# Group Decision Making and Temporal Reasoning

A thesis presented
by

**Luke Hunsberger**

to
The Division of Engineering and Applied Sciences
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of
Computer Science

Harvard University
Cambridge, Massachusetts

June, 2002

To the memory of my father
Isaac Moyer Hunsberger

# Abstract

The more capable and autonomous computer systems become, the more important it is for them to be able to act collaboratively, whether in groups consisting solely of other computers or in heterogeneous groups of computers and people. To act collaboratively requires that systems have effective group decision-making capabilities. This thesis makes four important contributions to the design of group decision-making mechanisms and algorithms for deploying them in collaborative, multi-agent systems. First, it provides an abstract framework for the specification of group decision-making mechanisms that computer agents can use to coordinate their planning activity when collaborating with other agents. Second, it specifies a combinatorial auction-based mechanism that computer agents can use to help them decide, both individually and collectively, whether to engage in a collaborative activity. Third, it extends the theory of Simple Temporal Networks by providing a rigorous theoretical analysis of an important family of temporal reasoning problems. Fourth, it provides sound, complete and polynomial-time algorithms for solving those temporal reasoning problems and specifies the use of such algorithms by agents participating in the auction-based mechanism.

# Contents

# List of Figures

# Acknowledgments

I would like to thank my advisor, Barbara J. Grosz, for her dedication to research and quality; for her hard work on my behalf; for her willingness to be available day in and day out; for introducing me to so many valuable lines of research; for encouraging me to stand up and fight for my fifty-year-old desk; and for teaching me the meaning of the word *comprise*. This thesis would not have been possible without her invaluable guidance.

I would like to thank Wheeler Ruml for being the best office-mate anyone could possibly hope for.

I would like to thank David Parkes for encouraging me to find the family of Temporal Decoupling Problems hiding behind the different algorithms I was trying to write; and Avi Pfeffer for many valuable comments and suggestions.

I would like to thank my brothers and sisters—Don, Elizabeth, Gretchen, Mark, Carol and Heidi—for their encouragement and support, especially during crunch time.

I would like to thank my mother for bringing me into this world and for praying for me down the stretch.

I would like to thank Paul and Kathy Szabo for their love and support over the years.

I would like to thank Mookie for not complaining too much even though he spent so many dog-years under my desk, waiting for me to finish working.

I would like to thank the Red Sox for staying in first place throughout *my* home stretch.

And, finally, most of all, I would like to thank Julie for her years of love and support, even through the really dark times.

Portions of Chapter 2 were described in the paper, *A Mechanism for Group Decision Making in Collaborative Activity* (Hunsberger and Zancanaro, 2000). Portions of Chapter 3 were described in the paper, *A Combinatorial Auction for Collaborative Planning* (Hunsberger and Grosz, 2000). Portions of Chapters 4 and 5 were described in the papers, *Generating Bids for Group-related Actions in the Context of Prior Commitments* (Hunsberger, 2002b) and *Algorithms for a Temporal Decoupling Problem in Multi-Agent Planning* (Hunsberger, 2002a).

This research was supported in part by NSF grants IIS-9978343, IRI-9618848, IRI-9525915 and CDA-9401024.

# Chapter 1

# Introduction

## 1.1 Motivation

The more capable, autonomous, and interactive that computer systems become, the more important it is for them to have some ability to act collaboratively, whether interacting with other computer systems or within heterogeneous groups of computers and people. For example, teams of mobile rovers gathering soil samples and taking atmospheric measurements on a distant planet (Estlin et al., 2000) need to be able to coordinate their activity to avoid duplicating each other's work, to avoid interfering with one another, and to offer assistance to one another when things go wrong. Similarly, large-scale military-training simulation systems in which computer agents simulate cooperating (and competing) battlefield participants (Tambe, 1997) demand a certain level of collaborative sophistication in their simulated combatants: teams of helicopter agents must carry out a bombing mission together; when one of them gets shot down, the others may want to think about picking up any survivors. Furthermore, consumer demand for more realistic video games provides a non-trivial economic incentive to construct computer agents able to act collaboratively (Laird, 2000).

## 1.2 Challenges of Group Decision Making in Collaborative Planning

It has been argued that "collaboration must be designed into systems from the start; it cannot be patched on" (Grosz, 1996). Attempts to design collaborative multi-agent systems without an underlying theory of collaboration have met a range of difficulties. For example, Jennings (1995) and Tambe (1997) report such undesirable behavior as (1) agents continuing to pursue an activity even after some team members discover it is destined to fail, (2) agents abandoning a collaborative activity without bothering to inform the rest of the team, and (3) agents needlessly waiting for the results of some activity that has been abandoned or significantly delayed.

Several researchers in Artificial Intelligence (AI) have proposed formal specifications of what it means for a group of agents to collaborate on a group activity (Levesque, Cohen, and Nunes, 1990; Kinny et al., 1994; Grosz and Kraus, 1996; Grosz and Kraus, 1999). In each case, the specifications stipulate an assortment of intentions, beliefs and mutual beliefs relating to their activity that collaborating agents must hold. However, none of these specifications generally explain the underlying source of such attitudes or the ways they change over time. They provide static definitions, but do not address the dynamics of group planning. A partial exception is the SharedPlans formalization of collaborative activity (Grosz and Sidner, 1990; Grosz and Kraus, 1996; Grosz and Kraus, 1999), which specifies that collaborating agents must be committed not only to *doing* their activity together, but also to *planning* their activity together. In particular, the SharedPlans formalization stipulates that collaborating agents must be committed to group planning processes aimed at selecting a method of doing their activity, assigning tasks to members of the group, and finding suitable values for action parameters.

However, the SharedPlans formalization does not specify any such processes nor make explicit how the *results* of such processes cause the intentions, beliefs and mutual beliefs of the group members to change. In effect, the SharedPlans formalization specifies discrete snapshots of collaborative activity, but does not explicate the transitions from one such snapshot to another. As a result, it provides an incomplete specification of the dynamic function of group decision making in collaborative activity.

This thesis addresses the need for a more complete specification of the dynamic function of group decision making in collaborative activity. It describes how group decisions authorize and oblige collaborating agents to coordinate the updating of their intentions as they expand their partial plans for a collaborative activity. It also (1) provides a framework for rigorously specifying group decision-making mechanisms that agents can use to generate group decisions, (2) specifies sample mechanisms according to that framework, and (3) provides a suite of algorithms that agents can use to effectively participate in such mechanisms.

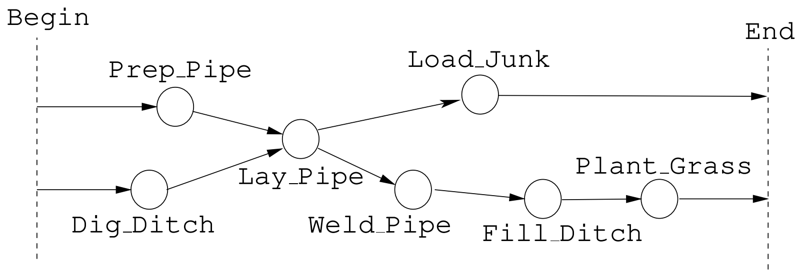
An important part of group decision making in collaborative activity centers on the problem of coordinating the times at which various tasks can be done. The second part of this thesis focuses on a range of temporal reasoning problems that agents need to be able to solve when planning their group activities. Algorithms for solving these temporal reasoning problems are derived, and their properties formally analyzed, using the Simple Temporal Networks framework (Dechter, Meiri, and Pearl, 1991).

## 1.3 Background and General Framework for Group Decision Making

### 1.3.1 Actions, Act-types and Recipes

The representation of actions, act-types and recipes used in this thesis follows that of Grosz and Kraus (1996). An action is an instance of an act-type. Act-types are either basic or complex. A basic action is an action that may be executed at will by an individual agent under appropriate conditions; a complex action is executed indirectly using a recipe. A *recipe* for a complex act-type is a set of subacts and constraints such that the doing of those subacts

Recipe: R39 (LAY_PIPELINE)




Subacts with Precedence Constraints

Additional Constraints

$$\begin{aligned} Agent(\texttt{Prep\_Pipe}) &= Agent(\texttt{Weld\_Pipe}) \\ Agent(\texttt{Lay\_Pipe}) &= Agent(\texttt{Load\_Junk}) \\ Agent(\texttt{Dig\_Ditch}) &= Agent(\texttt{Fill\_Ditch}) \end{aligned}$$

Figure 1.1: A sample recipe

under those constraints constitutes the doing of an action of that type. Typically, recipe constraints include precedence constraints on the execution times of the various subacts, as well as constraints that certain subacts be executed by the same agent or subgroup. As is standard in work on planning, act-type definitions specify preconditions, application constraints and necessary effects for an action of that type, as well as any parameters required in doing the action.

Figure 1.1 shows a sample recipe, R39, that specifies one way of doing an action of the LAY_PIPELINE type.[1] Precedence constraints are indicated by arrows in the figure; for example, the Weld_Pipe subact must be done before the Fill_Ditch subact. Although not shown in the figure, precedence constraints may include *offsets.* Thus, the Load_Junk subact might be constrained to begin no sooner than twenty minutes after the Lay_Pipe subact ends. Recipes may contain complex subacts; recursively choosing recipes for these subacts gives rise to a multi-level recipe hierarchy (Hunsberger, 1999). However, this thesis assumes that all recipes are fully expanded and, hence, that agents need only consider basic subacts.

[1]This thesis uses a teletype font for the names of act-types and recipes.

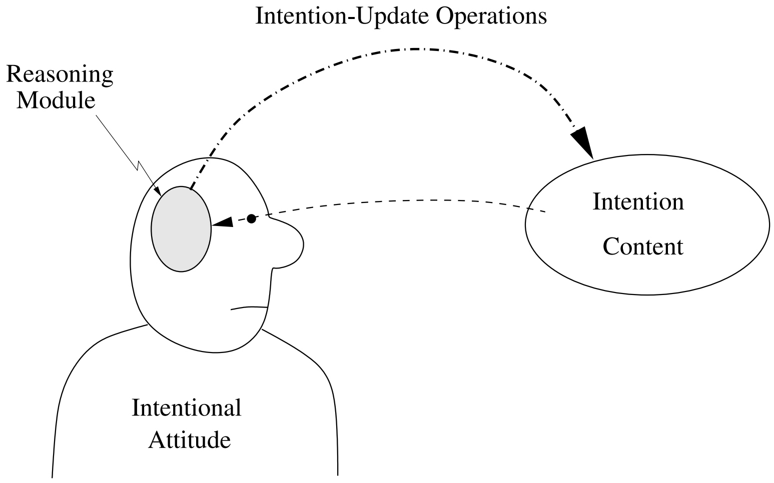



Figure 1.2: Intention cultivation in the context of single-agent activity

### 1.3.2 Intentions and Intention Cultivation

When an agent commits to doing some task, it typically forms a future-directed intention (Bratman, 1987). Intentions serve many purposes. They not only motivate, but also focus an agent's subsequent planning processes (Bratman, 1987). For example, Bob's intention to see a movie sometime next week might serve to focus his attention on deciding which movie he will see and whether he will drive or walk to the theater.

This thesis focuses on the class of planning decisions that agents use either to adopt new intentions or to update existing intentions. For example, given his intention to see a movie sometime next week, Bob might decide to see the movie *Notorious*, in which case he would update the content of his intention to reflect that choice; he would now intend to see *Notorious* sometime next week. Should he subsequently decide to drive to the theater, he would need to adopt a subsidiary intention (to drive to the theater). To simplify the presentation, the term *intention-update operation* will be taken to include operations for adopting new intentions as well as updating existing intentions.

In this thesis, the characteristic planning activity associated with an intention is called intention *cultivation*.[2] The process of intention cultivation is iterative: the content of an intention motivates an agent to make a planning decision, which causes it to update that intention, which motivates it to make further planning decisions, and so on. Importantly, in the context of single-agent activity, the iterative, intention-cultivation process is under the complete control of one agent, as illustrated in Figure 1.2. However, in the context of collaborative, multi-agent activity, the process of intention cultivation is more complex in several ways.

[2]Grosz and Kraus (1999) used this term to refer to the refinement of a particular class of intentions. This thesis uses the term in a more general sense.

### 1.3.3 The CCGI Model of the Coordinated Cultivation of Group-related Intentions

For collaborative, multi-agent activities, there are interactions among agents' individual intentions and, as a result, constraints on their intention-cultivation activity. These interactions and constraints pose several challenges for the design of collaboration-capable agents.

First, to maintain the compatibility of their group-related intentions over time, collaborating agents may not update their group-related intentions without some consensus from the group. For example, if a group of musicians are collaborating on a gig, none of them may make unilateral decisions about the type of van to rent or how much to charge for the gig; such decisions must be agreed to, or *authorized* by, the group. In the model presented in this thesis, the *coordinated-cultivation requirement* (CCR) explicitly represents this general prohibition against unilateral planning decisions in the context of collaborative activity.

Second, a group planning decision (e.g., to rent a luxurious van for transporting their musical equipment) not only authorizes the agents to update their group-related intentions, but also compels them to do so. In making an agreement, agents incur an obligation to update their intentions accordingly. This thesis explicitly represents the obligations that ensue from group decisions.

Third, agents require a means of generating group decisions. Although people working in small groups can be very adept at using casual methods of generating group decisions, larger groups benefit from structured decision-making mechanisms. For example, the people of the United States engage in a highly structured group decision-making process once every four years to decide which of them shall be the next president. This thesis provides a framework for rigorously specifying group decision-making mechanisms that computer agents can use to generate group decisions. The group decisions generated by any such mechanism must be mutually believed by the group; such mechanisms must also establish the obligations that result from the decisions they generate.

Fourth, since the decisions generated by a group decision-making mechanism typically subject agents to various obligations, agents must be able to reason about the obligations that might ensue from their participation in such a mechanism. For example, an agent bidding to do a set of tasks must be able to determine whether its bid might conflict with any of its pre-existing commitments. This thesis provides a set of algorithms that agents can use to solve the sorts of temporal reasoning problems that arise from the use of group decision-making mechanisms. Each algorithm solves a temporal reasoning problem that is formally defined and analyzed using the Simple Temporal Networks framework (Dechter, Meiri, and Pearl, 1991).

Figure 1.3 illustrates the model of the coordinated cultivation of group-related intentions in the context of collaborative activity (the CCGI model) proposed by this thesis. This model includes a specification of the coordinated-cultivation requirement that prohibits the unilateral updating of group-related intentions, depicted in Figure 1.3 as a clamp on the ability of agents to update their group-related intentions. The CCGI model also includes mechanisms for generating group decisions that both authorize and oblige agents to update their group-related intentions. The authorization and obligation resulting from such group

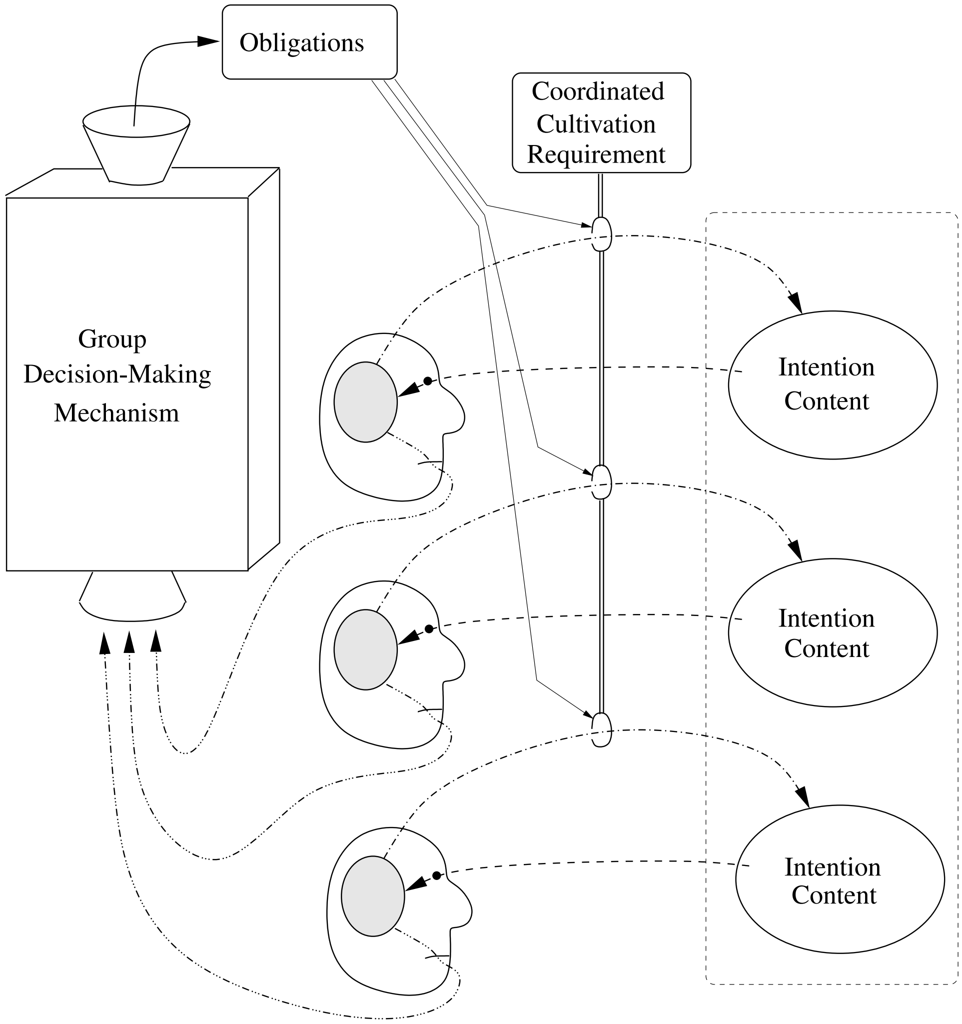



Figure 1.3: The CCGI model of the coordinated cultivation of group-related intentions in collaborative, multi-agent activity

decisions are depicted in Figure 1.3 by arrows emanating from the "obligations" box. Finally, as in the individual case, the content of an agent's group-related intention motivates it to update its intention; however, given the coordinated-cultivation requirement, the agent must seek the group's authorization for any intention-update operations it might want to perform. Thus, an agent's intention motivates it to participate in group decision-making mechanisms, as depicted by arrows from agents to the GDMM box in Figure 1.3.

### 1.3.4 Related Prior Work in Philosophy and AI

The CCGI model of the coordinated cultivation of group-related intentions described above succinctly captures many of the requirements of collaborative activity highlighted by a variety of researchers in philosophy and AI. In addition, the CCGI model provides a unifying framework for the different commitments to group planning processes specified by the SharedPlans formalization (Grosz and Kraus, 1996; Grosz and Kraus, 1999), as well as a more complete specification of the function of group decision making in the context of collaborative activity than any prior formalization.

#### Comparison to Approaches in Philosophy

Bratman (1999) argues that the intentions existing within the minds of agents are of one fundamental kind, whether in the context of single-agent or collaborative, multi-agent activity; that it is solely the *content* of agent intentions that distinguishes the two cases, as illustrated by the difference between my intending *to play my guitar by myself* versus my intending *that we paint the house together*. However, in his analysis of "We intend to $J$", Bratman requires not only that each of us "intend that we $J$", but also that each of us "intend that we $J$ in accordance with and because of [our intentions that we $J$] and meshing subplans [arising from our intentions that we $J$]."

In contrast to Bratman, Searle (1990) argues that intentions in single-agent and collaborative, multi-agent activity cannot be distinguished solely by their differing content. Instead, he claims that a fundamentally different kind of intention, called a *we-intention*, is at work in collaborative activity. According to Searle, a we-intention has the form "*We* intend that we perform act A." Although we-intentions exist in the minds of individual agents, Searle claims that "the reference to the collective lies outside the bracket that specifies the propositional content of the intentional state." Part of Searle's argument that a web of I-intentions is insufficient to generate "collective intentionality" is that "the notion of a we-intention, of collective intention, implies the notion of *cooperation*."

Tuomela (1995) argues that collaborative groups require what he calls a *socially-existing authority system* (SEAS) and that members of a group "submit their will to a group will" with respect to a range of issues related to their group activity. He also observes that "for an SEAS to do the work it is supposed to do, the members of [a group] must have some motivation to use it."

Gilbert (2000) argues that shared intentions are distinguished by a group's joint commitment to "intending as a body."

In the CCGI model, the coordinated-cultivation requirement forces me to seek your approval for planning decisions concerning our doing of $J$; the model thus provides a mechanism that accounts for how I can intend that we $J$ in accordance with *your* intention that we $J$. The CCGI model also explains how a web of I-intentions, suitably constrained by the coordinated-cultivation requirement, motivates agents to participate in group decision-making mechanisms and, hence, to cooperate and to plan their activity together. In that sense, it accommodates the notion of "group will".

As will be seen in Chapter 2, the CCGI model includes a framework for rigorously specifying group decision-making mechanisms. Furthermore, it explains the motivation of collaborating agents to *use* such mechanisms, as well as the ways agents can intend "as a body" (i.e., coordinate the cultivation of their group-related intentions).

### Comparison to Other Approaches in AI

The SharedPlans formalization of collaborative activity (Grosz and Kraus, 1996; Grosz and Kraus, 1999) stipulates that collaborating agents must be committed to selecting a recipe for their activity, to assigning tasks to agents or subgroups, and to finding suitable values for action parameters. The CCGI model provides a unifying framework for this network of commitments to planning processes by identifying and highlighting their common roots in the need for the coordinated cultivation of group-related intentions. In addition, the CCGI model distinguishes the motivation for participating in group decision-making mechanisms from the specification of the various decision problems that collaborating agents need to address. The motivation for participating in group decision-making mechanisms arises both from agent intentions and the coordinated-cultivation requirement; the various decision problems that agents must address arise from the content of their intentions. Finally, by explicitly representing group decisions and the resulting obligations, the CCGI model clarifies how group decisions cause agent intentions to change over time as a group expands its partial plan.

Levesque, Cohen and Nunes (1990), building on earlier work by Cohen and Levesque (1990), focus on the persistence of agent intentions in collaborative activity. However, as observed by Singh (1992), their model does not adequately provide for agents being motivated to eventually do something to achieve their goals.

