead to fair allocations when all agents follow the same definition for flexibility($Flex(\cdot)$). For instance, suppose we have a 2 agent MaSTN where in one case we allocate F units of flexibility to one agent and 0 units of flexibility to the other agent, and the second case where both agents get $\frac{F}{2}$ units of flexibility. Further, suppose both are pareto optimal solutions, then from a fair division perspective one would intuitively consider the latter case to be the more fair flexibility allocation solution, while our current social welfare definition which simply maximizes the sum does not distinguish between these 2 cases. The fair solution would have been achieved if we maximized the product of each agent's STN flexibility instead.

$$\max_{\mathbf{P_1},\ldots,\mathbf{P_N}} \prod_{i=1}^{N} Flex(\mathbf{P_i}) \tag{3.14}$$

Using the framework already presented, we can obtain the solution to Eqn (3.14) as long as the flexibility metric is concave and > 0 over the domain. We could achieve it by maximizing the objective in Eqn (3.15). The function in Eqn (3.15) is concave since composition of a monotonically increasing concave function with a concave function results in a concave function.

$$\sum_{i=1}^{N} log(Flex(\mathbf{P_i})) \tag{3.15}$$

Another interesting flexibility division mechanism that may be useful when all agents follow the same definition for flexibility is max-min allocation i.e. we try to maximize the flexibility for the agent receiving the least flexibility in the decoupled MaSTN. For concave flexibility metrics, max-min solution can be obtained in the existing framework by simply introducing consensus constraints over flexibility for each agent. Below we show the max-min primal problem.

Max-Min Primal

$$\max_{\substack{\mathbf{P_1},\ldots,\mathbf{P_N} \\ s_1,\ldots,s_N \\ \{q_{\cdot,\cdot}\}}} \sum_{i=1}^{N} s_i$$

s.t.

$$Flex(\mathbf{P_i}) \geq s_i, i \in \mathcal{V}$$
$$s_i = q_{i,j}\,, s_j = q_{i,j} \text{ where } j \in Neigh(i),\ i \in \mathcal{V}$$
$$\sum_{i=1}^{N} E_i \mathbf{P_i} = b$$
$$D_i \mathbf{P_i} \leq c_i,\ i \in \mathcal{V}$$
$$L_i \leq \mathbf{P_i} \leq U_i,\ i \in \mathcal{V}$$

Notice that the constraint $Flex(\mathbf{P_i}) \geq s_i$ is similar to the TRI constraint ($D_i P_i \leq c_i$) since all the variables in the constraint belong to only agent i, while the constraint $s_j = q_{i,j}$ is similar to any DEC constraint ($\sum_{i=1}^{N} E_i P_i = b$) where variables belonging only to 2 agents are present in each constraint. So the max-min optimization problem can be readily solved using the privacy preserving distributed algorithm framework we have developed.

Chapter 4

Conclusion

4.1 Summary And Discussions Of Contributions

In this thesis, we have presented a new distributed algorithm for decoupling inter-dependent multi-agent simple temporal networks (MaSTNs). We formulate the problem as a convex optimization problem and adapt a recently proposed consensus ADMM method to specify a distributed, iterative procedure that is guaranteed in the limit to converge to the optimum, given a concave flexibility objective. The procedure works by tolerating some amount of infeasibility across agent plans/schedules, and iteratively minimizing the magnitude of infeasibility (the dual objective) to some acceptable threshold. Experimental analysis of the procedure on a set of reference MaSTN decoupling problems has shown strong performance. Over this problem set, the solutions found are within .0016 - .0005 of 1% of the optimal decoupling, and convergence is observed to require a relatively small number of iterations.

Our ADMM-based decoupling algorithms improves on prior work in decoupling of MaSTNs in several ways. Most importantly, it provides for the first time a decoupling algorithm that addresses agent privacy concerns, requiring sharing of only the minimal amount of information necessary to produce an optimal decoupling. It is also the first distributed decoupling algorithm offering an optimality guarantee. It accommodates a range of decoupling objectives, making the approach more broadly applicable to more self-interested settings where agents may have different objectives. For applications where computing a quick feasible solution is sufficient we proposed Algorithm 3. Algorithm 3 also provides an efficient mechanism for the agents to move from one feasible decoupling to another with assurance that the social welfare is non-decreasing. In Algorithm 4, we presented a distributed algorithm where the penalty value is local to each agent and proposed a heuristic for modifying it at each iteration. The results showed slight promise empirically. Finally, we discussed a few popular social welfare division mechanisms and showed how they can be readily solved with distributed algorithms presented in this thesis simultaneously preserving privacy.

4.2 Future Work

One potential limitation of our decoupling approach, given its reliance on a matrix formulation, is its scalability to large agent plans/schedules. The issue with non scalability is due to that fact that the number of TRI constraints we impose is cubic w.r.t the number of time points present in an agent's STN. An important point not made explicit in our formulation is that our objective metric requires all edge information, which prohibits the use of Partial Path Consistency (PPC) [Planken *et al.*, 2008] as a way of reducing the number of TRI constraints. One thrust of our current work aims at investigating other approaches to reducing model size. Another future direction is on detecting infeasibility of the MaSTN decoupling problem efficiently. Existing literature has focussed on characterizing the primal and dual iterates generated during optimization [Raghunathan and Di Cairano, 2014] or through solving the primal-dual equations pair obtained from the KKT conditions of the

optimization problem [O'Donoghue *et al.*, 2013]. Neither of these existing techniques are directly applicable to distributed settings with privacy requirements such as ours.

Appendix

A Derivation For Algorithm 1

The steps shown below is identical to the proof in [Chang, 2014]. The primal problem is shown below, all DEC constraints have been converted to equality constraints by the addition of slack variables.

$$\text{Primal}$$

$$\max_{\mathbf{P_1},\ldots,\mathbf{P_N}} \sum_{i=1}^{N} Flex_i(\mathbf{P_i})$$

$$\text{s.t.}$$

$$\sum_{i=1}^{N} E_i\mathbf{P_i} = b$$

$$D_i\mathbf{P_i} \leq c_i,\ i \in \mathcal{V}$$

$$L_i \leq \mathbf{P_i} \leq U_i,\ i \in \mathcal{V}$$

Let $\{\Omega_i : D_i\mathbf{P_i} \leq c_i,\ L_i \leq \mathbf{P_i} \leq U_i\}$, then the consensus dual to the above problem is as follows-:

$$\text{Consensus Dual}$$

$$\min_{\substack{\{y_{\cdot}\}\\ \{t_{\cdot}\}}} \underbrace{\sup_{\Omega_1,\ldots,\Omega_N} \sum_{i=1}^{N} \left(Flex_i(\mathbf{P_i}) + \sum_{r\in R_E(i)} -y_{i,r}\left(E_{i,r}\mathbf{P_i} - b_{i,r}\right) \right)}_{\phi(\{y_{\cdot,\cdot}\},\{t_{\cdot,\cdot}\})}$$

$$\text{s.t.}$$

$$y_{i,r} = t_{i,r},\ y_{j,r} = t_{i,r},\ \text{where}\ j = Neigh(i,r),\ \forall (i,r) \in \mathcal{E}$$

$\phi(\{y_{\cdot,\cdot}\},\{t_{\cdot,\cdot}\})$ is a convex function since it is the pointwise maximum of a family of affine functions. $\{t_{\cdot,\cdot}\}$ denotes the set of all consensus variables for all agents $\{\{t_{i,r}\}_{\forall r\in R_E(i)}\}_{i=1}^{N}$ and $\{y_{\cdot,\cdot}\}$ denotes the set of all lagrange dual variables for the primal problem. We solve the consensus dual problem using ADMM. ADMM concerns the following augmented Lagrangian function

$$\mathcal{L}_\rho(y,t,v,w) \triangleq \phi(y,t) + \sum_{i=1}^{N}\sum_{r\in R_E(i)} v_{i,r}(y_{i,r} - t_{i,r}) + \frac{\rho_i}{2}(y_{i,r} - t_{i,r})^2 + w_{i,r}(y_{j,r} - t_{i,r}) + \frac{\rho_i}{2}(y_{j,r} - t_{i,r})^2$$