In the context of an alternative account of joint intentions, Kinny et al. (1994) provide algorithms agents can use to agree upon a completely specified plan, including a complete allocation of tasks to agents, at the time a team is formed for a proposed group activity. The members of the team establish the corresponding network of intentions, beliefs and mutual beliefs in a single, synchronized act. Like SharedPlans, the CCGI model generalizes the approach of Kinny et al. along several dimensions; for instance, recipes for group activities can be partial and agents can incrementally expand their partial plans. In addition, the CCGI model provides a more general explication of the process whereby group decision-making mechanisms lead to the establishment and subsequent modification of agent intentions over time.

## 1.4 Road Map for the Remainder of the Thesis

Chapter 2 presents a framework for rigorously specifying group decision-making mechanisms that agents can use to generate group decisions to coordinate the cultivation of their group-related intentions, as specified in the CCGI model. The properties of a sample, proposal-based mechanism are formally analyzed to show that, under a specified set of assumptions about agent beliefs, group decision-making mechanisms properly associate group decisions with the intention updates that they authorize. The coordinated-cultivation requirement is also formally defined in this chapter.

The remainder of the thesis focuses on the problem of how agents, when faced with a proposed group activity, can determine, both individually and collectively, whether to commit to that activity, a problem this thesis refers to as the Initial-Commitment Decision Problem (ICDP). Chapter 3 specifies a group decision-making mechanism based on a combinatorial auction in which agents bid on sets of tasks in a proposed group activity. To enable agents to protect the feasibility of their private schedules of pre-existing commitments, agents are allowed to condition their bids on temporal constraints. Thus, to participate in an ICDP auction, agents must be able to solve a variety of temporal reasoning problems.

Chapter 5 formally treats an important family of temporal reasoning problems that are directly related to the auction-based mechanism. Each problem is formally defined and analyzed using the Simple Temporal Networks framework (Dechter, Meiri, and Pearl, 1991). The existing theory of Simple Temporal Networks is summarized in Chapter 4. In addition, that chapter provides several useful extensions to the theory. Chapter 6 contains concluding remarks.

# Part I

# Group Decision Making

# Chapter 2

# A Framework for Specifying Group Decision-Making Mechanisms

Agents collaborating on a group activity need to make decisions about how they will carry out that activity. People make group decisions using both informal and formal mechanisms. For instance, people collaborating on projects such as putting a swing set together or taking a cross-country trip may informally debate and negotiate about the best way to do their activity. In contrast, the bylaws of a corporation may specify detailed mechanisms for certain corporate decisions. Although people can be quite adept at engaging in free-form group decision-making, computer agents generally are not. This chapter presents a framework for specifying group decision-making mechanisms that groups of computer agents can use to coordinate their group decision-making activity.

Throughout this chapter, we will use the example of our planning to go to the movies together. We need to make various planning decisions. The content of such decisions can be expressed in terms of intention-update operations. For example, if we decide to see *The Wizard of Oz*, we need to update our intentions accordingly: instead of intending that we see *some* movie, each of us now intends that we see *The Wizard of Oz*. Next, we may need to decide which theater to attend, and so on. Thus, agents make group decisions to coordinate the planning and eventual execution of the group activity to which they are committed. Decisions made in the context of existing intentions are called *context-bound* decisions.

Agents must also be able to make decisions outside of any pre-existing context. For example, prior to being committed to working together, a group of agents need to be able to decide whether to *establish* such a commitment in the first place. Such decisions are called *context-free* decisions. For example, our initial decision to commit to seeing a movie together was a context-free decision. Context-free decisions establish not only a group's commitment to a new activity, but also their commitment to doing that activity *collaboratively*. In other words, in collaborative activity, agents must be committed not only to *doing* the activity together, but also to *planning* the activity together.

To plan their activity together, agents must be able to make decisions together. They must be able to generate group decisions. The purpose of a group decision-making mechanism is to provide computer agents with a well-defined, reliable means of generating group decisions. This chapter presents a framework for rigorously specifying group decision-making mechanisms (GDMMs) that agents can use to coordinate their planning activity.

The GDMM framework can be used to specify a wide variety of group decision-making mechanisms. In a proposal-based mechanism, agents make proposals, vote on proposals and announce group decisions. In an auction-based mechanism, agents invoke auctions, submit bids, and announce auction results. The purpose of rigorously specifying a group decision-making mechanism is to allow important properties of the mechanism to be proven and to make explicit the assumptions required to ensure that those properties hold.

The mechanisms specified according to the GDMM framework may be used to generate context-free decisions—those that establish a group's initial commitment to some new activity—or context-bound decisions—those used to coordinate an activity to which the group is already committed.

To illustrate the use of the GDMM framework, a sample proposal-based mechanism is used as a running example throughout this chapter. The chapter concludes with a formal analysis of the properties of the sample, proposal-based mechanism. Chapter 3 uses the GDMM framework to specify a mechanism based on a combinatorial auction for solving the Initial-Commitment Decision Problem.

## 2.1 Background

The group decision-making mechanisms specified according to the GDMM framework make use of declarative speech-acts, intention-update operations and social obligations.

### 2.1.1 Declarative Speech-Acts

Austin (1962) observed that in certain circumstances, merely saying something can make it so. For example, a Justice of the Peace can, in certain circumstances, cause a couple to become married, merely by declaring them to be married. Declarative speech-acts—not to be confused with declarative *sentences* such as "The sky is blue"—are unique among speech-acts in that, under appropriate circumstances, they have the power to make their propositional content true merely by their being uttered (Searle, 1995).

It is exclusively through declarative speech-acts that agents participate in the group decision-making mechanisms specified according to the GDMM framework. For example, in the proposal-based mechanism, agents, in effect, make declarations such as, "I declare that I have made a proposal", "I declare that I have voted to accept your proposal", or "I declare that the group has made a decision to accept my proposal." Similarly, in the auction-based mechanism, agents make declarations such as, "I declare that I have submitted a bid for tasks A, B and C", or "I declare that your bid has been awarded."

Declarative speech-acts have the power to establish the truth of their propositional content only if certain conditions apply. In general, these conditions, which are called *authorization conditions*, depend on the type of propositional content. For example, the authorization conditions for a declaration of marriage are usually not the same as the authorization conditions for a declaration of war. In the GDMM framework, a specification for a group decision-making mechanism includes not only the classes of allowable propositional content (e.g., proposals, votes, announcements), but also the authorization conditions for each such class. In the formal specification of authorization conditions for declarative speech-acts, this chapter draws from the approach taken by Dignum and Weigand (1995a; 1995b).

### 2.1.2 Intention-Update Operations

If I intend to go to the movies sometime next week, I need to make certain planning decisions. For example, I need to pick a movie to see and a theater to see it in. If I decide to see *The Wizard of Oz*, I must update my intention. No longer do I intend to see any old movie; I intend to see *The Wizard of Oz*. Although agents collaborating on a group activity need to be careful about *how* they make their planning decisions, the result of each such planning decision is always some kind of intention-update operation.[1]

The formalization in this chapter presumes a set of intention-update operations that agents can use to update their intentions in the context of single-agent or group activity. For example, that act of updating the content of an intention identified by the constant $I$ to reflect the binding of a parameter ?P to the value v can be represented by the following action expression:[2]

$$Update(I, \texttt{Bind}(\texttt{?P}, \texttt{v})).$$

In general, if $\upsilon$ is an intention-update act-type and $a_1, \ldots, a_n$ is a set of arguments appropriate for $\upsilon$, then $Update(I, \upsilon(a_1, \ldots, a_n))$ represents an intention-update operation to be applied to the intention identified by $I$.

In addition to intention-update operations, this chapter presumes an intention-adoption act-type, $AdoptInt$, where $AdoptInt(I, \alpha, GR, \Upsilon)$ represents the (single-agent) action of adopting an intention to be associated with the identifier $I$, concerning the doing of an action of type $\alpha$ by the group $GR$; in the case of single-agent activity, the group $GR$ is a singleton. The optional fourth argument, $\Upsilon$, represents an ordered list of intention-update operations (e.g., a set of temporal constraints) to be applied to the new intention immediately upon its being established.

[1]There are some kinds of intention-update operations (e.g., selecting an agent or subgroup to do some constituent task) that only make sense in the context of group activity.

[2]Ortiz (1999) defines a generic $Update$ operator that can accommodate arbitrary intention updates in Hypothetical Logic, but does not explicitly categorize intention-update operations as described here.

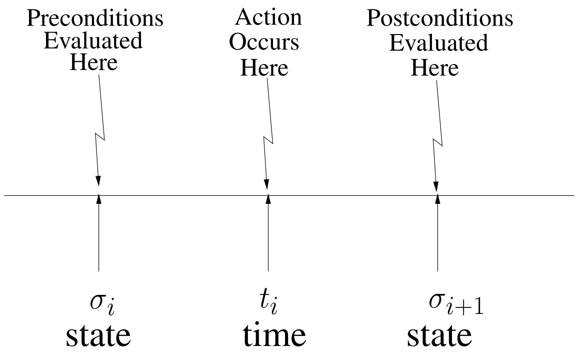



Figure 2.1: The semantics of DDLTLB

### 2.1.3 Social Obligations

The mechanisms specified by the GDMM framework establish group decisions in terms of *social obligations*.[3] A social obligation has the form

$$\mathcal{O}(G, GR, X),$$

which represents that agent $G$ is obliged toward the group $GR$ to do the action $X$. This thesis restricts attention to social obligations for which the obliged action is a (single-agent) mental action, either to adopt a new intention or to update the contents of an existing intention.

### 2.1.4 Dynamic Deontic Linear Time Temporal Logic (DDLTLB)

This chapter uses Dynamic Deontic Linear Time Temporal Logic (DDLTLB), developed by Dignum and Kuiper (1997), to represent speech-acts (in dynamic logic) and social obligations (in deontic logic) within a temporal framework.[4] The semantics of DDLTLB allows dynamic operators to be combined with temporal operators by evaluating the preconditions at a state $\sigma_i$ that can be arbitrarily close to, but *before* the time $t_i$ that an action is done, and the postconditions at a state $\sigma_{i+1}$ that can be arbitrarily close to, but *after* $t_i$, as illustrated in Figure 2.1.

Consider the following typical expression from dynamic logic: $[A]\phi$, which represents that the doing of action $A$ would result in $\phi$ holding. If this expression evaluates to `True` at state $\sigma_i$, then the doing of $A$ at time $t_i$ would ensure that $\phi$ evaluates to true at state $\sigma_{i+1}$, as illustrated in Figure 2.2.

[3] Singh (1991; 1996; 1999; 2000) and Castelfranchi (1995) refer to what we are calling social obligations as "social commitments". Royakkers and Dignum (1999) use expressions of the form $COMM(i, X, \beta)$ "to stand for 'agent $i$ commits to the group $X$ to perform action $\beta$'." Cohen and Levesque (1990), Levesque, Cohen and Nunes (1990) and Cohen, Levesque and Smith (1997) discuss joint commitments of agents engaged in group activity.

[4] The "B" in DDLTLB indicates that the temporal logic has "both" past and future tenses.

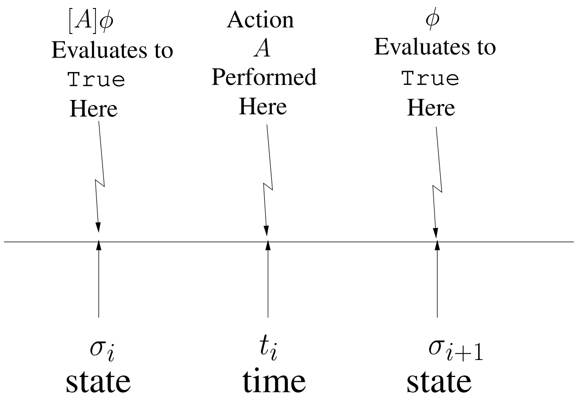



Figure 2.2: The semantics of $[A]\phi$ in DDLTLB

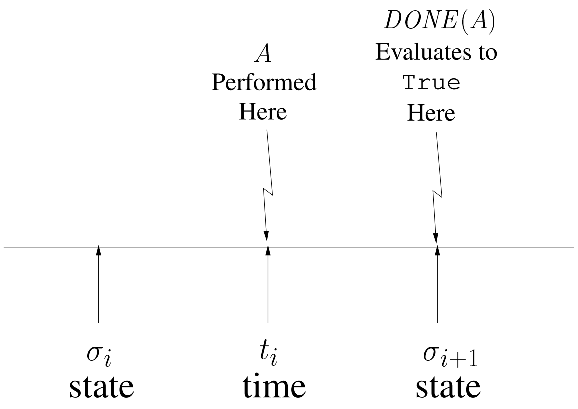



Figure 2.3: The semantics of $DONE$ in DDLTLB

DDLTLB also includes a $DONE$ operator used to represent that an action was just done. For example, the expression $DONE(A)$ would evaluate to `True` at the state $\sigma_{i+1}$ if the action $A$ was done at the time-point $t_i$, as illustrated in Figure 2.3.

**Temporal Operators.** DDLTLB includes many temporal operators, but only three are used in this chapter: $\mathcal{P}$, $\Diamond^-$ and $\Box$. Expressions of the form $\mathcal{P}\phi$ represent that $\phi$ held at the *previous state*. In other words, $\mathcal{P}\phi$ evaluates to `True` at the state $\sigma_{i+1}$ if $\phi$ evaluates to `True` at the state $\sigma_i$, as illustrated in Figure 2.4. Similarly, expressions of the form $\Diamond^-\phi$ represent that $\phi$ held at some time in the (possibly distant) past. This chapter also uses the following abbreviation:

$$\Diamond^{\leftarrow}\phi \;\equiv\; \phi \vee \Diamond^-\phi,$$

which represents that $\phi$ holds now or held at some point in the past. Finally, the definition of the coordinated-cultivation requirement (in Section 2.4) uses propositions of the form, $\Box\phi$, representing that $\phi$ holds now and at all times in the future.

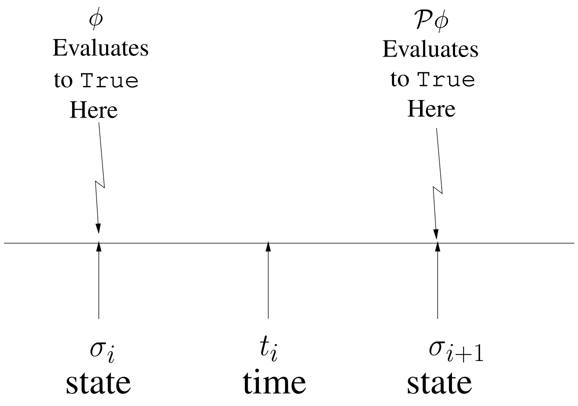



Figure 2.4: The semantics of $\mathcal{P}$ in DDLTLB

**The Obligation Operator.** In DDLTLB, the obligation operator $\mathcal{O}$ is applied to propositions. For example, $\mathcal{O}(\phi)$ represents that $\phi$ is obliged to hold in the current state. Our application of obligation operators to actions can easily be accommodated, for example, by the following translation:

$$\mathcal{O}(A) \ \mapsto\ \mathcal{O}(DONE(A)),$$

where $DONE$ is the DDLTLB operator seen earlier. Furthermore, for each agent $G$ and each group $GR$, the construction $\mathcal{O}(G, GR, \ldots)$ may be viewed as a distinct social-obligation operator.

## 2.2 Specifying a Group Decision-Making Mechanism

In the GDMM framework, the specification of a group decision-making mechanism includes:

- a specification of the allowable *content* of agent declarations (e.g., "that a proposal has been made" or "that a bid has been submitted"); and
- a specification of the authorization conditions for each type of allowable content (e.g., "an agent can only vote to accept a proposal that it has not already rejected" or "an agent can announce a group decision to accept a proposal only if every other agent voted to accept that proposal").

For a mechanism to generate group decisions, it must be possible for some combination of declarations to establish conditions that authorize some agent to declare on behalf of the group that the group has made a decision.

### 2.2.1 Declarations in the GDMM Framework

Agent inputs to a group decision-making mechanism are restricted to declarations. The representation of declarative speech-acts presented in this section draws from Dignum and

Weigand (1995a; 1995b).

The declaration of an agent $G$ to a group of agents $GR$ that a proposition $\phi$ holds is represented by the following action expression:

$$\delta(G, GR, \phi).$$

For convenience, we define the following abbreviation:

$$declared(G, GR, \phi) \equiv DONE(G, \delta(G, GR, \phi)).$$

We assume that the speaker of a declaration is always one of the hearers.

$$\models declared(G, GR, \phi) \Rightarrow (G \in GR)$$

However, in the case of one-on-one communication, we may write $\delta(G_1, G_2, \phi)$ instead of the more cumbersome $\delta(G_1, \{G_1, G_2\}, \phi)$.

**Axiom 2.1** *The declaration of a conjunction is equivalent to the corresponding conjunction of simultaneous declarations.*[5]

$$\models declared(G, GR, \wedge_i \phi_i) \equiv \wedge_i \; declared(G, GR, \phi_i)$$

### 2.2.2 Authorization Conditions

In the GDMM framework, following the approach taken by Dignum and Weigand (1995a; 1995b), the "right circumstances" for making a declaration are specified in terms of "authorizing conditions." However, in the context of group activity, there is always an authorizing agent or group; thus, authorizing conditions for the GDMM framework include the authorizing party as an argument. The predicate

$$auth(G, GR, D)$$

represents that agent $G$ is authorized by group $GR$ to do action $D$, where $D$ represents a declarative speech-act.

Axiom 2.2 shows how an authorization predicate defines the conditions under which a declaration establishes the truth of its propositional content, as illustrated in Figure 2.5.

**Axiom 2.2**

$$\models auth(G, GR, \delta(G, GR, \phi)) \Rightarrow [\delta(G, GR, \phi)]\, \phi$$

Notice that the intended interpretation of $[\delta(G, GR, \phi)]\, \phi$ in the above axiom is: if the agent $G$ declares to the group $GR$ that $\phi$ holds, then $\phi$ holds.

[5] Dignum and Weigand (1995a) make a similar assumption.

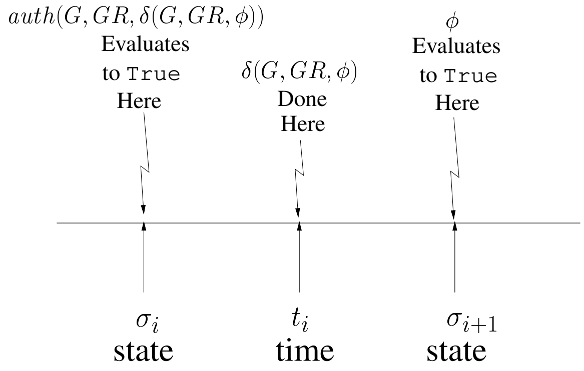



Figure 2.5: Authorization conditions for a declaration

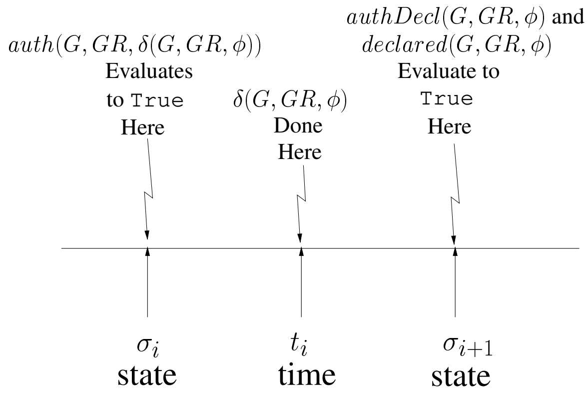



Figure 2.6: An authorized declaration

For convenience, we define an abbreviation for the occurrence of an authorized declaration.

$$authDecl(G, GR, \phi) \equiv \begin{cases} \mathcal{P}\ auth(G, GR, \delta(G, GR, \phi)) \\ \wedge \\ declared(G, GR, \phi) \end{cases}$$

where $\mathcal{P}$ is the "previous" operator described earlier. The semantics for an authorized declaration is illustrated in Figure 2.6.

### 2.2.3 Allowable content of agent declarations in a GDMM

The specification of a GDMM includes a specification of the classes of allowable content for declarations. For example, the content of declarations in the proposal-based mechanism is restricted to formulas involving special predicates such as $MadeCFP$, $AcceptedCFP$, and $RejectedCFP$. The following axiom stipulates that instances of such predicates hold only as a result of an authorized declaration.

**Axiom 2.3** *If* $\phi(G, GR, a_1, a_2, \ldots, a_n)$ *is an instance of any of the special predicates associated with a group decision-making mechanism (e.g., those listed above for the proposal-based mechanism), then*

$$\models \quad \phi(G, GR, a_1, a_2, \ldots, a_n) \;\Leftrightarrow\; authDecl(G, GR, \phi(G, GR, a_1, a_2, \ldots, a_n)).$$

### 2.2.4 Declarations for a Proposal-based Mechanism: Context-Free Proposals

This section presents the allowed propositional content for a sample, proposal-based mechanism. The allowed declarations include making proposals, voting on proposals and announcing decisions. The proposal-based mechanism can be used both for context-free decisions (e.g., to establish a group's commitment to some proposed activity) or for context-bound decisions (e.g., to update intentions relating to some activity to which the group is already committed). This section restricts attention to the case of context-free proposals. Context-bound proposals are treated in Section 2.2.7.

The content of declarations used in the proposal-based mechanism in the case of context-free proposals is restricted to one of the following forms, each shown with its intended interpretation):

- $\delta(G_1, GR, \mathit{MadeCFP}(G_1, GR, I, \alpha, \Upsilon_0))$

  Agent $G_1$ proposes to the group $GR$ that they establish a shared intention to do an action of type $\alpha$, to be subject to the coordinated-cultivation requirement, where $I$ is a unique identifier for the proposed intention (which also serves to identify the proposal itself) and $\Upsilon_0$ is an (optional) ordered list of intention-update operations to be applied to the new intention, if adopted (e.g., $\Upsilon_0$ might contain a set of *add-temporal-constraint* operations corresponding to a set of temporal constraints specific to the proposed activity).

- $\delta(G_2, G_1, \mathit{AcceptedCFP}(G_2, G_1, GR, I, \alpha, \Upsilon_0))$

  Agent $G_2$ declares to agent $G_1$ that it accepts the proposal that the group $GR$ commit to doing $\alpha$ together.

- $\delta(G_2, G_1, \mathit{RejectedCFP}(G_2, G_1, GR, I, \alpha, \Upsilon_0))$

  Agent $G_2$ declares to agent $G_1$ that it rejects the proposal that the group $GR$ commit to doing $\alpha$ together.

- $\delta(G_1, GR, CCR^+(GR, I, \alpha, \Upsilon_0))$

Agent $G_1$ declares that the group $GR$ has decided to accept the proposal that the group $GR$ commit to doing $\alpha$ together, where the formula, $CCR^+(GR, I, \alpha, \Upsilon_0)$, is an abbreviation for the following expression:[6]

$$\begin{cases} \Box\, CCR(GR, I) \\ \wedge \\ (\forall G \in GR)\ \mathcal{O}(G, GR, \mathit{AdoptInt}(I, \alpha, GR, \Upsilon_0)) \end{cases}$$

where $\Box$ is the "all-future-times" operator in DDLTLB (cf. Section 2.1.4), and the coordinated-cultivation requirement, $CCR(GR, I)$, is as defined below in Section 2.4.[7]

- $\delta(G_1, GR, \mathit{GroupRejectedCFP}(G_1, GR, I, \alpha, \Upsilon_0))$

  Agent $G_1$ declares that the group $GR$ has decided to reject the proposal that the group $GR$ commit to doing $\alpha$ together.

Notice that providing the rejection declarations is not strictly necessary, but may facilitate an agent's reasoning about the proposal-based mechanism.

### 2.2.5 Authorization for Declarations in the Proposal-Based Mechanism: Context-Free Proposals

This section specifies the conditions under which an agent is authorized to make the types of declarations allowed in the sample, proposal-based mechanism in the case of context-free proposals. Several of the authorization conditions employ the temporal operator $\Diamond^{\leftarrow}$ (cf. Section 2.1.4). For example, the authorization conditions for a voting declaration include the condition $\Diamond^{\leftarrow}\mathit{MadeCFP}(G_1, GR, I, \alpha, \Upsilon_0)$ (i.e., that the agent $G_1$ made the specified context-free proposal at some point in the past).

- **Context-Free Proposal**: Any member $G_1$ of a group $GR$ may make a context-free proposal to the group:

$$\begin{aligned} \models\ \ & \mathit{auth}(G_1, GR,\ \delta(G_1, GR, \mathit{MadeCFP}(G_1, GR, I, \alpha, \Upsilon_0)) \\ & \Leftrightarrow\ (G_1 \in GR) \end{aligned}$$

  where it is assumed that $I$ is a unique identifier to be assigned to the new intention (if adopted), $\alpha$ is a complex act-type, and $\Upsilon_0$ is an ordered list of intention-update operations.

- **Vote to Accept or Reject**: An agent $G_2$ is authorized to vote to accept or reject a proposal if and only if:

[6]The universally quantified clause is a convenient syntactic abbreviation for a conjunction of clauses, one for each agent in the group, made possible by the assumption that the contents of the group $GR$ are known and fixed.

[7]The formal analysis of the proposal-based mechanism in Section 2.3 is independent on the definition of the coordinated-cultivation requirement.

(1) the proposal was made at some point in the past,

(2) the voting agent is not the originator of the proposal,

(3) the voter has not already voted to accept,

(4) the voter has not already voted to reject, and

(5) the originator of the proposal has not already made an announcement that the group rejected the proposal.

$$\begin{aligned} \models \;& \mathit{auth}(G_2, G_1, \delta(G_2, G_1, \mathit{AcceptedCFP}(G_2, G_1, GR, I, \alpha, \Upsilon_0))) \\ & \Leftrightarrow\; V_1 \wedge V_2 \wedge V_3 \wedge V_4 \wedge V_5 \end{aligned}$$

where:

$$\begin{aligned} V_1 &\equiv \Diamond^{\leftarrow} \mathit{MadeCFP}(G_1, GR, I, \alpha, \Upsilon_0) \\ V_2 &\equiv (G_2 \neq G_1) \\ V_3 &\equiv \neg \Diamond^{\leftarrow}\, \mathit{declared}(G_2, G_1, \mathit{AcceptedCFP}(G_2, G_1, GR, I, \alpha, \Upsilon_0)) \\ V_4 &\equiv \neg \Diamond^{\leftarrow}\, \mathit{declared}(G_2, G_1, \mathit{RejectedCFP}(G_2, G_1, GR, I, \alpha, \Upsilon_0)) \\ V_5 &\equiv \neg \Diamond^{\leftarrow}\, \mathit{declared}(G_1, GR, \mathit{GroupRejectedCFP}(G_1, GR, I, \alpha, \Upsilon_0)) \end{aligned}$$

The authorization for voting to reject a proposal is the same.

- **Announce Group Acceptance**: An agent $G_1$ is authorized to announce that the group $GR$ has decided to commit to a proposed group activity if and only if:

  (1) $G_1$ originated the proposal,

  (2) $G_1$ has not already made an announcement that the group rejected the proposal,

  (3) $G_1$ has not already made an announcement that the group accepted the proposal, and

  (4) the rest of the agents in the group $GR$ voted to accept the proposal.

  $$\begin{aligned} \models \;& \mathit{auth}(G_1, GR,\ \delta(G_1, GR,\ CCR^{+}(GR, I, \alpha, \Upsilon_0))) \\ & \Leftrightarrow\; A_1 \wedge A_2 \wedge A_3 \wedge A_4 \end{aligned}$$

  where:

  $$\begin{aligned} A_1 &\equiv \Diamond^{\leftarrow} \mathit{MadeCFP}(G_1, GR, I, \alpha, \Upsilon_0) \\ A_2 &\equiv \neg \Diamond^{\leftarrow}\, \mathit{declared}(G_1, GR, \mathit{GroupRejectedCFP}(G_1, GR, I, \alpha, \Upsilon_0)) \\ A_3 &\equiv \neg \Diamond^{\leftarrow}\, \mathit{declared}(G_1, GR, CCR^{+}(GR, I, \alpha, \Upsilon_0)) \\ A_4 &\equiv (\forall G \in GR, G \neq G_1)\ \Diamond^{\leftarrow} \mathit{AcceptedCFP}(G, G_1, GR, I, \alpha, \Upsilon_0) \end{aligned}$$

  where the universally quantified proposition in $A_4$ is a syntactic abbreviation for a conjunction of propositions, one for each agent in the group other than $G_1$.

- **Announce Group Rejection**: An agent $G_1$ is authorized to announce that the group $GR$ has decided to reject a proposal that they commit to doing $\alpha$ together if and only if:

  (1) $G_1$ originated the proposal;

  (2) $G_1$ has not already made an announcement that the group rejected the proposal; and

  (3) $G_1$ has not already made an announcement that the group accepted the proposal.

  $$\begin{aligned}\models\ \ & auth(G_1, GR,\ \delta(G_1, GR, GroupRejectedCFP(G_1, GR, I, \alpha, \Upsilon_0)) \\ & \Leftrightarrow\ A_1 \wedge A_2 \wedge A_3\end{aligned}$$

  Thus, the originator is allowed to announce the group's rejection of a proposal even if all other agents voted in favor of it, which is useful if the originator has discovered, for example, that changes in the environment have made the proposal inconsistent with the group's shared intention.

### 2.2.6 Example: Using the Proposal-based Mechanism to Establish a Group's Commitment to some new Activity

Context-free proposals are used to establish a group's commitment to some new collaborative activity. The result of a group decision to accept a context-free proposal is that each agent is obliged to adopt a new intention relating to the group activity. Furthermore, that new intention shall be subject to the coordinated-cultivation requirement. This section illustrates how a simple sequence of declarations in the proposal-based mechanism can be used by a group $GR$ of agents to establish a group decision to commit to some new collaborative activity $\alpha$.

**The Proposal.** The proposal, made by an arbitrary agent $G_1 \in GR$, is a context-free proposal that the group commit to doing some activity $\alpha$ together:

$$declared(G_1, GR, MadeCFP(G_1, GR, I_{64}, \alpha, \Upsilon_0)),$$

where $I_{64}$ shall be the unique identifier for the shared intention should the group accept the proposal, and $\Upsilon_0$ represents an optional list of intention-update operations to be applied to the new intention, if adopted.

Since $G_1 \in GR$, the above declaration is authorized:

$$auth(G_1, GR, MadeCFP(G_1, GR, I_{64}, \alpha, \Upsilon_0)).$$

Thus, the propositional content is established:

$$MadeCFP(G_1, GR, I_{64}, \alpha, \Upsilon_0).$$

**Voting on the Proposal.** That $G_1$ made a proposal authorizes the rest of the agents in the group $GR$ to vote either to accept or reject that proposal:

$$(\forall G \in GR, G \neq G_1)\ \mathit{auth}(G, G_1, \delta(G, G_1, \mathit{AcceptedCFP}(G, G_1, GR, I_{64}, \alpha, \Upsilon_0)))$$
$$(\forall G \in GR, G \neq G_1)\ \mathit{auth}(G, G_1, \delta(G, G_1, \mathit{RejectedCFP}(G, G_1, GR, I_{64}, \alpha, \Upsilon_0))).$$

Suppose that each of them votes to accept the proposal:

$$(\forall G \in GR, G \neq G_1)\ \mathit{declared}(G, G_1, \mathit{AcceptedCFP}(G, G_1, GR, I_{64}, \alpha, \Upsilon_0)).$$

Since these are authorized declarations, the corresponding propositional content is established:

$$(\forall G \in GR, G \neq G_1)\ \mathit{AcceptedCFP}(G, G_1, GR, I_{64}, \alpha, \Upsilon_0).$$

**The Announcement of a Group Decision.** Since every agent voted to accept $G_1$'s proposal, $G_1$ is authorized to announce the group's acceptance of that proposal:

$$\mathit{auth}(G_1, GR,\ \delta(G_1, GR,\ CCR^{+}(GR, I_{64}, \alpha, \Upsilon_0))).$$

Suppose $G_1$ does so:

$$\mathit{declared}(G_1, GR,\ CCR^{+}(GR, I_{64}, \alpha, \Upsilon_0)).$$

Because this is an authorized declaration, its content

$$CCR^{+}(GR, I_{64}, \alpha, \Upsilon_0) \equiv \begin{cases} \Box\, CCR(GR, I_{64}) \\ \wedge \\ (\forall G \in GR)\ \mathcal{O}(G, GR, \mathit{AdoptInt}(I_{64}, \alpha, GR, \Upsilon_0)) \end{cases}$$

is established. Thus, the agents have established that each of them is obliged to adopt a new intention (to be identified by $I_{64}$), the cultivation of which shall be subject to the coordinated-cultivation requirement (cf. Section 2.4).

At this point, the agents are ready to begin cultivating their shared intention. Thus, they need to make context-bound decisions. The following sections add context-bound proposals to the sample mechanism.

### 2.2.7 Declarations for the Proposal-based Mechanism: Context-Bound Proposals

Context-bound proposals are used to coordinate the cultivation of existing, shared intentions. Thus, the content of a context-bound proposal specifies a list $\Upsilon$ of intention-update operations to be applied to the group's shared intention. This section presents the allowed propositional content for the sample, proposal-based mechanism in the case of context-bound proposals.

Some of the authorization conditions (e.g., a condition that *some* agent made an authorized declaration to establish the coordinated-cultivation requirement for this intention) would be naturally expressed using existential quantification; however, DDLTLB does not accommodate existential quantification. Therefore, to avoid the need for existential quantification in certain authorization conditions, the special predicates for context-bound proposals carry along the arguments from the context-*free* proposal that established the coordinate-cultivation requirement for the shared intention under consideration.

The content of declarations used in the proposal-based mechanism in the case of context-bound proposals is restricted to one of the following forms, each shown with its intended interpretation):

- $\delta(G_1', GR, \mathit{MadeCBP}(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0))$

  Agent $G_1'$ proposes to the group $GR$ that they perform the intention-update operations listed in $\Upsilon$. $P$ is a unique identifier for the proposal. The rest of the arguments are carried over from the context-*free* proposal that established the group's commitment to doing $\alpha$ together.

- $\delta(G_2, G_1', \mathit{AcceptedCBP}(G_2, G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0))$

  Agent $G_2$ declares to agent $G_1'$ that it accepts the context-bound proposal identified by $P$.

- $\delta(G_2, G_1', \mathit{RejectedCBP}(G_2, G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0))$

  Agent $G_2$ declares to agent $G_1'$ that it rejects the context-bound proposal identified by $P$.

- $\delta(G_1', GR, \mathit{GrAccCBP}^+(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0))$

  Agent $G_1'$ declares that the group $GR$ has decided to accept the context-bound proposal identified by $P$.

  In the above formula, $\mathit{GrAccCBP}^+(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0)$ abbreviates

$$\left\{ \begin{array}{l} (\forall G \in GR)\ \mathcal{O}(G, GR, \mathit{Update}(I, \Upsilon)) \\ \wedge\ \mathit{GroupAcceptedCBP}(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0). \end{array} \right.$$

- $\delta(G_1', GR, \mathit{GroupRejectedCBP}(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0))$

  Agent $G_1'$ declares that the group $GR$ has decided to reject the context-bound proposal identified by $P$.

### 2.2.8 Authorization for Declarations in the Proposal-Based Mechanism: Context-Bound Proposals

This section specifies the authorization conditions for the different classes of declarative speech-acts in the proposal-based mechanism in the case of context-bound proposals. The authorization conditions for voting on a context-bound proposal or announcing that the group has accepted or rejected a context-bound proposal are completely analogous to their context-free counterparts. In contrast, the authorization conditions for *initiating* a context-bound proposal include that the coordinated-cultivation requirement holds for the intention that the context-bound proposal concerns; thus, the authorization conditions refer to a past declaration that established the coordinated-cultivation requirement for that intention. As mentioned previously, carrying the arguments from the context-*free* proposal that established the coordinated-cultivation requirement for that intention enables the authorization conditions to avoid using existential quantification.

- **Context-Bound Proposal**: Any member of a group may make a context-bound proposal concerning a shared intention (identified by $I$) for which the coordinated-cultivation requirement holds. The following axiom says that an agent $G_1'$ is authorized to make a context-bound proposal if and only if at some point in the past a possibly different agent $G_1$ made an authorized declaration establishing the group's commitment to that activity.

$$\begin{aligned} \models\ & auth(G_1', GR, \delta(G_1', GR, MadeCBP(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0))) \\ & \Leftrightarrow\ (G_1' \in GR)\ \wedge\ \Diamond^{\leftarrow} authDecl(G_1, GR, CCR^{+}(GR, I, \alpha, \Upsilon_0)), \end{aligned}$$

where $CCR^{+}(GR, I, \alpha, \Upsilon_0)$ is as defined in Section 2.2.4. Notice that without carrying along the arguments from the context-free proposal, this authorization condition would be difficult to state without existential quantification.

- **Vote to Accept or Reject**:

$$\begin{aligned} \models\ & auth(G_2, G_1', \delta(G_2, G_1', AcceptedCBP(G_2, G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0))) \\ & \Leftrightarrow\ V_1' \wedge V_2' \wedge V_3' \wedge V_4' \wedge V_5' \end{aligned}$$

where:

$$\begin{aligned} V_1' &\equiv \Diamond^{\leftarrow} MadeCBP(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0) \\ V_2' &\equiv (G_2 \neq G_1') \\ V_3' &\equiv \neg \Diamond^{\leftarrow}\ declared(G_2, G_1', AcceptedCBP(G_2, G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0)) \\ V_4' &\equiv \neg \Diamond^{\leftarrow}\ declared(G_2, G_1', RejectedCBP(G_2, G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0)) \\ V_5' &\equiv \neg \Diamond^{\leftarrow}\ declared(G_1', GR, GroupRejectedCBP(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0)). \end{aligned}$$

The authorization to vote to reject a proposal is the same.

- **Announce Group Acceptance**:

$$\begin{aligned} \models\ & \mathit{auth}(G_1', GR, \delta(G_1', GR, \mathit{GrAccCBP}^+(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0))) \\ & \Leftrightarrow\ A_1' \wedge A_2' \wedge A_3' \wedge A_4' \end{aligned}$$

where:

$$\begin{aligned} A_1' &\equiv \Diamond^{\leftarrow} \mathit{MadeCBP}(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0) \\ A_2' &\equiv \neg \Diamond^{\leftarrow}\ \mathit{declared}(G_1', GR, \mathit{GroupRejectedCBP}(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0)) \\ A_3' &\equiv \neg \Diamond^{\leftarrow}\ \mathit{declared}(G_1', GR, \mathit{GrAccCBP}^+(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0)) \\ A_4' &\equiv (\forall G \in GR, G \neq G_1')\ \Diamond^{\leftarrow} \mathit{AcceptedCBP}(G, G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0) \end{aligned}$$

- **Announce Group Rejection**:

$$\begin{aligned} \models\ & \mathit{auth}(G_1', GR, \delta(G_1', GR, \mathit{GroupRejectedCBP}(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0)) \\ & \Leftrightarrow\ A_1' \wedge A_2' \wedge A_3' \end{aligned}$$

### 2.2.9 Example: Using the Proposal-based Mechanism to Make a Group Decision to do an Intention-Update Operation

The example in Section 2.2.6 ended with the agent $G_1$'s authorized declaration establishing the group's decision to commit to doing $\alpha$ together:

$$\mathit{authDecl}(G_1, GR, \mathit{CCR}^+(GR, I_{64}, \alpha, \Upsilon_0)),$$

where

$$\mathit{CCR}^+(GR, I_{64}, \alpha, \Upsilon_0) = \begin{cases} \Box\, CCR(GR, I_{64}) \\ \wedge \\ (\forall G \in GR)\ \mathcal{O}(G, GR, \mathit{AdoptInt}(I_{64}, \alpha, GR, \Upsilon)) \end{cases}.$$

**A Context-Bound Proposal.** As a result of the above declaration, each agent $G \in GR$ is now authorized to make context-bound proposals concerning candidate intention-update operations for the newly established intention (identified by $I_{64}$). For example, an agent $G_1'$ is authorized to propose that the group bind the parameter ?`K` to the value `Car`$_{39}$:

$$\mathit{auth}(G_1', GR, \mathit{MadeCBP}(G_1', \texttt{BIND}(?\texttt{K}, \texttt{Car}_{39}), P_{15}, G_1, GR, I_{64}, \alpha, \Upsilon_0)).$$

Suppose $G_1'$ makes such a proposal.

$$\mathit{declared}(G_1', GR, \mathit{MadeCBP}(G_1', \texttt{BIND}(?\texttt{K}, \texttt{Car}_{39}), P_{15}, G_1, GR, I_{64}, \alpha, \Upsilon_0)).$$

Since it is an authorized proposal, it establishes its propositional content:

$$\mathit{MadeCBP}(G_1', \texttt{BIND}(?\texttt{K}, \texttt{Car}_{39}), P_{15}, G_1, GR, I_{64}, \alpha, \Upsilon_0).$$

**Voting on the Proposal.** The above proposal authorizes the rest of the agents in the group to vote on it. Suppose that each of them votes to accept the proposal:

$$(\forall G \in GR, G \neq G'_1)$$
$$declared(G, G'_1, AcceptedCBP(G, G'_1, \texttt{BIND}(?\texttt{K}, \texttt{Car}_{39}), P_{15}, G_1, GR, I_{64}, \alpha, \Upsilon_0)).$$

Since these are authorized declarations, their propositional content is thereby established:

$$(\forall G \in GR, G \neq G'_1)\ AcceptedCBP(G, G'_1, \texttt{BIND}(?\texttt{K}, \texttt{Car}_{39}), P_{15}, G_1, GR, I_{64}, \alpha, \Upsilon_0).$$

**The Announcement of a Group Decision.** As a result of the above, $G'_1$ is authorized to announce the group decision:

$$declared(G'_1, GR, GrAccCBP^+(G'_1, \texttt{BIND}(?\texttt{K}, \texttt{Car}_{39}), P_{15}, G_1, GR, I_{64}, \alpha, \Upsilon_0)).$$

Since this is an authorized declaration, its content

$$GrAccCBP^+(G'_1, \texttt{BIND}(?\texttt{K}, \texttt{Car}_{39}), P_{15}, G_1, GR, I_{64}, \alpha, \Upsilon_0),$$

which is an abbreviation for

$$\begin{cases} (\forall G \in GR)\ \mathcal{O}(G, GR, Update(I_{64}, \texttt{BIND}(?\texttt{K}, \texttt{Car}_{39}))) \\ \wedge\ GroupAcceptedCBP(G'_1, \texttt{BIND}(?\texttt{K}, \texttt{Car}_{39}), P_{15}, G_1, GR, I_{64}, \alpha, \Upsilon_0), \end{cases}$$

is established. Hence, the agents in the group are obliged to update their intentions identified by $I_{64}$ to reflect that the parameter $?\texttt{K}$ has been bound to the value $\texttt{Car}_{39}$.

## 2.3 A Formal Analysis of the Proposal-Based Mechanism

This section states and proves several important properties of the proposal-based mechanism. The main result says that, under the assumptions about agent beliefs given below, an announcement of a group decision happens only if duly authorized by the group.

### 2.3.1 Assumptions about Agent Beliefs

The analysis in this section assumes a standard KD45 modal belief operator (Chellas, 1980). All axioms presented in this section are assumed to be common knowledge.

**Axiom 2.4 (Declarations and Mutual Belief)** *The recipients of a declaration have correct and complete mutual belief about the occurrence (or non-occurrence) of that declaration:*

$$\models declared(G, GR, \phi) \ \Leftrightarrow\ MB(GR, declared(G, GR, \phi))$$
$$\models \neg declared(G, GR, \phi) \ \Leftrightarrow\ MB(GR, \neg declared(G, GR, \phi))$$

*In addition, their mutual beliefs are correct and complete concerning declarations made in the past:*

$$\models\ \Diamond^{\leftarrow} declared(G, GR, \phi) \ \Leftrightarrow\ MB(GR, \Diamond^{\leftarrow} declared(G, GR, \phi))$$
$$\models\ \neg\Diamond^{\leftarrow} declared(G, GR, \phi) \ \Leftrightarrow\ MB(GR, \neg\Diamond^{\leftarrow} declared(G, GR, \phi))$$

Recall that the speaker of a declaration is one of the recipients and, thus, participates in the mutual beliefs.

**Axiom 2.5 (Sincerity)** *Agents only make declarations they believe they are authorized to make.*

$$\models \; declared(G, GR, \phi) \;\Rightarrow\; \mathcal{P}Bel(G, auth(G, GR, \delta(G, GR, \phi)))$$

The purpose of the following axiom is to preclude the need for explicit sets of agents in the syntax. For example, $GR$ is used instead of an explicit set such as: $\{G_1, G_2, \ldots, G_n\}$.