For the sake of brevity of notation we denote $\{y_{..}\}$ as y, and follow a similar convention for t, v, w. The ADMM updates for the problem at the k^{th} iteration are as follows-:

$$\begin{aligned} y^{(k)} &= \arg\min_{y} \mathcal{L}_\rho(y, t^{(k-1)}, v^{(k-1)}, w^{(k-1)}) && (1)\\ t^{(k)} &= \arg\min_{t} \mathcal{L}_\rho(y^{(k)}, t, v^{(k-1)}, w^{(k-1)}) && (2)\\ v_{i,r}^{(k)} &= v_{i,r}^{(k-1)} + \rho_i^{(k-1)}(y_{i,r}^{(k)} - t_{i,r}^{(k)}), \forall (i,r) \in \mathcal{E} \\ w_{i,r}^{(k)} &= w_{i,r}^{(k-1)} + \rho_i^{(k-1)}(y_{j,r}^{(k)} - t_{i,r}^{(k)}), \forall (i,r) \in \mathcal{E} \end{aligned}$$

We begin by solving Eqn 1, notice that it involves solving a min-max problem for a given t, v, w, let

$$y^{(k)} = \arg\min_{y} \sup_{\Omega_1,\ldots,\Omega_N} \hat{\mathcal{L}}_\rho(\mathbf{P}, y, t^{(k-1)}, v^{(k-1)}, w^{(k-1)}) \text{ where,} \quad (3)$$

$$\begin{aligned} \hat{\mathcal{L}}_\rho(\mathbf{P}, y, t^{(k-1)}, v^{(k-1)}, w^{(k-1)}) \triangleq & \left(\sum_{i=1}^{N} Flex_i(\mathbf{P_i}) + \sum_{r \in R_E(i)} -y_{i,r}\left(E_{i,r}\mathbf{P_i} - b_{i,r}\right) \right) + \\ & \sum_{i=1}^{N} \sum_{r \in R_E(i)} v_{i,r}(y_{i,r} - t_{i,r}^{(k-1)}) + \frac{\rho_i}{2}(y_{i,r} - t_{i,r}^{(k-1)})^2 + \\ & w_{i,r}^{(k-1)}(y_{j,r} - t_{i,r}^{(k-1)}) + \frac{\rho_i}{2}(y_{j,r} - t_{i,r}^{(k-1)})^2, \text{ where } j = Neigh(i,r) \end{aligned}$$

$\hat{\mathcal{L}}_\rho(\mathbf{P}, y, t^{(k-1)}, v^{(k-1)}, w^{(k-1)})$ is strictly convex w.r.t y and concave w.r.t $\mathbf{P}$, hence by applying Sion's Minimax theorem due to [Sion and others, 1958], the min and max can be interchanged in Eqn (3) yielding the same solution as the original problem i.e.

$$\mathbf{P}^{(k)} \triangleq \arg\max_{\Omega_1,\ldots,\Omega_N} \min_{y} \hat{\mathcal{L}}_\rho(\mathbf{P}, y, t^{(k-1)}, v^{(k-1)}, w^{(k-1)}) \quad (4)$$