**Axiom 2.6 (Agents, Groups, Identity)** *Every agent $G$ has correct and complete beliefs about whether any other agent $G'$ is a member of the group $GR$, and about its own identity. Furthermore, these propositions are timeless.*

$$\begin{array}{l}
\models \; Bel(G, (G' \in GR)) \Leftrightarrow (G' \in GR) \\
\models \; Bel(G, (G \neq G')) \Leftrightarrow (G \neq G') \\
\models \; (G' \in GR) \Leftrightarrow \Diamond^{\leftarrow}(G' \in GR) \\
\models \; (G \neq G') \Leftrightarrow \Diamond^{\leftarrow}(G \neq G') \\
\models \; Bel(G, (\forall G' \in GR, G' \neq G'')\; \phi) \;\Leftrightarrow (\forall G' \in GR, G' \neq G'')\; Bel(G, \phi)
\end{array}$$

**Axiom 2.7 (Beliefs about the Past)** *If an agent $G$ believes that at some point in the past $\phi$ held, then at some point in the past $G$ believed that $\phi$ currently held:*

$$\begin{array}{l}
\models \; Bel(G, \Diamond^{\leftarrow}\phi) \Rightarrow \Diamond^{\leftarrow} Bel(G, \phi) \\
\models \; Bel(G, \mathcal{P}\phi) \Rightarrow \mathcal{P}Bel(G, \phi).
\end{array}$$

Axiom 2.7 is a strong assumption that allows us to avoid complex issues of belief revision. It is equivalent to

$$\models \; \neg\Diamond^{\leftarrow} Bel(G, \phi) \Rightarrow \neg Bel(G, \Diamond^{\leftarrow}\phi),$$

which says that if at no point in the past did $G$ believe that $\phi$ currently held, then $G$ does not currently believe that $\phi$ ever held in the past.

### 2.3.2 An Analysis of Context-Free Proposals

This section presents a formal analysis of the sample, proposal-based group decision-making mechanism in the case of context-free proposals. The analysis culminates with Theorem 2.13 which says that if an agent $G$ believes that some other agent $G_1$ declared that the group $GR$ accepted a context-free proposal, then $G_1$ was, in fact, so authorized, and did make such a declaration. This result says that, given the assumptions in Section 2.3.1, agents can trust their beliefs about agent declarations in the proposal-based group decision-making mechanism.

To simplify the presentation, the proofs have been placed in Appendix A. Section 2.3.3 extends these results to context-bound proposals.

**Theorem 2.8** *If an agent G believes that a possibly different agent $G_1$ made an authorized declaration of a context-free proposal at some point in the past, then $G_1$ did, in fact, do so:*

$$\begin{aligned} \models \quad & \mathit{Bel}(G,\ \Diamond^{\leftarrow} \mathit{authDecl}(G_1, GR, \mathit{MadeCFP}(G_1, GR, I, \alpha, \Upsilon_0))) \\ & \Rightarrow\ \Diamond^{\leftarrow} \mathit{authDecl}(G_1, GR, \mathit{MadeCFP}(G_1, GR, I, \alpha, \Upsilon_0)). \end{aligned}$$

**Theorem 2.9** *If an agent G believes it is authorized to vote either to accept or reject a context-free proposal, then G is, in fact, so authorized:*

$$\models \quad \mathit{Bel}(G, \mathit{auth}(G, G_1,\ \delta(G, G_1, \xi))) \ \Rightarrow\ \mathit{auth}(G, G_1,\ \delta(G, G_1, \xi)),$$

*where $\xi$ is $\mathit{AcceptedCFP}(G, G_1, GR, I, \alpha, \Upsilon_0)$ or $\mathit{RejectedCFP}(G, G_1, GR, I, \alpha, \Upsilon_0)$. Similarly, if the originator $G_1$ of the proposal believes G is authorized to vote on that proposal, then G is, in fact, so authorized.*

**Theorem 2.10** *If an agent $G_1$ believes that another agent G accepted a context-free proposal originated by $G_1$, then G did, in fact, accept that proposal:*

$$\begin{aligned} \models \quad & \mathit{Bel}(G_1, \mathit{AcceptedCFP}(G, G_1, GR, I, \alpha, \Upsilon_0)) \\ & \Rightarrow\ \mathit{AcceptedCFP}(G, G_1, GR, I, \alpha, \Upsilon_0). \end{aligned}$$

**Theorem 2.11** *If an agent $G_1$ believes that it is authorized to declare, on behalf of the group, that the group has accepted a context-free proposal, then $G_1$ is, in fact, so authorized:*

$$\begin{aligned} \models \quad & \mathit{Bel}(G_1, \mathit{auth}(G_1, GR,\ \delta(G_1, GR, CCR^+(GR, I, \alpha, \Upsilon_0)))) \\ & \Rightarrow\ \mathit{auth}(G_1, GR,\ \delta(G_1, GR, CCR^+(GR, I, \alpha, \Upsilon_0))). \end{aligned}$$

**Theorem 2.12** *If an agent $G_1$ declares, on behalf of the group, that the group has accepted a context-free proposal, then that agent was, in fact, authorized to make that declaration:*

$$\begin{aligned} \models \quad & \mathit{declared}(G_1, GR,\ CCR^+(GR, I, \alpha, \Upsilon_0)) \\ & \Rightarrow\ \mathit{auth}(G_1, GR,\ \delta(G_1, GR, CCR^+(GR, I, \alpha, \Upsilon_0))). \end{aligned}$$

*Hence,*

$$\begin{aligned} \models \quad & \mathit{declared}(G_1, GR,\ CCR^+(GR, I, \alpha, \Upsilon_0)) \\ & \Rightarrow\ \mathit{authDecl}(G_1, GR,\ CCR^+(GR, I, \alpha, \Upsilon_0)). \end{aligned}$$

**Theorem 2.13** *If an agent G believes that a possibly different agent $G_1$ declared that the group $GR$ accepted a context-free proposal, then $G_1$ was, in fact, authorized to make such a declaration and did so:*

$$\begin{aligned} \models \quad & \mathit{Bel}(G, \Diamond^{\leftarrow} \mathit{declared}(G_1, GR,\ CCR^+(GR, I, \alpha, \Upsilon_0))) \\ & \Rightarrow\ \Diamond^{\leftarrow} \mathit{authDecl}(G_1, GR,\ CCR^+(GR, I, \alpha, \Upsilon_0)). \end{aligned}$$

### 2.3.3 Extending the Analysis to Context-Bound Proposals

The difference between group decisions based on context-free proposals and those based on context-bound proposals is primarily in their content. The authorization conditions for the various declarations (making a proposal, voting on a proposal, announcing that the group has rejected or accepted a proposal) are very similar. As a result, the sequence of theorems presented in the formal analysis of context-free proposals (i.e., Theorems 2.8–2.13) may, with some minor adjustments, be carried over to a formal analysis of context-bound proposals.

The primary difference between the authorization conditions for declarations concerning context-free and context-bound proposals is in the authorization conditions for *initiating* a proposal. In particular, the authorization conditions for a context-bound proposal concerning some group activity make reference to the authorized declaration that established the coordinated-cultivation requirement for that activity. Therefore, the primary modifications to the sequence of theorems for the analysis of context-free proposals are restricted to the theorem that addresses the authorization conditions for making a proposal (i.e., Theorem 2.8). The rest of the theorems carry over with a straightforward translation of declarations. For example, the declaration of a vote to accept a context-free proposal is translated into a declaration of a vote to accept a context-bound proposal.

Since the first theorem in the sequence is the only one requiring non-trivial modifications, it is the only one addressed in this chapter. Theorem 2.14 is the modified version of Theorem 2.8. Its proof is given in Appendix A.

**Theorem 2.14** *If an agent $G$ believes that a possibly different agent $G_1'$ made an authorized declaration of a context-bound proposal at some point in the past, then $G_1'$ did, in fact, do so:*

$$\begin{aligned}\models\;& Bel(G, \diamond^{\leftarrow} authDecl(G_1', GR, MadeCBP(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0)))\\ &\Rightarrow\; \diamond^{\leftarrow} authDecl(G_1', GR, MadeCBP(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0)).\end{aligned}$$

Given Theorem 2.14 as a starting point, modifying the rest of the theorems (and their proofs) to make them applicable to the context-bound case is trivial. The final theorem in the sequence states that an agent's beliefs about a declaration of another agent announcing a group's acceptance of a context-bound proposal are reliable. Thus, as in the context-free case, agents may trust their beliefs about announced group decisions concerning context-bound proposals.

## 2.4 The Coordinated-Cultivation Requirement

This section formally defines the coordinated-cultivation requirement (CCR) that prohibits collaborating agents from making unilateral intention-update decisions for their group-related intentions. The definition of the CCR is based on the treatment of conditional obligations, general prohibitions and explicit permissions by Brown (2000), illustrated by the following example: "You ought not to drive through an intersection while the traffic light is red unless explicitly permitted to do so (by a duly constituted authority)." The CCR stipulates that an agent ought not to update its shared intention unless explicitly permitted to do so by an authorized declaration of a group decision. Unlike Brown, the obligation operator $\mathcal{O}$ used in this thesis represents a *social obligation* that includes an argument specifying the group to which an agent is obliged.

**Definition 2.15 (Coordinated-Cultivation Requirement)** *The coordinated-cultivation requirement for a group $GR$ of agents and an intention identifier $I$ is represented by the formula $CCR(GR, I)$ which abbreviates the following:*

$$(\forall \Upsilon) \begin{cases} (\exists G_1 \in GR) \diamond^{\leftarrow} authDecl(G_1, GR, (\forall G \in GR)\ \mathcal{O}(G, GR, Update(I, \Upsilon))) \\ \vee \\ (\forall G \in GR)\ \mathcal{O}(G, GR, \neg Update(I, \Upsilon)). \end{cases}$$

This formula, in which the universally quantified variable $\Upsilon$ represents an arbitrary set of intention-update operations, says that unless some member of the group made an authorized declaration establishing that each member $G$ of the group $GR$ is explicitly obliged to do the set $\Upsilon$ of intention-update operations, then each member of the group is obliged *not* to do them

This definition of the coordinated-cultivation requirement employs universal and existential quantification, which are not provided for in the current formulation of the logic DDLTLB used in this thesis. Extending the semantics of DDLTLB to provide for universal and existential quantification is beyond the scope of this thesis. However, the formal analysis of the sample, proposal-based mechanism presented in Section 2.3 is independent of the definition of the coordinated-cultivation requirement.

In practice, an agent can maintain a list of explicitly authorized declarations relevant to the CCR. For its intention identified by $I$, the agent is obliged to do all of the intention-update operations authorized and obliged by such explicit declarations and no others.

## 2.5 Enabling an Agent's Participation

The definition of any group decision-making mechanism must specify:

- the legal inputs agents can make; and
- the ways of combining inputs from multiple agents to generate authorizations for various group decisions.

To participate in such a mechanism, an agent must be able to:

- generate candidates for its input to the mechanism; and
- determine whether a given candidate might lead to obligations that might conflict with its current commitments.

The candidate-testing problem arises because any input an agent makes to a mechanism might combine with the inputs of other agents to authorize a group decision and, hence, subject the agent to various obligations. An agent typically honors such obligations by modifying an existing intention or creating a new intention. However, a rational agent does not knowingly create conflicts among its intentions. Thus, an agent must be able to reason about the interaction between its set of pre-existing commitments and any obligations that might ensue from its participation in a group decision-making mechanism which, of course, leads to the well-known *ramification problem* (Russell and Norvig, 1995).

One way an agent might be able to ease its computational burden when reasoning about its participation in a group decision-making mechanism would be to limit the portion of its private knowledge base that such reasoning takes into consideration. For example, when considering whether to support a proposal that our group rent a '62 Chevy or a proposal that we rent an '86 Ford Bronco, I might only need to consider things like gas mileage and cargo space. This thesis refers to the portion of an agent's knowledge base used by that agent when reasoning about its participation in a group decision-making mechanism as the agent's *mechanism-participation context* (MPC). This thesis refers to the problem of generating a suitable mechanism-participation context as the *MPC-Generation Problem*.

When generating a mechanism-participation context, an agent must ensure (as much as possible) that the generated MPC is in some sense *decoupled* from the rest of its knowledge base, relative to the reasoning problems associated with its participation in the given mechanism. For example, if I limit my reasoning about the '62 Chevy and the '86 Bronco to issues of gas mileage and cargo space, I'd better be sure that my knowledge of off-road routes is not relevant. One way to make sure of this would be to decide that I will not go off road between here and New York City; in fact, I might decide to stay on the Interstate the whole way. In general, the application of additional constraints is necessary to decouple an MPC from the rest of an agent's private knowledge base.

When a group decision is reached, implementing the decision may give rise to various coordination issues. For example, suppose we have decided that I shall buy the food for dinner and you shall buy the wine. Since certain combinations of food and wine are more pleasing than others, my choice of food and your choice of wine are not independent. We could coordinate our choices in many ways. For example, we might negotiate by cell phone as we head to our different stores. Alternatively, we might decide to act independently, each making the choice that is least likely to offend the palette given our uncertainty about what the other will buy. We refer to the problem of coordinating agent activity after a group decision has been implemented as the *post-decision coordination problem*.

In summary, to facilitate their participation in group decision-making mechanisms, agents must be able to deal with the following problems:

- The candidate-generation problem
- The candidate-testing problem
- The MPC-generation problem

- The post-decision coordination problem

The first three of these problems are single-agent problems; the last is a multi-agent coordination problem.

The next chapter focuses on the problem of how agents can determine, both individually and collectively, whether to commit to some proposed group activity, that is, the Initial-Commitment Decision Problem (ICDP). It introduces a group decision-making mechanism based on a combinatorial auction that agents can use to solve instances of the ICDP. Each of the problems listed above has a more specific counterpart for the ICDP mechanism. Subsequent chapters provide algorithms that agents can use to solve such problems and, thus, to participate effectively in the auction-based ICDP mechanism.

# Chapter 3

# A Mechanism for the Initial-Commitment Decision Problem

When rational, autonomous agents encounter an opportunity to collaborate on some proposed group activity, they must decide, both individually and collectively, whether to commit to that activity. This thesis refers to this problem as the Initial-Commitment Decision Problem (ICDP). This chapter presents a group decision-making mechanism based on a *combinatorial auction* (Sandholm, 1999; Fujishima, Leyton-Brown, and Shoham, 1999; Rassenti, Smith, and Bulfin, 1982) that agents can use to solve instances of the ICDP. In this mechanism, called the ICDP mechanism, agents bid on sets of tasks in the proposed activity. Of central importance to the mechanism is that agents can make their bids dependent on temporal constraints, thereby enabling each agent to protect its private schedule of pre-existing commitments. In addition to solving instances of the ICDP, the ICDP mechanism can also be used to solve task-allocation problems in the context of multi-agent activities to which the group is already committed.

The rest of this chapter is organized as follows. Section 3.1 describes the Initial-Commitment Decision Problem. Section 3.2 briefly reviews standard combinatorial auctions. Section 3.3 presents a modified combinatorial auction in which agents bid on sets of roles in a proposed group activity. Section 3.4 specifies the combinatorial auction-based ICDP according to the GDMM framework from Chapter 2. Section 3.5 identifies several problems that individual agents must be able to solve to participate effectively in the ICDP mechanism. Section 3.6 discusses related work.

## 3.1 The Initial-Commitment Decision Problem

When rational, autonomous agents encounter an opportunity to collaborate on some group activity, they must decide, both individually and collectively, whether to commit to doing that activity. We refer to this problem as the Initial-Commitment Decision Problem. It is assumed that new opportunities for collaborative action arise in context: each agent may have pre-existing commitments to other individual and group activities. It is further assumed that agents are utility-maximizers and, thus, will only commit to group activities if their individual net profit is expected to be non-negative.

When evaluating the potential cost of participating in a proposed group activity, an agent must consider the context of its existing commitments. For example, the fact that I'm already planning to drive to the airport has a dramatic impact on my estimate of the cost of giving you a ride. Horty and Pollack (2001) initiated research into how an individual agent can evaluate new opportunities for single-agent action in the context of its pre-existing commitments. The ICDP is a generalization of that problem to the group context, which introduces two significant complications. First, no single agent has complete information about the pre-existing commitments of all of the agents in the group; in other words, the background context is distributed. Second, the approach (i.e., choice of method and distribution of tasks) that is best for the group may not be best for any single agent.

An agent participating in multi-agent planning incurs significant costs, including

- time and computational resources devoted to group decision-making processes;
- opportunity costs for commitments, not only to doing its share of tasks in the group activity, but also to supporting the actions of others; and
- costs of doing the actions to which it commits.

To decide whether to join a proposed collaboration, an agent needs to assess the potential impact of that collaboration on its ability to do other work. Since the planning, decision-making, and opportunity costs associated with collaborative activity can be substantial, it is preferable for agents to be able to determine some upper bound on that impact prior to committing to the group activity.

In deciding whether to commit to a new group activity, each agent must estimate two factors:

- the potential contributions it could make to the group activity (i.e., the constituent subacts it could do or participate in) and the costs of those contributions; and
- the possibilities for the remaining tasks to be assigned to other group members in an individually rational manner.

The first factor requires that agents examine information about their individual background contexts of commitments. Because agents may be unwilling or unable to share complete information about their individual contexts, this factor is best computed "locally" by individual agents. The second factor requires a global computation that takes into account the potential contributions of all of the agents.

The proposal-based mechanism presented in Chapter 2 can be used by agents to establish their initial commitment to some proposed group activity; however, that mechanism requires that some agent be able to generate a proposal that the rest of the agents in the group would be willing to accept. To enable agents to assess the impact of such a proposal on their individual schedules, such a proposal might need to specify the recipe for the activity, a mapping from tasks to agents, and a set of temporal constraints to coordinate the execution of the activity.[1] The generation of such a proposal would require (1) knowledge about the private schedules of the potential participants which may not be available to any single agent, and (2) the ability to perform a potentially intractable computation. As a result, it is not generally practical to use the proposal-based mechanism to solve instances of the ICDP.

In contrast, the ICDP mechanism presented in this chapter uses a combinatorial auction (Sandholm, 1999; Fujishima, Leyton-Brown, and Shoham, 1999; Rassenti, Smith, and Bulfin, 1982) to coordinate the sorts of local and global computations described above. Each potential contribution to the group activity is stated in terms of a locally-computed bid that specifies a set of tasks, a cost for doing those tasks, and a set of constraints on the execution times of those tasks. The global computation determines the best combination of such bids (i.e., potential contributions). It is based on an existing winner-determination algorithm for combinatorial auctions (Sandholm, 1999) that was modified to enable it to accommodate bids with temporal constraints. Notice that the ICDP mechanism not only establishes a group's commitment, but also generates a potential allocation of tasks to agents that is consistent with the private schedules of the participants.

By distributing the computational burden in this way, the ICDP mechanism allows agents to maintain the privacy of their pre-existing commitments and to focus their computational efforts on their own potential for contributing to the proposed collaboration. Furthermore, being able to condition bids on temporal constraints allows agents to protect the feasibility of their pre-existing commitments. In addition, although the global computation may be carried out centrally, it may also be carried out in a distributed fashion, with agents searching different portions of the highly-structured search space. Finally, to decide whether to commit to a proposed group activity typically does not require finding an optimal task allocation. Agents may decide to commit to an activity based on a provisional allocation of tasks that they may try to improve on later.

## 3.2 Combinatorial Auctions

In a *combinatorial auction* (Sandholm, 1999; Fujishima, Leyton-Brown, and Shoham, 1999; Rassenti, Smith, and Bulfin, 1982), there are multiple items for sale; there are participants who may place bids on arbitrary subsets of those items; and there is an auctioneer who must determine which *awardable* combination of bids maximizes revenue. Figure 3.1 shows a combinatorial auction in which there are four items—$A, B, C$, and $D$—for sale, and five participants who have made bids such as: "\$4 for $\{A, B\}$" and "\$9 for $\{A, B, D\}$."

In general, let $\mathbf{I} = \{I_1, I_2, \ldots, I_n\}$ be the set of $n$ items being auctioned, and let $\mathbf{B}$ be the set of submitted bids. For each bid $b \in \mathbf{B}$, let $\mathit{Items}(b) \subseteq \mathbf{I}$ denote the subset of items covered by the bid, and let $Amount(b)$ denote the amount of the bid. A *bid-set* (i.e., a collection of bids) is called

- *disjoint*, if each item being auctioned is covered by at *most* one bid in the set,

[1] Such an approach would be similar to that of Kinny et al. (1994), as discussed in Chapter 1.

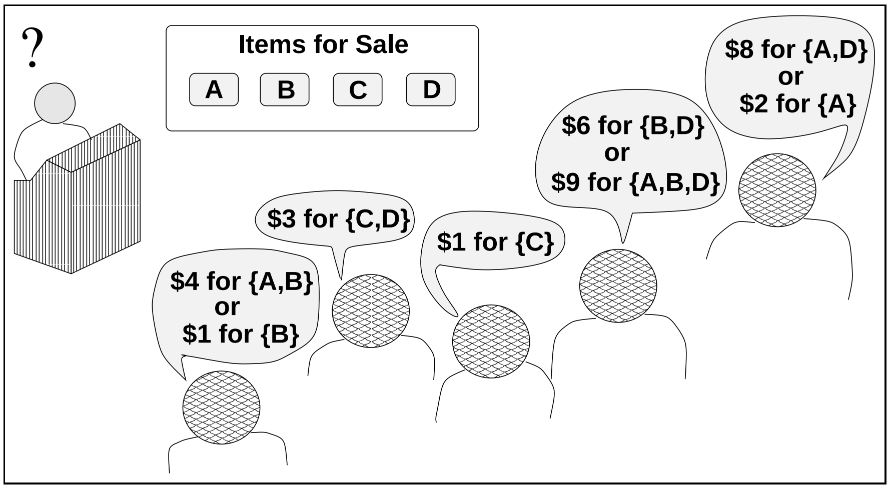



Figure 3.1: A combinatorial auction

- *covering*, if each item is covered by at *least* one bid in the set; and
- *awardable*, if it is both disjoint and covering.

An awardable bid-set is also called a *solution.* For any disjoint bid-set $BS$, the revenue that $BS$, if awarded, would generate for the auctioneer is

$$Value(BS) \quad = \quad \sum_{b \in BS} Amount(b).$$

The winning bid-set is an awardable bid-set that maximizes revenue:

$$BS^* \quad = \quad \underset{BS \,\in\, \mathcal{ABS}}{argmax} \; Value(BS),$$

where $\mathcal{ABS}$ is the set of awardable bid-sets. The revenue generated by the winning bid-set is: $Value(BS^*)$. A winning bid-set is also called an *optimal solution.*

The awardable bid-sets for the auction in Figure 3.1 are shown in Figure 3.2. The last two bid-sets shown in Figure 3.2 yield the maximum revenue (i.e., $\$10$); thus, either may be chosen to be the winning bid-set.

### 3.2.1 Existing Winner-Determination Algorithms

The general winner-determination (WD) problem for combinatorial auctions is $\mathcal{NP}$-complete (Sandholm, 1999; Fujishima, Leyton-Brown, and Shoham, 1999). However, Sandholm (1999) and Fujishima et al. (1999) have independently presented WD algorithms that scale to problems involving scores of items and thousands of bids. The main insight underlying these algorithms is that, in practice, bids necessarily only sparsely populate the space of possible bids; thus, the search for the winning bid-set may be restricted to the space of bid-sets composed of actual—not possible—bids.

$$\boxed{\$2 \text{ for } \{A\}} + \boxed{\$1 \text{ for } \{B\}} + \boxed{\$3 \text{ for } \{C, D\}} \quad \Rightarrow \quad \$6$$

$$\boxed{\$2 \text{ for } \{A\}} + \boxed{\$6 \text{ for } \{B, D\}} + \boxed{\$1 \text{ for } \{C\}} \quad \Rightarrow \quad \$9$$

$$\boxed{\$4 \text{ for } \{A, B\}} + \boxed{\$3 \text{ for } \{C, D\}} \quad \Rightarrow \quad \$7$$

$$\boxed{\$9 \text{ for } \{A, B, D\}} + \boxed{\$1 \text{ for } \{C\}} \quad \Rightarrow \quad \$10$$

$$\boxed{\$8 \text{ for } \{A, D\}} + \boxed{\$1 \text{ for } \{B\}} + \boxed{\$1 \text{ for } \{C\}} \quad \Rightarrow \quad \$10$$

Figure 3.2: The awardable bid-sets for the auction in Figure 3.1.

At the core of each algorithm is a depth-first search through the space of disjoint bid-sets. Along each depth-first path in the search, a disjoint bid-set is incrementally constructed by successively appending individual bids. The path may end in a covering, and hence awardable, bid-set or it may reach a dead-end with a non-covering bid-set for which there are no compatible bids to append. Each algorithm keeps track of the best (i.e., revenue-maximizing) covering bid-set found so far and, thus, may be used as an anytime algorithm. Each algorithm uses an item-indexing scheme to ensure that each disjoint bid-set is generated at most once during the search. Sandholm organizes the received bids into an auxiliary data structure (called a *bid-tree*) to enable efficient generation of bids that are compatible with the current (partial) bid-set. Fujishima et al. partition the received bids into bins, each of which corresponds to a subtree of Sandholm's bid-tree.[2] In addition, each algorithm uses the same A$^*$-admissible heuristic that estimates the revenue that items not yet covered by the bid-set might bring in; this heuristic speeds up the search by enabling pruning of partial bid-sets that are certain not to bring in as much revenue as the best solution found so far.

The differences between the approaches of Sandholm and Fujishima et al. lie mostly in their proposed improvements to their basic algorithms. This chapter does not address these improvements because each sacrifices optimality in auctions involving temporal constraints and, in the current application, as described below, agents use temporal constraints in their bids to protect the feasibility of their private schedules of pre-existing commitments.

## 3.3 A Modified Combinatorial Auction for the ICDP

The ICDP arises when a group of agents encounter an opportunity for collaborative activity and each agent must decide whether to participate in that activity.

[2]Since the bins are coarser than Sandholm's bid-tree, additional checks are required to avoid redundant search.

**Definition 3.1 (Opportunity for Group Activity)** *An* opportunity *for a complex, group activity is a specification of the following: (1) a complex act-type representing the group activity; (2) a set of temporal constraints governing the activity; and (3) a payment that a group would receive for doing the activity.*

In the presentation below, the following notation is used:

- $\alpha(Opp)$ denotes the act-type of the opportunity $Opp$;
- $\mathcal{C}(Opp)$ denotes the temporal constraints on time-points associated with the opportunity $Opp$; and
- $P(Opp)$ denotes the payment associated with the opportunity $Opp$.

Given an opportunity $Opp$, if a group of agents can find some way of doing the activity $\alpha(Opp)$, at a cost less than $P(Opp)$, while satisfying the temporal constraints in $\mathcal{C}(Opp)$, then they could profit from doing it. This section presents a modified combinatorial auction, called an ICDP auction, that agents can use to determine whether doing the activity specified in a given opportunity is feasible for them.[3] In an ICDP auction, agents bid on sets of *roles* in a proposed group activity.[4] Our use of roles is described in Section 3.3.1.

A novel feature of the ICDP auction is that agents are allowed to condition their bids on temporal constraints. Allowing temporal constraints in bids is essential in this application since agents must be able to protect the feasibility of their private schedules of pre-existing commitments without having to reveal the details of those commitments to other agents. Winner determination in the ICDP auction is handled by an algorithm based on Sandholm's WD algorithm, but modified to accommodate temporal constraints in bids, as described in Section 3.3.5.

### 3.3.1 Roles

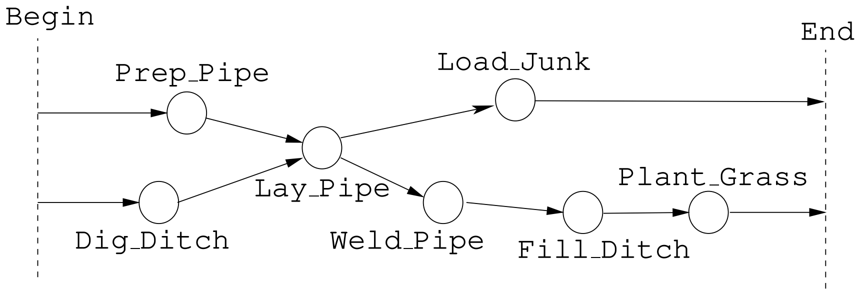
A role is a special type of parameter that is filled by an agent or subgroup of agents. The agent or subgroup filling a role is responsible for doing the subacts associated with that role. This chapter restricts attention to roles filled by single agents.

The value of clustering subacts into roles may be illustrated by an example: the representation of a transaction act-type in electronic commerce. No matter which protocol (or recipe) is used to govern the transaction, some tasks must be done by the buyer, others by the seller. In addition, various preconditions, postconditions and application constraints may be succinctly stated in terms of the buyer and seller roles (e.g., the seller must own the object being sold prior to the start of the transaction). However, despite the buyer and seller roles being naturally associated with the transaction act-type, the tasks covered by each role

[3] The same type of auction may also be used to solve task-allocation problems when a group is already committed to some activity.

[4] As will be seen, having agents bid on roles, rather than subacts, reduces the number of items up for bid and, thus, enables the auction to scale to problems involving larger numbers of subacts. It may also reduce the computational burden of bid generation, since once an agent discovers that it is unable to carry out one of the subacts covered by some role, it may immediately move on to considering other roles instead.

Recipe: `R39 (LAY_PIPELINE)`




Subacts with Precedence Constraints

Additional Constraints

$$
\begin{aligned}
Agent(\texttt{Prep\_Pipe}) &= Agent(\texttt{Weld\_Pipe}) \\
Agent(\texttt{Lay\_Pipe}) &= Agent(\texttt{Load\_Junk}) \\
Agent(\texttt{Dig\_Ditch}) &= Agent(\texttt{Fill\_Ditch})
\end{aligned}
$$

Figure 3.3: A sample recipe

may be determined by the protocol chosen to govern the transaction. For example, in one protocol, the buyer might have to list the objects being purchased, while in another the buyer might have to go through a complex set of identification-verification steps. Beyond specifying the tasks to be covered by the act-type roles, some protocols may introduce additional, protocol-specific roles. For example, one protocol might require an additional monitor role, responsible for carrying out a variety of transaction-monitoring tasks.

This chapter extends the representation of act-types and recipes to include *roles.* The roles associated with an act-type $\alpha$ are denoted by $\texttt{ActTypeRoles}(\alpha)$; the roles associated with a recipe $R$ are denoted by $\texttt{RecipeRoles}(R)$. The recipe must specify the set of subacts covered by each act-type role and each recipe role. Each subact must be covered by exactly one role. The agent filling a role is responsible for doing all of the subacts covered by that role.

Figure 3.3 repeats the sample recipe `R39` (for the `LAY_PIPELINE` act-type) seen previously in Figure 1.1. Figure 3.4 shows that same recipe (and act-type) modified to incorporate roles. The modified `LAY_PIPELINE` act-type specifies three roles: `DIGGER`, `LOADER` and `WELDER`. For each role, the modified recipe `R39` specifies the set of subacts covered by that role. For example, the agent filling the `LOADER` role is responsible for doing the `Lay_Pipe` and `Load_Junk` subacts. The modified recipe also includes an additional, recipe-specific role that arises from this recipe's particular way of subdividing the `LAY_PIPELINE` activity. The modified recipe specifies the responsibilities of this

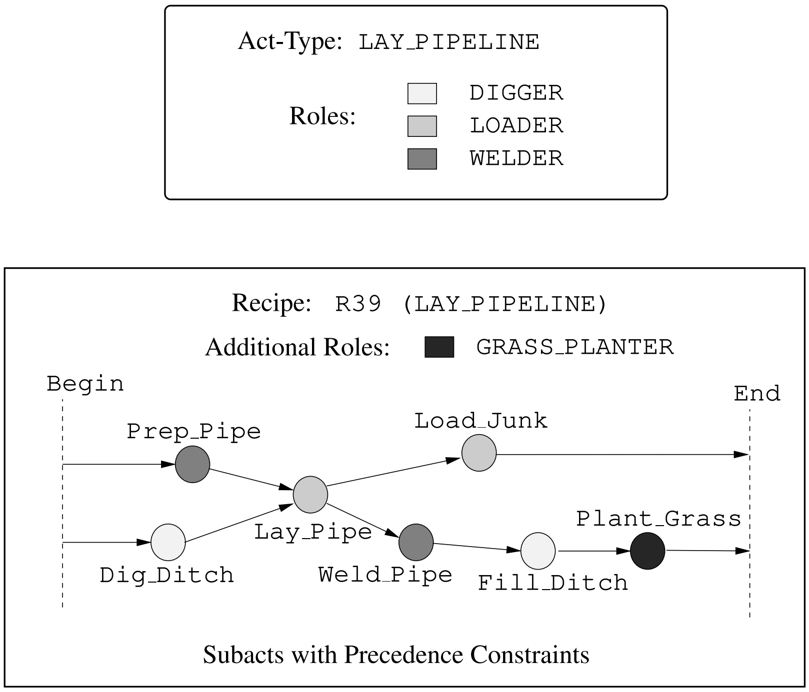



Figure 3.4: Act-type and recipe with roles

additional role: the agent filling the GRASS_PLANTER role is responsible for doing the Plant_Grass subact. Notice that the "additional constraints" in Figure 3.3 are implicit in the role specifications in the modified recipe.

Roles can reduce the computational burden in two ways. First, if a particular agent finds that it is unable to do one of the subacts covered by some role, then that agent may immediately move on to considering other roles instead. Second, instead of needing to identify agents to do each of the subacts, the group need only identify agents to fill the various act-type and recipe roles; if there are many fewer roles than subacts, then there are fewer decisions to make.

### 3.3.2 Dealing with Multiple Recipes

The items up for bid in an ICDP auction are the roles in the proposed group activity. Since an agent's costs for covering a set of roles may be quite different depending on the choice of recipe, and since each recipe may include a different set of additional, recipe-specific roles, each ICDP auction specifies a particular recipe. If there are multiple recipes available for a given act-type, agents may hold multiple, simultaneous ICDP auctions—possibly using different agents to carry out the auctioneer function for different auctions, thereby distributing the overhead of multiple winner-determination computations.

| | |
|---|---|
| Roles | {WELDER, GRASS_PLANTER} |
| Amount | \$300 |
| Temporal Constraints | $Begin(\alpha) \geq$ 3 : 30 p.m.<br>$End(\alpha) \leq$ 7 : 30 p.m.<br>$End$(Prep_Pipe) $-$ $Begin$(Weld_Pipe) $\leq$ 2 hours<br>$Duration$(Plant_Grass) $\geq$ 1 hour |

Figure 3.5: A sample bid pertaining to the recipe from Figure 3.4

### 3.3.3 Bids in an ICDP Auction

Figure 3.5 shows a sample bid for an ICDP auction for the LAY_PIPELINE activity using the R39 recipe from Figure 3.4. In this bid, the bidder proposes to do the WELDER act-type role and the GRASS_PLANTER recipe role for a payment of \$300 under the conditions that the group activity occur between 3:30 and 7:30 p.m., that the pipe-preparation subact end no more than two hours after the pipe-welding subact begins, and that the duration of the grass-planting subact be at least one hour.

### 3.3.4 Specification of an ICDP Auction

**Items up for Bid.** For an auction corresponding to an act-type $\alpha$ and a recipe $R$, the items up for bid are given by

$$\mathbf{I} \;=\; \texttt{ActTypeRoles}(\alpha) \,\cup\, \texttt{RecipeRoles}(R).$$

**Bids.** Each bid must specify a set of roles, an amount of money, and a set of temporal constraints. For each bid $b$,

- $Roles(b) \subseteq \mathbf{I}$ denotes the set of roles covered by the bid;
- $Amount(b)$ denotes the amount of the bid;
- $TemporalConstraints(b)$ denotes the temporal constraints in the bid, which must be of the form, $t_j - t_i \leq \delta$, where $t_i$ and $t_j$ are time-points associated with the activity (e.g., the beginning or ending points of subacts) and $\delta$ is a real number.[5]

**Awardable Bid-Sets.** Disjoint and covering bid-sets are defined as for a standard combinatorial auction; however, the awardable bid-sets are defined differently. A bid-set $BS$ is called *awardable* (with respect to recipe $R$) if, in addition to its being disjoint and covering, there exists a set of execution times for the subacts in $R$ that satisfy all of the temporal

[5]This form of temporal constraint is that found in Simple Temporal Networks (Dechter, Meiri, and Pearl, 1991), as described in detail in Chapter 4.

constraints deriving from (1) the act-type $\alpha$, (2) the recipe $R$, (3) the opportunity, and (4) the bids in $BS$.[6] The *cost* of an awardable bid-set is given by:

$$Cost(BS) \;=\; \sum_{b \in BS} Amount(b).$$

**The Winning Bid-Set.** Let $\mathcal{ABS}$ be the set of awardable bid-sets. The winning bid-set is given by

$$BS^* \;=\; \underset{BS\,\in\,\mathcal{ABS}}{argmin}\;\; Cost(BS).$$

The cost of the winning bid-set is: $Cost(BS^*)$.

### 3.3.5 The Modified WD Algorithm

To determine the winner of a combinatorial auction involving bids that include temporal constraints, we modified Sandholm's basic winner-determination (WD) algorithm (described in Section 3.2.1).[7] The modifications included adding a consistency check to the bid-set construction process and minimizing cost instead of maximizing revenue. The modified WD algorithm follows each depth-first path until one of the following occurs:

(1) the cost of the current bid-set is greater than that of the best bid-set found so far,

(2) the temporal constraints in the current bid-set are inconsistent with the temporal constraints associated with the proposed group activity,

(3) the current bid-set is not covering, but there are no compatible bids to append, or

(4) the current bid-set is awardable.

In the first three cases, the current-bid set is pruned; in the last case, the current bid-set becomes the best found so far.

#### Consistency Check

Determining whether the temporal constraints associated with the current set of bids are consistent with the temporal constraints associated with the proposed group activity is handled by keeping track of all of the relevant temporal constraints in a Simple Temporal Network (Dechter, Meiri, and Pearl, 1991). The number of time-points in the network depends only on the number of subacts in the recipe and is thus fixed. The number of temporal constraints is no more than $N(N-1)$, where $N$ is the number of time-points. Whenever a new bid is appended to the current bid-set, the temporal constraints from that bid are added

[6] This satisfiability condition is easily represented using Simple Temporal Networks (Dechter, Meiri, and Pearl, 1991), as described in detail in Chapters 4 and 5.

[7] Sandholm's algorithm was chosen primarily because the bid-tree structure used in that algorithm ensures non-redundant search.

to the network and their effects propagated throughout the network. As will be seen in the next chapter, detecting the consistency of a Simple Temporal Network is easy. If the constraints from the extended bid-set do not make the network inconsistent, the search may continue; otherwise, the search must backtrack.

### Experimental Evaluation of the Modified WD Algorithm

This section evaluates various characteristics of the performance of the modified WD algorithm. The first experiment quantifies the performance improvement that arises from using roles to bundle subacts. The remaining experiments clarify the tradeoffs that arise from allowing bids to include temporal constraints on the group activity. The inclusion of temporal constraints in bids results in fewer consistent bid combinations, which tends to increase the cost of an optimal solution (i.e., a least-cost awardable bid-set); in extreme cases, temporal constraints might result in there being no solution at all. This section also examines how the number of bids affects the relationship between the temporal constraints in bids and the likelihood of their yielding a solution.

In each experiment, the group action was constrained to occur within the time interval $[0, 100]$. Recipes were generated randomly such that in each recipe: (1) the number of precedence constraints was the same as the number of subacts, and (2) subacts were assigned to roles with equal probability. The bid-generation process was simulated by randomly generating bids according to various parameters (described separately for each experiment). The cost of a bid was randomly generated such that the cost-per-subact was uniformly distributed between $10$ and $20$.

The experiments described in this section included only unary temporal constraints in bids. In addition, the recipes and bids for each run were randomly generated. Although the resulting distributions may be different from those encountered in the context of actual multi-agent activity, they were sufficient for the purposes of these experiments. Finally, since the experiments were intended only as a pilot study to demonstrate the feasibility of the approach and to illustrate various performance tradeoffs, the implementation of the winner-determination algorithm was not optimized.

**Experiment 1.** The goal of this experiment was to determine the improvement in performance generated by bundling subacts into roles. The number of subacts was fixed at 40; the number of roles covering those subacts was 6, 8 or 10 (data collected for each case). Because winner determination in combinatorial auctions is exponential in the number of items being auctioned (Sandholm, 1999), the performance of the algorithm was expected to improve as the number of roles decreased (i.e., as the number of subacts covered by each role increased). The results of this experiment are plotted in Figure 3.6. Each point in the plot shows the time (averaged over 40 runs)[8] required to reach a solution of a particular *relative cost*, where the relative cost of a solution is given by

[8]In one of the runs with 10 roles, there was no solution. For that case, the results were averaged over the remaining 39 runs.

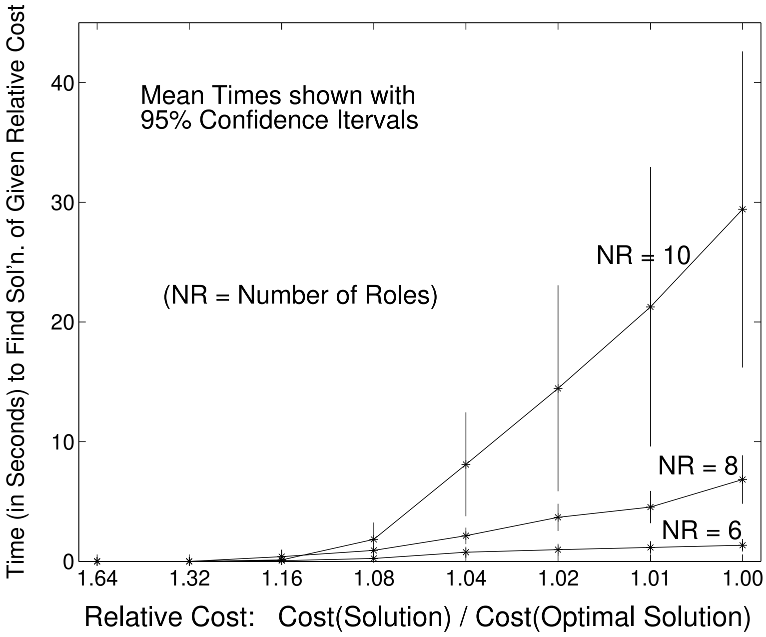



Figure 3.6: Varying the number of roles (NR)

$$Cost(\text{Solution}) \;/\; Cost(\text{Optimal Solution}).$$

Thus, the relative cost of the optimal solution is 1 (at the far right of the horizontal axis), whereas that of suboptimal solutions is greater than 1. There were 50 bids in each run, each bid covering 1 or 2 roles (uniformly distributed). Bids did not contain temporal constraints in this experiment.

The results clearly indicate a substantial improvement in performance as the number of roles covering the subacts decreases. If roles were not used, the number of biddable items would be the same as the number of subacts (i.e., 40), resulting in much poorer performance.

The results of this experiment also indicate that solutions with low relative costs are typically found within a relatively short period of time. For the Initial-Commitment Decision Problem, such near-optimal solutions may suffice. Once agents commit to the group activity, based on a near-optimal solution, they may use this solution as a baseline while continuing to search for lower-cost solutions.