For solving Eqn (4), we obtain a closed form solution for inner minimization for a given $\mathbf{P}$ and use that solution to obtain an expression for solving the outer maximization. Let $y^*(P)$ be the solution to the inner minimization for a given $\mathbf{P}$. By setting $\nabla_{y^*} \hat{\mathcal{L}}_\rho(\mathbf{P}, y, t^{(k-1)}, v^{(k-1)}, w^{(k-1)}) = 0$, it can be shown that,

$$y_{i,r}^*(\mathbf{P}) = \frac{E_{i,r}\mathbf{P_i} - b_{i,r} - v_{i,r}^{(k-1)} - w_{j,r}^{(k-1)}}{\rho_i + \rho_j} + \frac{\rho_i t_{i,r}^{(k-1)} + \rho_j t_{j,r}^{(k-1)}}{\rho_i + \rho_j}, \text{ where } j = Neigh(i,r) \quad (5)$$

$$\mathbf{P^{(k)}} = \arg\sup_{\Omega_1,\ldots,\Omega_N} \hat{\mathcal{L}}_\rho(\mathbf{P}, y^*(\mathbf{P}), t^{(k-1)}, v^{(k-1)}, w^{(k-1)})$$

$$\begin{aligned} &= \arg\sup_{\Omega_1,\ldots,\Omega_N} \sum_{i=1}^{N} Flex_i(\mathbf{P_i}) + \sum_{r \in R_E(i)} \frac{-1}{2\left(\rho_i^{(k-1)} + \rho_j^{(k-1)}\right)}(E_{i,r}\mathbf{P_i})^2 + \\ &\qquad E_{i,r}\mathbf{P_i}\left(\frac{v_{i,r}^{(k-1)} + w_{j,r}^{(k-1)} + b_{i,r}}{\rho_i^{(k-1)} + \rho_j^{(k-1)}} - \frac{\rho_i^{(k-1)} t_{i,r}^{(k-1)} + \rho_j^{(k-1)} t_{j,r}^{(k-1)}}{\rho_i^{(k-1)} + \rho_j^{(k-1)}} \right) \\ &= \arg\sup_{\Omega_1,\ldots,\Omega_N} \sum_{i=1}^{N} Flex_i(\mathbf{P_i}) + \\ &\sum_{r \in R_E(i)} \frac{-1\left(E_{i,r}\mathbf{P_i} - v_{i,r}^{(k-1)} - w_{j,r}^{(k-1)} - b_{i,r} + \rho_i^{(k-1)} t_{i,r}^{(k-1)} + \rho_j^{(k-1)} t_{j,r}^{(k-1)}\right)^2}{2\left(\rho_i^{(k-1)} + \rho_j^{(k-1)}\right)} \\ &\text{where, } j = Neigh(i,r) \end{aligned}$$

Using the optimal value for $\mathbf{P^{(k)}}$, we can substitute into Eqn (5) to obtain the following

$$y_{i,r}^{(k)} = \frac{E_{i,r}\mathbf{P_i}^{(k)} - b_{i,r} - v_{i,r}^{(k-1)} - w_{j,r}^{(k-1)}}{\rho_i^{(k-1)} + \rho_j^{(k-1)}} + \frac{\rho_i^{(k-1)} t_{i,r}^{(k-1)} + \rho_j^{(k-1)} t_{j,r}^{(k-1)}}{\rho_i^{(k-1)} + \rho_j^{(k-1)}}, \forall (i,r) \in \mathcal{E}$$
$$\text{where, } j = Neigh(i,r)$$

Solving for Eqn (2) by setting $\nabla_{t^{(k)}} \mathcal{L}_\rho(y^{(k)}, t, v^{(k-1)}, w^{(k-1)}) = 0$, we obtain

$$t_{i,r}^{(k)} = \frac{v_{i,r}^{(k-1)} + w_{i,r}^{(k-1)}}{2\rho_i^{(k-1)}} + \frac{y_{i,r}^{(k)} + y_{j,r}^{(k)}}{2}, \text{ where } j = Neigh(i,r), \forall (i,r) \in \mathcal{E}$$