**Experiment 2.** The goal of the second experiment was to show how the likelihood of a given set of bids yielding a solution depends both on the number of bids received and the number of constraints in each bid. In this experiment, each recipe contained 40 subacts covered by 10 roles. Each data point corresponds to 40 runs in which both the number of bids (NB) and the density of bid constraints (DBC) were fixed. A DBC value of 0 represents that the bid contained no subact-execution temporal constraints; a value of 1 represents that the bid constrained—with both lower and upper bounds—the execution

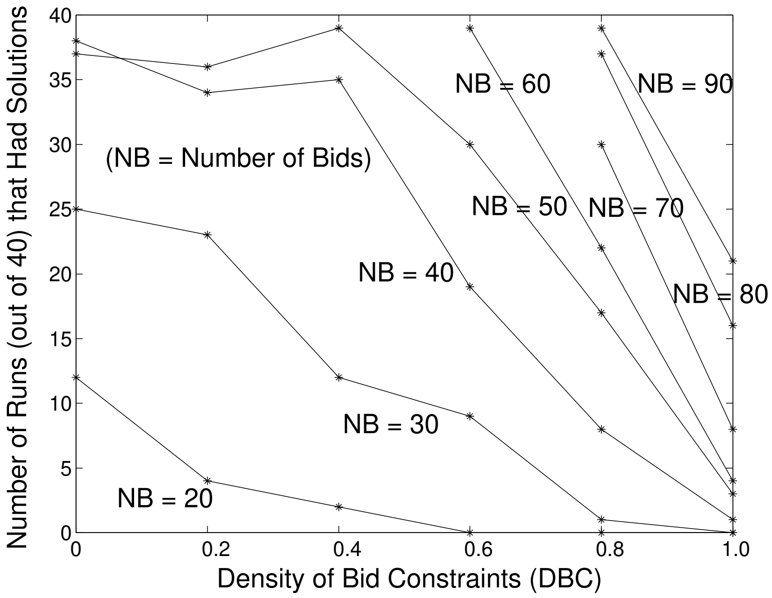



Figure 3.7: Varying the number of bids and the density of bid constraints

time of each subact covered by that bid's roles.[9] Each data point indicates the number of runs (out of 40) that yielded a solution. The results are plotted in Figure 3.7. They clearly show that the likelihood of the bids yielding a solution falls off sharply as the density of bid constraints increases. The results also indicate how many bids would be required to ensure a certain likelihood of finding a solution for a given density of bid constraints. Although these results do not directly apply to settings in which agents are focused on minimizing their own expected costs, they do provide guidance for agent design. Even in such settings, agents will want to identify solutions that are individually rational. These results indicate that the fewer constraints they place on their bids, the fewer bids they will need to submit, as a group, to find a solution.

**Experiment 3.** The goal of the third experiment was to determine how the density of bid constraints affects the cost of an optimal solution, the time required to find an optimal solution, and the time required to exhaust the search space (which is necessary to determine optimality).

The parameters were chosen such that solutions were generated in at least 90% of the runs for each DBC value. The number of subacts was 30, the number of roles was 8, the

[9] Temporal constraints in bids were determined as follows. First, in a random order, execution times consistent with the recipe's temporal constraints were selected for the subacts covered by the bid; for each subact $S$, a time point $T_s$ was randomly chosen, uniformly distributed between the subact's greatest lower bound (glb) and least upper bound (lub), and the effects of this assignment were propagated through the temporal network (see Chapter 4). Second, for each subact $S$, temporal constraints (simulating interactions with the agent's schedule of pre-existing commitments) of the form $Time(S) \leq T_s + 10$ and $Time(S) \geq T_s - 10$ were generated. Each temporal constraint was included in the bid with probability equal to the DBC.

number of bids was 75, and the number of roles covered by each bid ranged from 1 to 3 (uniformly distributed).

The results of the experiment are shown in Figure 3.8. Plot (a) shows that the cost of an optimal solution rises as the density of bid constraints rises; plot (b) shows that the time required to find an optimal solution or to exhaust the search space decreases as the density of bid constraints rises. In both cases, the reason is that the presence of bid constraints effectively shrinks the pool of awardable bid-sets. An optimal solution in the absence of constraints might become an inconsistent, and hence non-awardable, bid-set in the presence of constraints.

**Implications of the Results for the ICDP.** As mentioned previously, the ICDP requires only that agents determine, both individually and collectively, whether to commit to some proposed group activity; it does not require agents to commit to a particular allocation of tasks. Thus, when running an ICDP auction, agents typically do not need to find an *optimal* allocation of tasks; they need only find a *satisficing* allocation of tasks (Simon, 1969). After committing to the group activity, they may run additional auctions with the purpose of finding better task allocations.

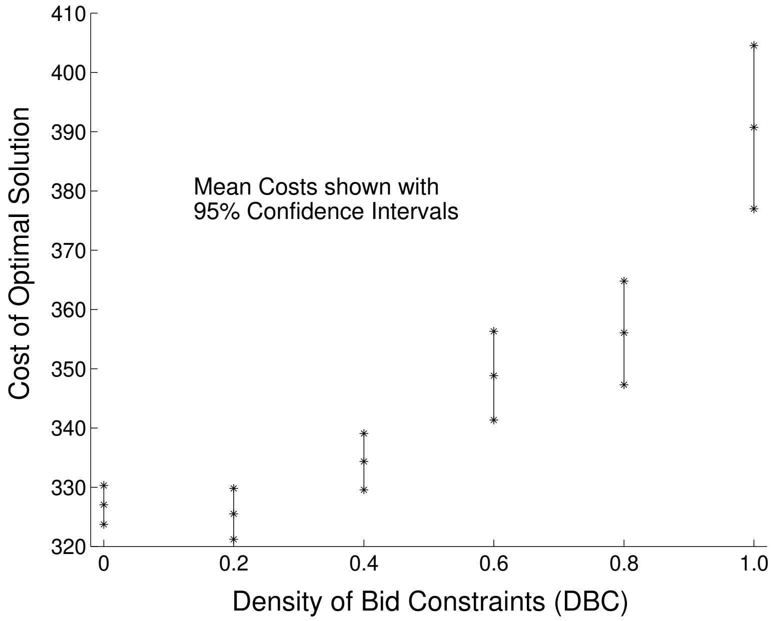
In view of these considerations, the results of these experiments suggest possible strategies that agents might employ when participating in an ICDP auction. For example, the individual contexts of the agents might be such that each agent could choose between making minimally-constrained, higher-cost bids or maximally-constrained, lower-cost bids. This would be the case if adding constraints could ensure that low-cost methods could be used to do various tasks. In early iterations, agents might be encouraged to generate maximally-constrained, lower-cost bids that could be examined quickly to determine whether they yielded a solution of sufficiently low cost. If not, agents could generate additional bids involving fewer constraints, thereby enlarging the pool of awardable bid-sets. As the pool of awardable bid-sets grows, the time to carry out an exhaustive search grows. However, the results of the first experiment (Figure 3.6) show that the modified WD algorithm tends to find near-optimal solutions very quickly; exhausting the search space tends to provide only a minimal reduction in solution cost. Thus, at each iteration, the modified WD algorithm could be run until a solution is found with cost below some threshold or until some time limit is reached.

## 3.4 The ICDP Group Decision-Making Mechanism

The ICDP auction described above is an example of a group decision-making mechanism. This section specifies the ICDP mechanism according to the GDMM framework presented in Chapter 2. The specification of the ICDP mechanism is similar to the specification of the proposal-based mechanism in the case of context-free proposals.

Specifying the ICDP mechanism according to the GDMM framework illustrates how that framework unifies group decision-making mechanisms for collaborative activity. Chapter 2 ended by listing several general problems that agents must solve to participate effectively in any group decision-making mechanism specified according to the GDMM frame-

(a) COST OF OPTIMAL SOLUTION




(b) TIME TO FIND OPTIMAL SOLUTION AND TO EXHAUST THE SEARCH SPACE

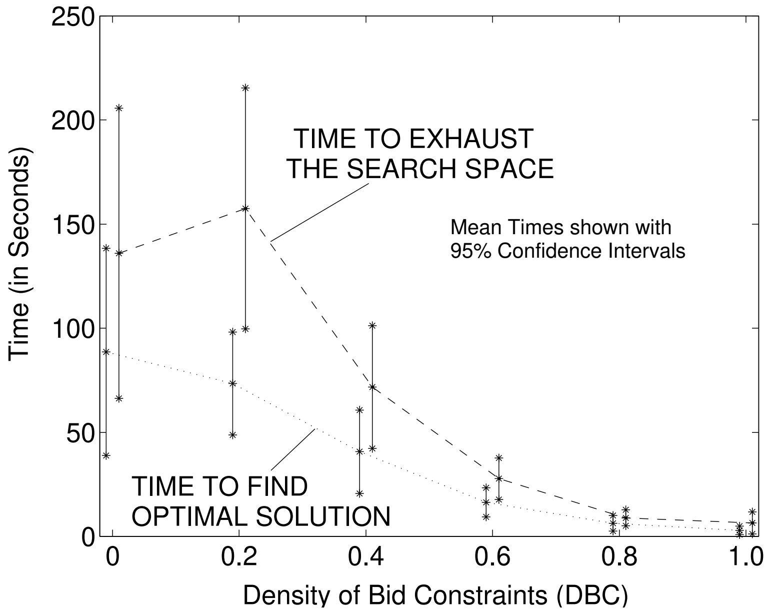



Figure 3.8: Plots from Experiment 3

work. This chapter ends by listing the more specific versions of those problems that are applicable to the ICDP mechanism. For example, agents must be able to generate temporal constraints for their bids to protect their private schedules of pre-existing commitments. In addition, they must be able to coordinate their post-auction activity, for example, by adding temporal constraints sufficient to decouple the tasks being done by different agents. Algorithms for solving the temporal-reasoning problems arising from the ICDP mechanism are presented in Chapter 5.

The specification of a group decision-making mechanism includes the classes of allowable content for the declarative speech-acts used in the mechanism and, for each such class, the conditions under which an agent is authorized to make declarations in that class.

**Allowable Content of Declarations**

There are three classes of allowable content for declarations in the ICDP mechanism:

- **Auction Invocation**: an agent proposes that the group hold an auction concerning a proposed group activity, using a specified recipe.
- **Bid**: an agent submits a bid to cover some of the roles in the group activity for a specified payment and subject to a specified set of temporal constraints.
- **Auction Results Announcement**: an agent announces the results of an auction that it invoked. The announcement may specify an awardable bid-set, if one exists, or the empty set, indicating no solution.

The declarations have the following forms, respectively:

- $\delta(G_1, GR, InvokedAuction(G_1, GR, I, Opp, R))$

  An agent $G_1$ proposes to the group $GR$ that they hold an auction concerning the opportunity $Opp$ (cf. Definition 3.1) using the recipe $R$. Should the group decide to commit to the proposed activity, the identifier $I$ will be used for the group-related intentions; it may also be used to identify the auction.

- $\delta(G_2, G_1, Bid(G_2, G_1, GR, I, Opp, R, B))$

  An agent $G_2$ submits a bid $B$ (cf. Section 3.3.4) to agent $G_1$ in the auction identified by $I$.

- $\delta(G_1, GR, AnnAucSoln(G_1, GR, I, Opp, R, BS))$

  An agent $G_1$ declares the results of the auction identified by $I$. $BS$ is either a set of bids or the empty-set.

**Authorizing Conditions**

The authorizing conditions for the three classes of declarative content for the ICDP mechanism are given below. Some of the conditions refer to metric temporal constraints. Since the logic DDLTLB (cf. Section 2.1.4) does not handle metric temporal constraints, no attempt is made to codify the various temporal consistency requirements in DDLTLB. Instead, the following predicates are used, each listed with its intended interpretation:

- $\mathit{WellFormedAuction}(\mathit{Opp}, R)$ represents that the opportunity $\mathit{Opp}$ is well-formed and compatible with the recipe $R$. In particular, $\mathit{WellFormedAuction}(\mathit{Opp}, R)$ holds only if $\alpha(\mathit{Opp})$ is a complex act-type, $P(\mathit{Opp})$ is an amount of money, $\mathcal{C}(\mathit{Opp})$ is a set of temporal constraints on the time-points defined by the act-type $\alpha$, and $R$ is a recipe for actions of type $\alpha(\mathit{Opp})$, such that the temporal constraints in the act-type $\alpha$, the set $\mathcal{C}(\mathit{Opp})$, and the recipe $R$ are mutually satisfiable.

- $\mathit{Awardable}(BS, \mathit{Opp}, R)$ represents that the bid-set $BS$ is awardable (i.e., $BS$ is disjoint and covering, and there exist execution times for the subacts in the recipe $R$ that satisfy all of the temporal constraints deriving from the act-type $\alpha(\mathit{Opp})$, the recipe $R$, the opportunity $\mathit{Opp}$, and the bids in $BS$).

One of the desired properties of the auction mechanism is that no more than one announcement of an auction solution be made by the agent that invoked the auction. Thus, the authorization conditions for an *announce-auction-solution* declaration must stipulate that no previous announcements were made for that auction. Such a condition is naturally expressed using existential quantification; however, the logic DDLTLB does not provide existential quantification.

To avoid the need for existential quantification in the authorizing conditions, a special predicate $\mathit{AnnSomeAucSoln}$ (for "announce some auction solution") is introduced. To achieve the desired effect, it is assumed that whenever a more-specific $\mathit{AnnAucSoln}$ declaration is made, a less-specific $\mathit{AnnSomeAucSoln}$ declaration is simultaneously made, as represented by the following axiom:

$$\begin{aligned}\models\;\; & \mathit{declared}(G_1, GR, \mathit{AnnAucSoln}(G_1, GR, I, \mathit{Opp}, R, BS)) \\ & \Rightarrow\; \mathit{declared}(G_1, GR, \mathit{AnnSomeAucSoln}(G_1, GR, I, \mathit{Opp}, R))\end{aligned}.$$

Notice that the more-specific $\mathit{AnnAucSoln}$ declaration contains a variable $BS$ that does not appear in the less-specific $\mathit{AnnSomeAucSoln}$ declaration.

- **Invoke an Auction**: Any member of the group is authorized to invoke an ICDP auction as long as it uses a recipe compatible with the well-formed opportunity.

$$\begin{aligned}\models\;\; & \mathit{auth}(G_1, GR, \delta(G_1, GR, \mathit{InvokedAuction}(G_1, GR, I, \mathit{Opp}, R))) \\ & \Leftrightarrow\;\; (G_1 \in GR) \;\wedge\; \mathit{WellFormedAuction}(\mathit{Opp}, R).\end{aligned}$$

- **Submit a Bid**: Any member of the group is authorized to submit a bid for an auction as long as the bid is well-formed and the agent that invoked the auction has not already announced the auction results.

$$\models \quad auth(G, G_1, \delta(G, G_1, Bid(G, G_1, GR, I, Opp, R, B)))$$
$$\Leftrightarrow \quad \begin{cases} \Diamond^{\leftarrow} InvokedAuction(G_1, GR, I, Opp, R) \\ \wedge \neg \Diamond^{\leftarrow}\ declared(G_1, GR, AnnSomeAucSoln(G_1, GR, I, Opp, R)) \end{cases}$$

- **Announce Auction Results**: The agent that invoked an auction is authorized to announce the results of that auction at most once. In the case of an announcement of no auction solution, there are no additional conditions; in the case of an announcement of an auction solution, the bid-set $BS$ must awardable. The case of no auction solution is indicated by the bid-set $BS$ being empty.

$$\models \quad auth(G, G_1, \delta(G, G_1, AnnAucSoln(G_1, GR, I, Opp, R, BS)))$$
$$\Leftrightarrow \quad \begin{cases} \neg \Diamond^{\leftarrow}\ declared(G_1, GR, AnnSomeAucSoln(G_1, GR, I, Opp, R)) \\ \wedge ((BS = \emptyset) \vee Awardable(BS, Opp, R)). \end{cases}$$

When the auction results in a solution, and thus the agents decide to commit to the proposed activity, there are two possibilities: they can decide to commit to the task allocation and temporal constraints determined by the bids in the bid-set $BS$ or not. Should they decide *not* to commit to the task allocation and temporal constraints determined by the bids, they will need to make task-allocation decisions later.

So far, the formal specification of the auction mechanism in the case of a successful auction is merely to establish that the special predicate $AnnAucSoln(G_1, GR, I, Opp, R, BS)$ holds for some awardable bid-set $BS$. It does not yet establish any obligations to adopt intentions toward the proposed activity.

If the desired result is that the agents commit to the new group activity *without* committing to the task allocation and temporal constraints determined by the bid-set $BS$, then an axiom of the form

$$\models \quad (BS \neq \emptyset) \ \wedge \ authDecl(G_1, GR, AnnAucSoln(G_1, GR, I, Opp, R, BS))$$
$$\Rightarrow \ authDecl(G_1, GR, CCR^{+}(GR, I, \alpha(Opp), \Upsilon(Opp, R)))$$

suffices, where $CCR^{+}$ is as defined in Section 2.2.4, $\alpha(Opp)$ is the act-type associated with the opportunity, and $\Upsilon(Opp, R)$ is an abbreviation for the following set of intention-update operations: to select the recipe $R$ for the group activity, and to add the temporal constraints in the set $\mathcal{C}(Opp)$.

On the other hand, if the desired result is that the agents also commit to the task allocation and temporal constraints determined by the bids in the set $BS$, then the above axiom need only change the set of intention-update operations to include the task allocation and temporal constraints corresponding to the bids in $BS$, which may be abbreviated by $\Upsilon(Opp, R, BS)$.

By formally specifying the ICDP mechanism according to the GDMM framework, properties of the ICDP mechanism may be formally analyzed using techniques similar to those used in the formal analysis of the proposal-based mechanism in Chapter 2. In addition, the special $Awardable$ predicate provides a clean interface between the DDLTLB logic and the algorithms based on Simple Temporal Networks provided in Chapter 5. Ultimately, it would be desirable for agents to be able to reason about metric temporal constraints within a logic like DDLTLB; however, providing such expressivity to DDLTLB is beyond the scope of this thesis.

Chapter 2 presented a straightforward set of modifications to the context-free version of the proposal-based mechanism to enable it to accommodate context-bound proposals. A similar set of modifications may be applied to the auction-based mechanism to enable it to be used as a task-allocation mechanism in the context of an activity to which a group is already committed. In that case, the result of a successful auction would be to establish the group's obligation to do the intention-update operations corresponding to the task allocation and temporal constraints determined by the awarded bids.

## 3.5 Participating Effectively in the ICDP Mechanism

To participate in the combinatorial auction-based ICDP mechanism, an agent must be able to construct bids for doing new tasks that are compatible with its existing commitments. To do so requires solving the following temporal reasoning problems:

- **Temporal-Constraint-Generation (TCG) Problem**: For any set of tasks that the agent might bid on, it must be able to generate a set of temporal constraints to protect its private schedule of pre-existing commitments in case the bid is ultimately awarded.

- **Auction-Participation-Context-Generation Problem**: To keep bid-generation computations manageable, an agent should restrict the portion of its schedule of pre-existing commitments that it uses as the basis for its bid-generation computations. That portion of its schedule, which must be temporally decoupled from the rest of its schedule, is called the agent's Auction-Participation Context (APC).

Furthermore, since there may be temporal dependencies among tasks awarded to different agents, a group of agents may need to coordinate their post-auction activity, a problem this thesis refers to as the Post-Auction Coordination (PAC) Problem. This thesis identifies several strategies agents can use to solve the Post-Auction Coordination Problem: (1) they can add new temporal constraints to ensure that the tasks being done by different agents are decoupled; (2) they can add a weaker set of temporal constraints to decouple the tasks being done by *some* of the agents; or (3) they can add constraints sufficient to form a hierarchy of temporal dependence relationships that gives the greatest flexibility to the agents that need it the most.

Algorithms that agents can use to solve each of these problems are provided in Chapter 5.

## 3.6 Related Work

The modified combinatorial auction described in this chapter was first presented in a conference paper (Hunsberger and Grosz, 2000). This section describes related work on winner-determination algorithms in combinatorial auctions, auction-based approaches to task allocation, and market-oriented solutions to combinatorial problems, as well as subsequent work on combinatorial auctions with temporal constraints.

#### Other Approaches to Winner-Determination in Combinatorial Auctions.

Fujishima et al. (1999) present a *Virtual Simultaneous Auction* as an alternative to their winner-determination algorithm for a standard combinatorial auction (described earlier, cf. Section 3.2.1). For each original bid, they create a virtual bidder that tries to secure the items in that bid. Their experimental results showed that this approach was competitive with their basic algorithm, although not quite as fast.

Andersson, Tenhunen and Ygge (2000) argue that formulating the winner-determination problem for standard combinatorial auctions as an integer-programming problem can lead to competitive algorithms using standard integer-programming packages.

Sandholm et al. (2001) have developed an improved winner-determination algorithm for standard combinatorial auctions. That algorithm, called CABOB, uses numerous tricks, including treating subproblems using integer-programming techniques, to optimize its performance. Determining whether CABOB can be usefully modified to accommodate temporal constraints, as in the WD algorithm for auctions in the ICDP mechanism, is left to future work.

#### An Auction-based Approach to Task Allocation

In Collins et al. (2000), a Customer agent selects a recipe and issues a call for bids from a set of Supplier agents. Each bid is on a set of tasks, and specifies not only a price for doing that set of tasks, but also a price for each task if awarded separately. Bids may constrain task execution times. The Customer agent uses a generalized simulated-annealing search with heuristics based on cost, risk, feasibility and task-coverage. The primary differences between their work and the approach taken in this chapter are as follows. First, they do not bundle tasks into roles and, thus, lose the corresponding computational advantage. Second, they allow the Customer agent to construct awardable bid-sets using *pieces* of bids; thus, a bid on $n$ items might be split into any one of $2^n$ pieces. Furthermore, the cost of any piece covering fewer than $n$ items is based on summing the individual costs of the corresponding subacts and thus ignores any sub-additivity or super-additivity relationships that might exist among them. In contrast, one of the main purposes of a combinatorial auction is to take advantage of such relationships. Finally, because their heuristic is based on a combination of risk, cost, feasibility and task-coverage, their algorithm may explore vast regions of the search space involving infeasible "solutions", whereas the approach taken in this chapter restricts the search to the space of feasible bid-sets.

#### Market-Oriented Solutions to Combinatorial Problems

Walsh and Wellman (1998), Kutanoglu and Wu (1997), and Fujishima et al. (1999) present market-oriented solutions to a range of combinatorial problems. Each approach involves iterative auctions. Walsh and Wellman (1998) present a decentralized, asynchronous protocol for allocating and scheduling tasks. In their approach, each good (representing either a physical resource or a service provided by some agent) is auctioned in a separate (M+1)$^{st}$-price auction. A separate agent is dedicated to the production of each single unit of a good. Auctions are run until they reach *quiescence* and conditions are given under which quiescence necessarily yields a valid solution. To handle time, Walsh and Wellman form (a subset of) the cross product of the discrete time line with the set of goods (resulting in a proliferation of goods, and hence auctions) and introduce special "bundling arbitrageur" agents responsible for procuring goods over various time intervals. The approach taken in this chapter handles temporal constraints without such an adverse computational impact.

Walsh, Wellman and Ygge (2000) subsequently investigated the use of a standard combinatorial auction in a multi-level supply-chain problem in which various agents submit bids either to produce or to consume various goods. The goal of the auction-based approach was to find a set of producer and consumer agents willing to form a supply-chain in which a specified set of goods could be produced. They found that the auction-based approach invariably leads to optimal allocations when agents bid their true valuations, but that strategic bidding by producer agents, which can be quite beneficial to them, can cause the auction-based approach to fail to find solutions in some cases. Their supply-chain problem did not consider temporal constraints.

Kutanoglu and Wu (1997) use an iterative-auction approach (but with only a single auction) to solve a distributed resource-scheduling problem in which a set of jobs must be performed, each job consisting of a set of operations, each operation requiring a particular machine for some duration. They associate an agent with each job and the set of biddable items is the set of discrete machine/time-slot pairs, each pair having an associated price. For each auction iteration, each agent generates a single bid; the auctioneer examines the bids and then updates the prices in an attempt to reduce resource conflicts. The procedure stops when the auctioneer finds that all of the bids are compatible. The primary differences between the approach taken in this chapter and that of Kutanoglu and Wu are: (1) they map agents to jobs in a one-to-one fashion and use the auction to find sets of machine/time-slot pairs to satisfy the needs of each job, whereas the ICDP mechanism uses an auction to find a mapping from agents to jobs; and (2) their biddable items are machine/time-slot pairs, of which there are very many, whereas the biddable items in an auction in the ICDP mechanism are roles, of which there are comparatively few.

**Auctions with Temporal Constraints**

Parkes and Ungar (2001) investigate the use of multiple, iterative, combinatorial auctions in a train-scheduling domain. In their work, the train-scheduling domain comprises (1) separate networks of train tracks (called *territories*), each controlled by "an autonomous dispatch agent responsible for the flow of trains [through that] territory"; and (2) a set of independent trains, each "represented by a self-interested agent that bids for the right to travel across the [global] network from its source to destination, submitting bids to multiple dispatch agents along its route as necessary."

In their decentralized approach to the scheduling problem, each dispatch agent runs an iterative combinatorial auction in which train agents bid for the right to travel across the territory controlled by that dispatch agent. Each train agent has pre-assigned entry and exit points for each territory. The global scheduling problem is to determine entry and exit *times* that will satisfy safety requirements (e.g., that trains will not collide) while maximizing "the total cost-adjusted value over all trains."

Each bid submitted by a train agent is allowed to specify unary temporal constraints of the form, 'enter before time T' or 'exit before time T'. Train agents may submit multiple bids for a given auction, but will be awarded at most one bid per auction. The auction for each territory is iterative to enable agents to coordinate their bids across several auctions.

In a suite of experiments on scenarios involving "linear chains of dispatcher territories", Parkes and Ungar found that their decentralized, auction-based approach generated better schedules than a comparable centralized approach. In addition, the winner-determination algorithm for the auction-based approach appeared to scale well "with the number of train agents and in particular with the number of dispatchers."

The approach taken by Parkes and Ungar is similar to the approach taken in this chapter in that it allows bids in a combinatorial auction to include (unary) temporal constraints. Their formulation of the winner-determination problem as a mixed-integer programming problem (using "the big M technique") could provide an alternative means of formulating the winner-determination problem for the ICDP mechanism described in this chapter. The bids submitted by train agents may be viewed as covering a *single, fixed* item (rather than an arbitrary set of items) subject to at most two unary temporal constraints, whereas in the ICDP mechanism, agents bid on arbitrary *sets* of roles, and may include numerous binary temporal constraints among time-points associated with the proposed group activity.

# Part II

# Temporal Reasoning

# Chapter 4

# Simple Temporal Networks

The temporal reasoning problems addressed in this thesis are stated and analyzed in terms of Simple Temporal Networks (Dechter, Meiri, and Pearl, 1991). A Simple Temporal Network (STN) comprises a set of time-point variables and binary constraints among those variables. In this thesis, STNs are used to represent (1) temporal constraints among the tasks to which an agent has committed itself; (2) constraints on the durations of actions in act-type and recipe definitions; (3) precedence relations (with temporal offsets) among subacts in recipes; (4) additional temporal constraints associated with particular action instances; and (5) temporal constraints in bids. The standard operations on STNs (e.g., determining whether a solution exists, propagating new constraints, and determining which constraints may be safely added) require only polynomial time.

This chapter summarizes the theory of Simple Temporal Networks and provides some extensions to that theory to facilitate the analysis in Chapter 5. The concepts in the first section derive, directly or indirectly, from Dechter et al. (1991). However, in some cases, the definitions and theorems have been recast in equivalent terms to facilitate their later use.

## 4.1 Existing Theory of Simple Temporal Networks

**Definition 4.1 (Simple Temporal Network)** *(Dechter, Meiri, and Pearl, 1991) A* Simple Temporal Network *$\mathcal{S}$ is a pair $(\mathcal{T}, \mathcal{C})$, where $\mathcal{T}$ is a set $\{t_0, t_1, \ldots, t_N\}$ of time-point variables and $\mathcal{C}$ is a finite set of binary constraints on those variables, each constraint having the form $t_j - t_i \leq \delta$, for some real number $\delta$. The "variable" $t_0$ represents an arbitrary, fixed reference point on the time-line. (In this thesis, we fix $t_0$ to the value $0$ and frequently refer to it as the* zero time-point variable*, or $z$.) The constraints in $\mathcal{C}$ are called the* explicit constraints *in $\mathcal{S}$.*

**Definition 4.2 (STN Solution)** *A* solution *to an STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ is a complete set of variable assignments*

$$\{z = 0,\ t_1 = w_1,\ \ldots,\ t_N = w_N\},\ \text{each } w_i \in \mathbb{R}$$

*that satisfies all the constraints in $\mathcal{C}$.*

**Definition 4.3 (Consistent STN)** *An STN that has at least one solution is called* consistent.

**Definition 4.4 (Equivalent STNs)** *Two STNs $\mathcal{S}_1$ and $\mathcal{S}_2$ (over the same set of time point variables) are said to be* equivalent*, written $\mathcal{S}_1 \equiv \mathcal{S}_2$, if they admit identical solution sets.*

Unary constraints on time-point variables in an STN are represented as binary constraints involving $z$, as follows (where $L$ and $U$ are constants):

$$\begin{array}{rcl c rcl c rcl} L & \leq & t_i & \Leftrightarrow & 0 - t_i & \leq & -L & \Leftrightarrow & z - t_i & \leq & -L \\ t_i & \leq & U & \Leftrightarrow & t_i - 0 & \leq & U & \Leftrightarrow & t_i - z & \leq & U. \end{array}$$

**Definition 4.5 (Trivial Constraints, Lean Constraint Set)** *A constraint $t_j - t_i \leq \delta$ in $\mathcal{C}$ is called* trivial in $\mathcal{C}$ *if there exists a constraint $t_j - t_i \leq \delta'$ in $\mathcal{C}$ such that $\delta' < \delta$; otherwise, $t_j - t_i \leq \delta$ is called* non-trivial in $\mathcal{C}$. *A set $\mathcal{C}$ of constraints is called* lean *if it contains only non-trivial constraints.*

**Proposition 4.6** *Each STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ is equivalent to a unique lean STN $\mathcal{S}_\ell = (\mathcal{T}, \mathcal{C}_\ell)$, where $\mathcal{C}_\ell \subseteq \mathcal{C}$.*

**Proof** The set $\mathcal{C}_\ell$ may be obtained by removing all trivial constraints from $\mathcal{C}$. Since removing a trivial constraint does not affect the set of solutions, $\mathcal{S}$ and $\mathcal{S}_\ell$ are equivalent. Uniqueness follows from the observation that a set contains only one copy of each element and thus there can exist at most one non-trivial constraint for each ordered pair $(t_i, t_j)$ of time-points. ■

**Definition 4.7 (Entailment)** *Given sets $\mathcal{C}_1$ and $\mathcal{C}_2$ of constraints over a set $\mathcal{T}$ of time-points, the constraints in $\mathcal{C}_1$ are said to* entail *the constraints in $\mathcal{C}_2$ if any solution to the STN $(\mathcal{T}, \mathcal{C}_1)$ is necessarily a solution to the STN $(\mathcal{T}, \mathcal{C}_2)$.*

**Definition 4.8 (Equivalent Constraint Sets)** *Given sets $\mathcal{C}_1$ and $\mathcal{C}_2$ of constraints over a set $\mathcal{T}$ of time-points, if the constraints in $\mathcal{C}_1$ entail those in $\mathcal{C}_2$, and the constraints in $\mathcal{C}_2$ entail those in $\mathcal{C}_1$, then $\mathcal{C}_1$ and $\mathcal{C}_2$ are called* equivalent *constraint sets, in which case we write: $\mathcal{C}_1 \equiv \mathcal{C}_2$.*

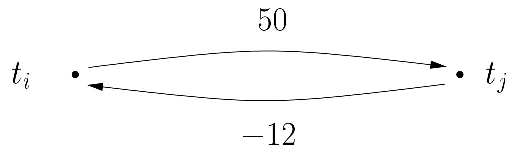
**Definition 4.9 (Constraints Consistent with STN)** *A set $\mathcal{C}'$ of constraints over the time-points in $\mathcal{T}$ is said to be* consistent *with the STN $(\mathcal{T}, \mathcal{C})$ if the STN $(\mathcal{T}, \mathcal{C} \cup \mathcal{C}')$ is consistent.*

**Definition 4.10 (Distance Graph)** *(Dechter, Meiri, and Pearl, 1991) The* distance graph *for an STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ is a weighted, directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, whose vertices correspond to the time-points of $\mathcal{S}$ and whose directed edges correspond to the* lean *temporal constraints of $\mathcal{S}$, as follows:*

$$\mathcal{V} = \mathcal{T} \qquad \text{and} \qquad \mathcal{E} = \{(t_i, \delta, t_j) : (t_j - t_i \leq \delta) \in \mathcal{C}_\ell\}.$$

*The value $\delta$ in an edge $(t_i, \delta, t_j)$ is called the* length *of the edge.*

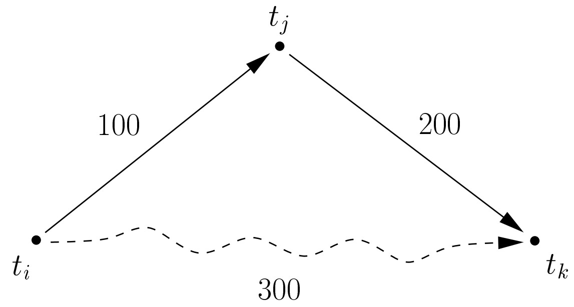
Thus, each *non-trivial* temporal constraint $t_j - t_i \leq \delta$ in an STN is represented in the corresponding distance graph by a directed edge from $t_i$ to $t_j$ with weight (or length) $\delta$. For example, the constraints $t_j - t_i \leq 50$ and $t_i - t_j \leq -12$ would be represented by the edges shown below:




In an STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$, the *explicit* constraints in $\mathcal{C}$ may give rise to additional *implicit* constraints.[1] For example, the explicit constraints $t_j - t_i \leq 100$ and $t_k - t_j \leq 200$ combine (using simple arithmetic) to entail the implicit constraint $t_k - t_i \leq 300$, as follows:

$$t_k - t_i \;=\; (t_k - t_j) + (t_j - t_i) \;\leq\; 200 + 100 \;=\; 300.$$

In graphical terms, the edges from $t_i$ to $t_j$ and $t_j$ to $t_k$ form a *path* of length 300 from $t_i$ to $t_k$, shown as a squiggly arc below.




In general, each path corresponds to an implicit constraint on its endpoints.

**Definition 4.11 (Path)** *A* path *from $t_i$ to $t_j$ in the distance graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a sequence $P = (\tau_1, \ldots, \tau_m)$ such that:*

- $m \geq 1$*;*
- $\tau_1 = t_i$ *and* $\tau_m = t_j$*;*
- $\tau_1, \ldots, \tau_m \in \mathcal{V}$*; and*
- *for each $i \in \{1, \ldots, m-1\}$, there is an edge of the form $(\tau_i, \delta_i, \tau_{i+1})$ in $\mathcal{E}$.*

*(If $P$ is a path in the distance graph of an STN $\mathcal{S}$, then we may also say that $P$ is a path in $\mathcal{S}$.) The length of $P$, denoted $|P|$, is the sum of the lengths of the edges constituting $P$:*

$$|P| \;=\; \sum_{i=1}^{m-1} \delta_i$$

*If $\tau_1 = \tau_m$, then $P$ is called a* loop.

[1]For convenience, an explicit constraint is taken to be a special case of an implicit constraint. Thus, the set of implicit constraints contains the set of explicit constraints.

Notice that the length of a path may be negative.

**Definition 4.12 (Shortest Path)** *A path $P$ from $t_i$ to $t_j$ in a distance graph $\mathcal{G}$ is called a* shortest path *from $t_i$ to $t_j$ in $\mathcal{G}$ if:*

$$|P| \;=\; \min\,\{\,|P'| : P' \text{ is a path from } t_i \text{ to } t_j \text{ in } \mathcal{G}\,\}.$$

**Proposition 4.13** *(Dechter, Meiri, and Pearl, 1991) If there exists a path $P$ from $t_i$ to $t_j$ in the distance graph of an STN $\mathcal{S}$, then the implicit constraint, $t_j - t_i \le |P|$, is entailed by the explicit constraints in $\mathcal{S}$. In other words, the length of $P$ is an upper bound on the temporal difference, $t_j - t_i$, in any solution for $\mathcal{S}$.*

**Proof** For any path $P = (\tau_1, \ldots, \tau_m)$:

$$\begin{array}{rcccccccc} \tau_m - \tau_1 & = & (\tau_m - \tau_{m-1}) & + & (\tau_{m-1} - \tau_{m-2}) & + & \ldots & + & (\tau_2 - \tau_1) \\ & \le & \delta_{m-1} & + & \delta_{m-2} & + & \ldots & + & \delta_1 \\ & = & |P|. \ \blacksquare & & & & & & \end{array}$$

**Definition 4.14 (Temporal Distance)** *(Dechter, Meiri, and Pearl, 1991)*[2] *The* temporal distance *from $t_i$ to $t_j$ in an STN $\mathcal{S}$ is the length of the* shortest path *from $t_i$ to $t_j$ in the distance graph $\mathcal{G}$ (or negative infinity if no such path exists):*

$$dist(t_i, t_j) \;=\; \mathrm{glb}\,\{\,|P| : P \text{ is a path from } t_i \text{ to } t_j \text{ in } \mathcal{G}\,\}.$$

Since path lengths may be negative, temporal distances also may be negative. In fact, if there is a *loop* with negative path length, then some temporal distances will necessarily be $-\infty$. (However, as will be seen, a consistent STN cannot have any loops with negative path length.)

**Definition 4.15 (Distance Matrix)** *(Dechter, Meiri, and Pearl, 1991)*[3] *The* distance matrix *for an STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ is a matrix $\mathcal{D}$ each entry of which equals the temporal distance between the corresponding pair of time-points in $\mathcal{T}$:*

$$\mathcal{D}(i,j) \;=\; dist(t_i, t_j).$$

Thus, $\mathcal{D}$ is a $|\mathcal{T}|$-by-$|\mathcal{T}|$ matrix. Abusing notation slightly, we may write $\mathcal{D}(t_i, t_j)$ where $t_i$ and $t_j$ are time-point variables rather than indices. Notice that the fixed, zero-time-point variable $z$ is associated with the zeroeth row and column of the distance matrix.

**Proposition 4.16** *For any time-points $t_i$ and $t_j$ in an STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ with distance matrix $\mathcal{D}$, the following implicit constraints are entailed by the constraints in $\mathcal{C}$:*

$$-\mathcal{D}(t_j, t_i) \;\le\; t_j - t_i \;\le\; \mathcal{D}(t_i, t_j).$$

[2] The concept of temporal distance, implicit in Dechter et al., is made explicit in Tsamardinos (2000).

[3] The concept of the distance matrix is implicit in Dechter et al. Tsamardinos (2001) uses the term *distance array*.

If the temporal *distance* $\mathcal{D}(t_i, t_j)$ is positive infinity, which happens if and only if there are no paths from $t_i$ to $t_j$ in the distance graph, then the temporal *difference* $t_j - t_i$ is unconstrained. If the temporal *distance* $\mathcal{D}(t_i, t_j)$ is negative infinity, which happens if and only if there is a loop in the distance graph having negative path length, then the temporal *difference* $t_j - t_i$ is impossibly constrained. Similar remarks apply to $\mathcal{D}(t_j, t_i)$.

**Definition 4.17 (Strongest Implicit Constraint)** *For any time-points $t_i$ and $t_j$ in an STN $\mathcal{S}$ with distance matrix $\mathcal{D}$, the implicit constraint, $t_j - t_i \leq \mathcal{D}(t_i, t_j)$, is called the* strongest implicit constraint *from $t_i$ to $t_j$ in $\mathcal{S}$.*

The definition of temporal distance in terms of shortest paths leads to the following property.

**Proposition 4.18 (The Triangle Inequality)** *(Tsamardinos, 2000) For any STN $\mathcal{S}$ with distance matrix $\mathcal{D}$, the following inequality holds among each triple of time-points, $t_i, t_j$ and $t_k$:*

$$\mathcal{D}(t_i, t_k) \;\leq\; \mathcal{D}(t_i, t_j) + \mathcal{D}(t_j, t_k).$$

**Definition 4.19 (Tight Edge/Constraint)** *(Morris and Muscettola, 2000) If an explicit constraint (or edge), $(t_j - t_i \leq \delta)$, in an STN $\mathcal{S}$ is such that $\delta = \mathcal{D}(t_i, t_j)$, then that constraint is said to be* tight *in $\mathcal{S}$.*[4]

**Proposition 4.20** *Any subpath $P'$ of a shortest path $P$ is itself a shortest path. Hence, any edge along a shortest path must be tight.*

**Lemma 4.21** *For an edge $E$ of the form $t_j - t_i \leq \delta$, if $E$ is* tight*, then the following inequalities necessarily hold:*

$$\mathcal{D}(t_i, z) - \mathcal{D}(t_j, z) \;\leq\; \delta \quad \textit{and} \quad \mathcal{D}(z, t_j) - \mathcal{D}(z, t_i) \;\leq\; \delta.$$

**Proof** The first inequality may be proven as follows:

$$\begin{array}{ll}
\mathcal{D}(t_i, z) \;\leq\; \mathcal{D}(t_i, t_j) + \mathcal{D}(t_j, z) & \text{(Triangle Inequality)} \\
\mathcal{D}(t_i, z) \;\leq\; \delta + \mathcal{D}(t_j, z) & \text{(Since } E \text{ is a tight edge)} \\
\mathcal{D}(t_i, z) - \mathcal{D}(t_j, z) \;\leq\; \delta & \text{(Rearrange terms)}
\end{array}$$

The second inequality follows from applying the Triangle Inequality to the edge from $z$ to $t_j$. ■

**Proposition 4.22 (Computing the Distance Matrix)** *As observed by Dechter et al. (1991), since the entry $\mathcal{D}(t_i, t_j)$ is equal to the length of the shortest path from $t_i$ to $t_j$ in the graph $\mathcal{G}$, the distance matrix for a given STN may be computed from scratch in polynomial time, for example, using Floyd-Warshall's $O(|\mathcal{T}|^3)$ "all-pairs shortest-path" algorithm (Cormen, Leiserson, and Rivest, 1990).*

[4] Dechter et al. (1991) define a tightness relation among STNs that is altogether different from an edge being tight in an STN.

### 4.1.1 The d-graph and the d-STN

Although the distance graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ for an STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ is an important analytical tool, another type of graph, called the *d-graph* of $\mathcal{S}$, is also very important. For example, Theorem 4.27 (below) shows that any set of variable assignments satisfying the constraints in the d-graph of $\mathcal{S}$ can always be extended to a solution for $\mathcal{S}$.

Whereas the edges in the distance graph correspond one-to-one to the (non-trivial) constraints in $\mathcal{C}$, the edges in the d-graph correspond one-to-one to the *shortest paths* in $\mathcal{S}$.[5] Thus, whereas the distance graph contains only $|\mathcal{C}_\ell|$ edges and only implicitly specifies temporal distances in $\mathcal{S}$ in terms of shortest paths, the d-graph contains $|\mathcal{T}|^2$ edges, each of which explicitly specifies a temporal distance in $\mathcal{S}$.

**Definition 4.23 (d-Graph)** *(Dechter, Meiri, and Pearl, 1991) The* d-graph *for an STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ is a* complete, *directed graph $\mathcal{G}_d = (\mathcal{V}_d, \mathcal{E}_d)$ whose vertices correspond to the time-points of $\mathcal{S}$ and whose edges correspond to the temporal* distances *among those time-points:*

$$\mathcal{V}_d = \mathcal{T} \quad \text{and} \quad \mathcal{E}_d = \{(t_i, \mathcal{D}(i,j), t_j) : t_i, t_j \in \mathcal{T}\}.$$

Corresponding to the d-graph $\mathcal{G}_d$ (of $\mathcal{S}$) is an STN $\mathcal{S}_d$ called the d-STN (of $\mathcal{S}$).

**Definition 4.24 (d-STN)** *(Dechter, Meiri, and Pearl, 1991)*[6] *The* d-STN *of an STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ is an STN $\mathcal{S}_d = (\mathcal{T}, \mathcal{C}_d)$ whose time-points are the same as those of $\mathcal{S}$, but whose constraints correspond directly to the edges in the d-graph of $\mathcal{S}$ (and hence to the entries in the distance matrix of $\mathcal{S}$), as follows:*

$$\mathcal{C}_d \;=\; \{t_j - t_i \leq \mathcal{D}(i,j) : t_i, t_j \in \mathcal{T}\}.$$

Figure 4.1 illustrates the relationships among $\mathcal{S}$, its distance graph $\mathcal{G}$, its d-graph $\mathcal{G}_d$, and its d-STN $\mathcal{S}_d$. Notice that the d-graph of $\mathcal{S}$ is the distance graph of the d-STN of $\mathcal{S}$.