B Special Case, Derivation of Algorithm 2

If $v_{i,r}^{(0)}, w_{i,r}^{(0)} = 0$, $\forall (i,r) \in \mathcal{E}$, and further let $\rho_i^{(k)} = \rho, \forall k$ and $i = 1, \ldots, N$ then it can be shown that

$$v_{i,r}^{(k)} + w_{i,r}^{(k)} = 0, \forall (i,r) \in \mathcal{E} \text{ and } \forall\, k \tag{6}$$

$$t_{i,r}^{(k)} = \frac{y_{i,r}^{(k)} + y_{j,r}^{(k)}}{2}, \forall k, \forall (i,r) \in \mathcal{E} \text{ where } j = Neigh(i,r) \tag{7}$$

Further let,

$$p_{i,r}^{(k)} = v_{i,r}^{(k)} + w_{j,r}^{(k)}, \text{ where } j = Neigh(i,r) \tag{8}$$

Substituting Eqns (6), (7), (8) in updates for $\mathbf{P^{(k)}}, y^{(k)}$ we get

$$\mathbf{P_i^{(k)}} = \arg\sup_{\Omega_i} Flex_i(\mathbf{P_i}) + \sum_{r \in R_E(i)} \frac{-\rho}{4} \left(\frac{E_{ir}\mathbf{P_i} - p_{i,r}^{(k-1)} - b_{i,r}}{\rho} + y_{i,r}^{(k-1)} + y_{j,r}^{(k-1)} \right)^2$$
$$\text{where, } j = Neigh(i,r)$$

$$y_{i,r}^{(k)} = \frac{E_{i,r}\mathbf{P_i^{(k)}} - b_{i,r} - p_{i,r}^{(k-1)} + \rho(y_{i,r}^{(k-1)} + y_{j,r}^{(k-1)})}{2\rho} \text{ where, } j = Neigh(i,r)$$

Since in the above equations, v, w was effectively replaced by p, the update for p using Eqn (8) is

$$\begin{aligned} p_{i,r}^{(k)} &= v_{i,r}^{(k)} + w_{j,r}^{(k)} \\ &= v_{i,r}^{(k-1)} + \rho(y_{i,r}^{(k)} - t_{i,r}^{(k)}) + w_{j,r}^{(k-1)} + \rho(y_{i,r}^{(k)} - t_{j,r}^{(k)}) \\ &= p_{i,r}^{(k-1)} + \rho(2y_{i,r}^{(k)} - y_{i,r}^{(k)} - y_{j,r}^{(k)}) \\ &= p_{i,r}^{(k-1)} + \rho(y_{i,r}^{(k)} - y_{j,r}^{(k)}) \end{aligned}$$

Acknowledgements

I would like to thank my advisers Prof. Stephen Smith and Dr. Zachary Rubinstein for their invaluable guidance and support since I arrived at the Robotics Institute. Thank you Prof. Steve for your guidance and patience during the many months I took for deciding on a research problem. I truly appreciate the flexibility and freedom you provide your graduate students. I thank my other thesis committee members Prof. Soummya Kar and Dr. Matt Wytock for being very helpful and pointing me to relevant literature for any questions I had. I would like to acknowledge my office mates Achal Arvind, Allen Hawkes and rest of my colleagues at ICL lab. I feel very lucky to have become a part of this group. I would like to thank DARPA and CMU Robotics Institute for funding my research. Finally, I wish to thank my family members for passionately supporting my academic endeavors, I could not have done it without them.

Bibliography

James C. Boerkoel and Edmund H. Durfee. Distributed algorithms for solving the multiagent temporal decoupling problem. In *Proceedings 10th International Conference on Autonomous Agents and Multiagent Systems*, May 2011.

James C. Boerkoel and Edmund H. Durfee. Distributed reasoning for multiagent simple temporal problems. *Journal of Artificial Intelligence Research*, 47:95–156, 2013.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

Ting-Hau Chang, Mingyi Hong, and Xiongfei Wang. Multi-agent distributed optimization via inexact consensus admm. *Signal Processing, IEEE Transactions on*, 63(2):482–497, 2015.

Tsung-Hui Chang. A proximal dual consensus admm method for multi-agent constrained optimization. *arXiv preprint arXiv:1409.3307*, 2014.

Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, May 1991.

Tom Goldstein, Brendan O'Donoghue, Simon Setzer, and Richard Baraniuk. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences*, 7(3):1588–1623, 2014.

Luke Hunsberger. Algorithms for a temporal decoupling problem in multi-agent planning. In *18th National Conf. on Artificial intelligence*, pages 468–475, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

Philipe Laborie and Malik Ghallab. Planning with sharable resource constraints. In *Proceedings 1995 International Joint Conference on Artifical Intelligence*, pages 1643–1649, 1995.

Jayanth Krishna Mogali, Stephen F Smith, and Zachary B Rubinstein. Distributed decoupling of multiagent simple temporal problems. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, July 2016.

I Necoara and JAK Suykens. Interior-point lagrangian decomposition method for separable convex optimization. *Journal of Optimization Theory and Applications*, 143(3):567–588, 2009.

Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Operator splitting for conic optimization via homogeneous self-dual embedding. *arXiv preprint arXiv:1312.3039*, 2013.

Hua Ouyang, Niao He, Long Tran, and Alexander Gray. Stochastic alternating direction method of multipliers. In *Proceedings of the 30th International Conference on Machine Learning*, pages 80–88, 2013.

Léon Planken, Mathijs De Weerdt, and Roman van der Krogt. P3c: A new algorithm for the simple temporal problem. In *Proceedings Eighteenth International Conference on Automated Planning and Scheduling*, pages 256–263, Sydney, Australia, 2008.

Léon R Planken, Mathijs M De Weerdt, and Cees Witteveen. Optimal temporal decoupling in multiagent systems. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 789–796. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

Nicola Policella, Amedeo Cesta, Angelo Oddi, and Stephen F. Smtih. Solve-and-robustify: Synthesizing partial order schedules by chaining. *Journal of Scheduling*, 12(3), 2009.

Arvind U Raghunathan and Stefano Di Cairano. Infeasibility detection in alternating direction method of multipliers for convex quadratic programs. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 5819–5824. IEEE, 2014.

Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin. On the linear convergence of the admm in decentralized consensus optimization. *Signal Processing, IEEE Transactions on*, 62(7):1750–1761, 2014.

Maurice Sion et al. On general minimax theorems. *Pacific Journal of Mathematics*, 8:171–176, 1958.

Stephen F. Smith and C. Cheng. Slack-based heuristics for constraint satisfaction scheduling. In *Proceedings 11th National Conference on Artificial Intelligence*, July 1993.

David E. Smith, Jeremy Frank, and Ari K. Jonsson. Bridging the gap between planning and scheduling. *Knowledge Engineering Review*, 15(1):47–83, Mar 2000.

Stephen F. Smith, Anthony T. Gallagher, Terry Lyle Zimmerman, Laura Barbulescu, and Zack Rubinstein. Distributed management of flexible times schedules. In *Proceedings 6th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2007.

Changkyu Song, Sejong Yoon, and Vladimir Pavlovic. Fast admm algorithm for distributed optimization with adaptive penalty. *arXiv preprint arXiv:1506.08928*, 2015.

Taiji Suzuki. Dual averaging and proximal gradient descent for online alternating direction multiplier method. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 392–400, 2013.

Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

Michel Wilson, Tomas Klos, Cees Witteveen, and Bob Huisman. Flexibility and decoupling in the simple temporal problem. In F. Rossi, editor, *Proceedings International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2422 – 2428. AAAI Press, Menlo Park, CA, 2013.

Cees Witteveen, Michel Wilson, and Tomas Klos. Optimal decoupling in linear constraint systems. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14)*, pages 2381–2387. AAAI Press, Menlo Park, CA, August 2014.