### 4.1.2 The Decomposability of a Consistent STN

**Definition 4.25 (Relevant Constraints)** *Given an STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ and a non-empty set of time-points $\mathcal{T}' \subseteq \mathcal{T}$, the set of constraints in $\mathcal{C}$* relevant *to $\mathcal{T}'$ is denoted by $\mathcal{C}|_{\mathcal{T}'}$, where:*

$$\mathcal{C}|_{\mathcal{T}'} = \{(t_j - t_i \leq \delta) \in \mathcal{C} : t_i, t_j \in \mathcal{T}'\}.$$

**Definition 4.26 (Locally Consistent Assignment)** *(Dechter, Meiri, and Pearl, 1991) Given sets of time-points $\mathcal{T}' \subseteq \mathcal{T}$ and a set of constraints $\mathcal{C}$ over the time-points in $\mathcal{T}$, a set of variable assignments for the time-points in $\mathcal{T}'$ is called* locally consistent *with respect to $\mathcal{C}$ if those variable assignments satisfy the constraints in $\mathcal{C}|_{\mathcal{T}'}$.*

[5] Morris and Muscettola (2000) refer to the d-graph as the *all-pairs graph*.

[6] We use the term d-STN to highlight the connection between d-STNs and d-graphs. Dechter et al. do not use this term. Instead, they refer to what we are calling the d-STN as the *minimal network representation* for $\mathcal{S}$—minimal in the sense that it explicitly specifies the minimal *domain* for each temporal difference.

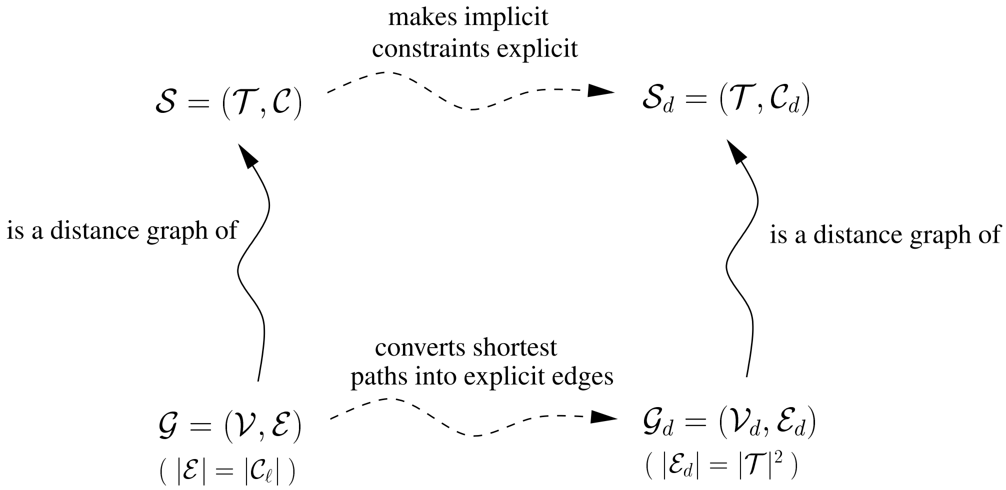



Figure 4.1: The relationships among $\mathcal{S}$, $\mathcal{G}$, $\mathcal{S}_d$ and $\mathcal{G}_d$

In general, a locally consistent assignment is not guaranteed to be extendible to a solution for an entire STN. For example, if $\mathcal{T}' = \{z, t_1\}$, then a locally consistent assignment for the variables in $\mathcal{T}'$ consists solely of an assignment of $t_1$ to a value that satisfies all of the constraints in $\mathcal{C}$ involving $z$ and $t_1$ (i.e., all unary constraints on $t_1$). In general, such an assignment is not guaranteed to satisfy other constraints in $\mathcal{C}$, say, among $t_1$ and some other time-point $t_2$. However, the following theorem says that a locally consistent assignment with respect to $\mathcal{C}_d$ (i.e., the constraints in the d-STN of $\mathcal{S}$) can always be extended to a solution for $\mathcal{S}$.

**Theorem 4.27 (Decomposability Relative to d-STN)** *(Dechter, Meiri, and Pearl, 1991) Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be a consistent STN. Let $\mathcal{C}_d$ be the constraints in the d-STN associated with $\mathcal{S}$. Then any set of variable assignments to time-points in any set $\mathcal{T}' \subseteq \mathcal{T}$ that is locally consistent with respect to $\mathcal{C}_d$ can be extended to a solution for $\mathcal{S}$.*

**Corollary 4.28** *(Dechter, Meiri, and Pearl, 1991) Two STNs are equivalent if and only if their distance matrices are identical.*

**Proposition 4.29** *(Dechter, Meiri, and Pearl, 1991) The distance matrices for $\mathcal{S}$ and $\mathcal{S}_d$ are identical (i.e., $\mathcal{D} = \mathcal{D}_d$) and, thus, $\mathcal{S} \equiv \mathcal{S}_d$.*

**Theorem 4.30** *(Dechter, Meiri, and Pearl, 1991)*[7] *An STN $\mathcal{S}$ is consistent if and only if its corresponding distance graph $\mathcal{G}$ has no negative cycles (i.e., if and only if the path length around any loop is non-negative).*

[7]Dechter et al. cite others regarding this theorem (Shostak, 1981; Liao and Wong, 1983; Leiserson and Saxe, 1983).

**Corollary 4.31** *An STN $\mathcal{S}$ is consistent if and only if all of the diagonal entries in its distance matrix are zero:*

$$\mathcal{D}(t_i, t_i) \;=\; 0, \text{ for all } t_i \in \mathcal{T}.$$

**Proposition 4.32** *(Tsamardinos, 2000) Given Theorem 4.30, the following inequality holds among each pair of time-points $t_i$ and $t_j$ in a consistent STN:*

$$\mathcal{D}(t_i, t_j) + \mathcal{D}(t_j, t_i) \geq 0.$$

Proposition 4.32 says that the length of any loop from $t_i$ to $t_j$ and back to $t_i$ is non-negative.

### 4.1.3 Adding Constraints to an STN

Typically, adding a constraint to an STN causes some entries in the distance matrix to change.[8] The following theorem specifies which constraints can be added to an STN without threatening its consistency.

**Theorem 4.33** *(Dechter, Meiri, and Pearl, 1991) For any time-points $t_i$ and $t_j$ in a consistent STN $\mathcal{S}$, the new constraint $t_j - t_i \leq \delta$ will not threaten the consistency of $\mathcal{S}$ if and only if $\delta$ satisfies $-\mathcal{D}(t_j, t_i) \leq \delta$. Furthermore, $\mathcal{S}$ has a solution in which $t_j - t_i = \sigma$ if and only if $\sigma \in [-\mathcal{D}(t_j, t_i), \mathcal{D}(t_i, t_j)]$.*

**Corollary 4.34** *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be the distance matrices for the consistent STNs $\mathcal{S}_1 = (\mathcal{T}, \mathcal{C}_1)$ and $\mathcal{S}_2 = (\mathcal{T}, \mathcal{C}_2)$. Then the following are equivalent:*

- *The constraints in $\mathcal{C}_1$ entail the constraints in $\mathcal{C}_2$.*
- *$\mathcal{D}_1(t_i, t_j) \leq \mathcal{D}_2(t_i, t_j)$, for all $t_i, t_j \in \mathcal{T}$.*

**Proof** ($\Rightarrow$) Suppose the constraints in $\mathcal{C}_1$ entail those in $\mathcal{C}_2$. Let $t_i, t_j \in \mathcal{T}$ be arbitrary. By Theorem 4.33, there exists a solution $\mathcal{X}_1$ to $\mathcal{S}_1$ for which $t_j - t_i = \mathcal{D}_1(t_i, t_j)$. Since the constraints in $\mathcal{C}_1$ entail the constraints in $\mathcal{C}_2$ (cf. Definition 4.7), $\mathcal{X}_1$ must also be a solution for $\mathcal{S}_2$, which, by Proposition 4.16, implies that $t_j - t_i \leq \mathcal{D}_2(t_i, t_j)$. Thus,

$$\mathcal{D}_1(t_i, t_j) \;=\; t_j - t_i \;\leq\; \mathcal{D}_2(t_i, t_j).$$

($\Leftarrow$) Suppose $\mathcal{D}_1(t_i, t_j) \leq \mathcal{D}_2(t_i, t_j)$, for all $t_i, t_j \in \mathcal{T}$. Let $\mathcal{X}_1$ be any solution for $(\mathcal{T}, \mathcal{C}_1)$. Let $E$: $t_j - t_i \leq \delta$ be an arbitrary constraint in $\mathcal{C}_2$. Since $\mathcal{X}_1$ is a solution for $(\mathcal{T}, \mathcal{C}_1)$, we have that $t_j - t_i \leq \mathcal{D}_1(t_i, t_j)$. In addition, since $E$ is a constraint in $\mathcal{C}_2$, we have that $\mathcal{D}_2(t_i, t_j) \leq \delta$. Putting these inequalities together with the assumption that $\mathcal{D}_1(t_i, t_j) \leq \mathcal{D}_2(t_i, t_j)$ yields the following:

$$t_j - t_i \;\leq\; \mathcal{D}_1(t_i, t_j) \;\leq \mathcal{D}_2(t_i, t_j) \;\leq \delta,$$

which implies that $E$ is satisfied in $\mathcal{X}_1$. Since $E$ was arbitrary in $\mathcal{C}_2$, $\mathcal{X}_1$ is a solution for $(\mathcal{T}, \mathcal{C}_2)$. Since $\mathcal{X}_1$ was arbitrary, the constraints in $\mathcal{C}_1$ entail those in $\mathcal{C}_2$. ■

[8] When a new constraint is added to the network, rather than recomputing the distance matrix from scratch, it is more efficient to incrementally update the distance matrix using constraint propagation techniques (Chleq, 1995).

**Corollary 4.35** *The quantity* $\mathcal{D}(t_i,t_j)+\mathcal{D}(t_j,t_i)$*, which specifies the length of the interval* $[-\mathcal{D}(t_j,t_i),\mathcal{D}(t_i,t_j)]$*, also specifies the maximum amount by which the strongest implicit constraint from* $t_i$ *to* $t_j$ *may be* tightened.

Corollary 4.35 says that the quantity $\mathcal{D}(t_i,t_j)+\mathcal{D}(t_j,t_i)$ specifies the maximum amount by which the constraints joining the time-points $t_i$ and $t_j$ (in either direction) may be tightened. Adding a constraint $t_j - t_i \leq \delta$ in the extreme case where $\delta = -\mathcal{D}(t_j,t_i)$ (recall Theorem 4.33), causes the *updated* distance matrix entries to satisfy:

$$-\mathcal{D}(t_j,t_i) \;=\; t_j - t_i \;=\; \mathcal{D}(t_i,t_j).$$

In such a case, the temporal difference $t_j - t_i$ is constrained to be fixed; equivalently, $\mathcal{D}(t_i,t_j) + \mathcal{D}(t_j,t_i) = 0$.

**Definition 4.36 (Rigidly Connected Time-Points)** *(Tsamardinos, Muscettola, and Morris, 1998; Gerevini, Perini, and Ricci, 1996; Wetprasit and Sattar, 1998) The time-points* $t_i$ *and* $t_j$ *are said to be* rigidly connected *in an STN* $\mathcal{S} = (\mathcal{T},\mathcal{C})$ *if:*

$$\mathcal{D}(t_i,t_j) + \mathcal{D}(t_j,t_i) = 0.$$

*A set of time-points* $\mathcal{T}_R \subseteq \mathcal{T}$ *forms a* rigidly-connected component *(or a* rigid component*) in* $\mathcal{S}$ *if each pair of time-points in* $\mathcal{T}_R$ *is rigidly connected in* $\mathcal{S}$*.*

### 4.1.4 Measures of Flexibility and Rigidity for an STN

If the time-points of an STN represent the execution times of tasks to which an agent is committed, then the more flexible the constraints in that STN, the greater the flexibility of that agent in, for example, integrating new tasks into its schedule. (Tasks subject to flexible constraints can be moved around to accommodate new tasks.) Thus, the flexibility of an STN measures one aspect of the *goodness* of an STN from the viewpoint of the agent. Similarly, the rigidity of an STN is a measure of the *badness* of an STN from the agent's viewpoint.[9]

The *relative flexibility* of the time-points $t_i$ and $t_j$ (Definition 4.37, below) specifies the maximum amount by which the current strongest implicit constraint on $t_i$ and $t_j$ may be strengthened (in either direction). The relative flexibility of the pair, $t_i$ and $t_j$—which is the same as the relative flexibility of the pair $t_j$ and $t_i$—is a number in the interval $[0,\infty)$. The relative flexibility is $0$ if the time-points are rigidly connected, $\infty$ if their temporal difference is unconstrained. The *relative rigidity* of a pair of time-points (Definition 4.38, below) is (almost) the reciprocal of their relative flexibility. The *root-mean-square (RMS) rigidity* of an STN (Definition 4.38) is defined in terms of the relative rigidity among each pair of time-points in the STN.

[9]The measure of rigidity defined here is similar to the measure of rigidity defined in an earlier paper (Hunsberger, 2002b).

**Definition 4.37 (Relative Flexibility)** *Given time-points $t_i$ and $t_j$ in a consistent STN, the relative* flexibility *of $t_i$ and $t_j$ is the (non-negative) quantity:*

$$Flex(t_i, t_j) \;=\; \mathcal{D}(t_i, t_j) + \mathcal{D}(t_j, t_i).$$

**Definition 4.38 (Relative Rigidity, RMS Rigidity)** *The* relative rigidity *of the pair of time-points $t_i$ and $t_j$ in a consistent STN is the quantity:*

$$Rig(t_i, t_j) \;=\; \frac{1}{1 + Flex(t_i, t_j)} \;=\; \frac{1}{1 + \mathcal{D}(t_i, t_j) + \mathcal{D}(t_j, t_i)}.$$

*The* RMS rigidity *of a consistent STN $\mathcal{S}$ is the quantity:*

$$Rig(\mathcal{S}) \;=\; \sqrt{\frac{2}{|\mathcal{T}|(|\mathcal{T}|-1)} \sum_{i<j} [Rig(t_i, t_j)]^2}\;.$$

From $Flex(t_i, t_j) \in [0, \infty)$, it follows that $Rig(t_i, t_j) \in [0, 1]$ and $Rig(\mathcal{S}) \in [0, 1]$. If $t_i$ and $t_j$ are part of a rigid component, then $Flex(t_i, t_j) = 0$ and $Rig(t_i, t_j) = 1$. Similarly, if $\mathcal{S}$ is completely rigid, then $Rig(\mathcal{S}) = 1$. At the opposite extreme, if $\mathcal{S}$ is completely unconstrained, then $Rig(\mathcal{S}) = 0$.

A pair of time-points may be rigidly connected to one another, yet not be fixed on the time-line. However, points that are rigidly connected to the zero time-point $z$, which is fixed on the time-line, are themselves necessarily fixed on the time-line.

**Proposition 4.39** *A time-point $t_i$ is rigidly connected to the zero time-point $z$ if and only if $t_i$ is constrained to be fixed. For example, $t_j - z = 10$ if and only if $t_j = 10$.*

### 4.1.5 Dealing with Rigid Components in an STN

Figure 4.2 shows an STN with two rigid components, $\{t_1, t_2, t_3\}$ and $\{t_4, t_5, t_6\}$. Several researchers (Tsamardinos, 2000; Gerevini, Perini, and Ricci, 1996; Wetprasit and Sattar, 1998) have presented algorithms for effectively *decoupling* rigid components from an STN.[10] For example, Tsamardinos (2000) presents an algorithm that:

(1) selects from each rigid component $RC$ a single time-point $t_{RC}$ that will represent the rigid component in the modified network;

(2) rearranges the edges in the network so that any edge in the modified network that interacts with a time-point in a rigid component $RC$ does so by interacting with the representative $t_{RC}$ of that component; and

[10]These researchers do not refer to this process as "decoupling" the rigid components from the network. Instead, they invariably refer to it as "collapsing" the rigid components down to single points. The word "decoupling" was used here to highlight the connection between these algorithms and the much more general temporal decoupling problems that are the subject of Chapter 5.

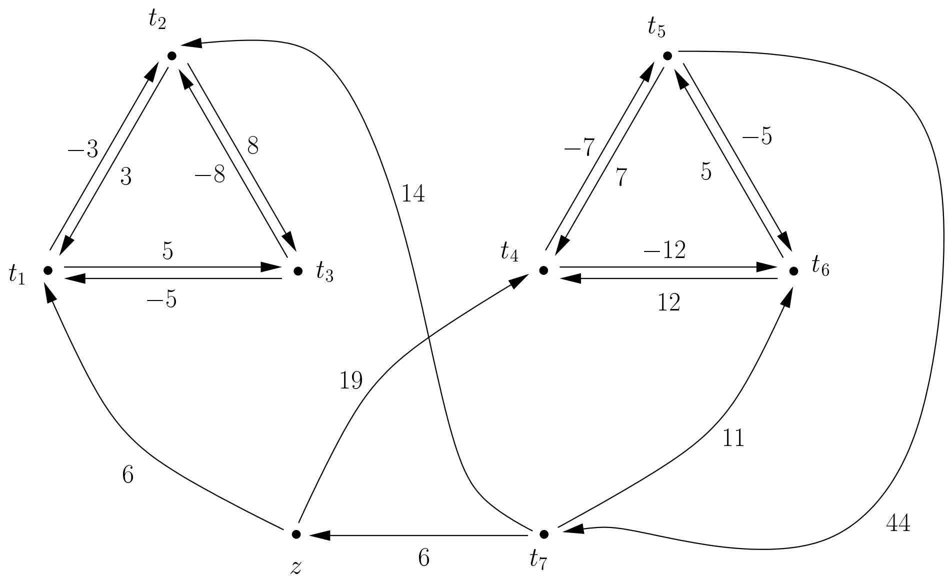



Figure 4.2: An STN with rigid components

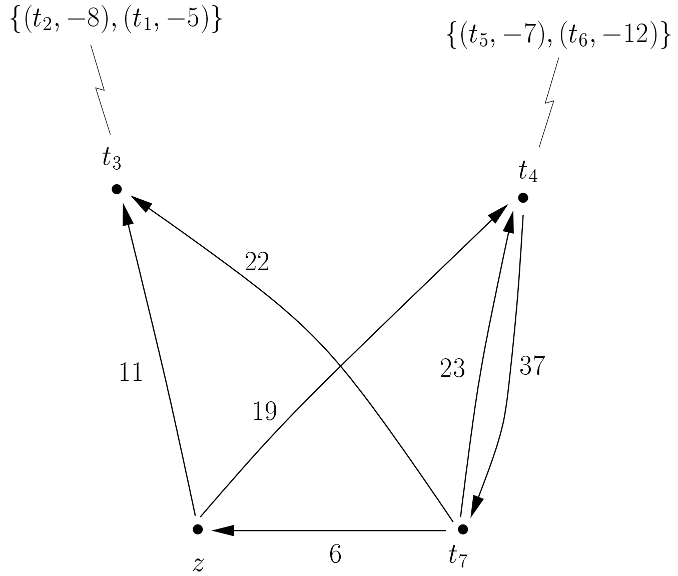



Figure 4.3: The STN from Figure 4.2 after collapsing its rigid components

(3) associates with each representative, additional information sufficient to reconstruct that representative's component.

When applied to the STN from Figure 4.2, the above algorithm generates the STN shown in Figure 4.3 (assuming that the time-points $t_3$ and $t_4$ are the chosen representatives). Notice that the algorithm has modified each edge involving a non-representative time-point in a rigid component to ensure that the edge's interface with the rigid component is through the component's representative. For example, the edge, $t_7 - t_5 \leq 44$, has been converted into the edge, $t_7 - t_4 \leq 37$, since $t_4$ is the representative for the rigid component containing $t_5$. Notice that the weight of the edge has been modified from $44$ to $37$, the difference of $-7$ being the distance from $t_4$ to $t_5$ in that component.

The additional information associated with each component's representative may take many forms. In Figure 4.3, each component's representative has a set of pairs associated with it, one pair for each non-representative time-point in the component, each offset specifying the distance from the representative to the non-representative time-point. For example, the pair $(t_2, -8)$ associated with the representative $t_3$ specifies that the distance from $t_3$ to $t_2$ is $-8$ in the rigid component represented by $t_3$.

## 4.2 Some Useful Extensions to the Theory of STNs

It is sometimes desirable (e.g., when transmitting a set of temporal constraints in a message) to represent an STN by an equivalent STN having the minimum number of explicit constraints. This section derives a canonical STN representation that is guaranteed to have the minimum number of explicit constraints. The canonical representation is constructed by first decoupling the rigid components as described in Section 4.1, and then removing all *dominated* constraints from the rest of the network. (A constraint is said to be dominated if removing it from the network does not change the set of solutions.)

This section also introduces the concept of a d-subnetwork, which is used in several key theorems in the next chapter.

**Definition 4.40 (Relevant Distance Submatrix)** *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be an STN with distance matrix $\mathcal{D}$. Let $\mathcal{T}'$ be a set of time-points such that $z \in \mathcal{T}' \subseteq \mathcal{T}$. The* distance submatrix *of $\mathcal{D}$* relevant to *$\mathcal{T}'$, denoted $\mathcal{D}|_{\mathcal{T}'}$, is the $|\mathcal{T}'|$-by-$|\mathcal{T}'|$ matrix given by:*

$$\mathcal{D}|_{\mathcal{T}'}(t_i, t_j) \;=\; \mathcal{D}(t_i, t_j), \textit{ for each } t_i, t_j \in \mathcal{T}'.$$

Notice that the indices of the two matrices typically do not correspond. Using variable names instead of integer indices to refer to the entries of the matrices sweeps this trivial problem under the rug.

**Definition 4.41 (d-Subnetwork)** *Given a consistent STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ and a set of time-points $\mathcal{T}'$ such that $z \in \mathcal{T}' \subseteq \mathcal{T}$, the* d-subnetwork *of $\mathcal{S}$ with respect to $\mathcal{T}'$ is the the STN $\mathcal{S}_d|_{\mathcal{T}'} = (\mathcal{T}', \mathcal{C}_d|_{\mathcal{T}'})$, where (combining Definitions 4.24 and 4.25):*

$$\mathcal{C}_d|_{\mathcal{T}'} = \{(t_j - t_i \leq \mathcal{D}(t_i, t_j)) : t_i, t_j \in \mathcal{T}'\}.$$

In other words, for each pair of time-points $t_i$ and $t_j$ in $\mathcal{T}'$, the d-subnetwork contains an explicit constraint of the form $t_j - t_i \leq \mathcal{D}(t_i, t_j)$. Notice that a shortest path from $t_i$ to $t_j$ in $\mathcal{S}$ may pass through points that are not in $\mathcal{T}'$. However, in the d-subnetwork, the *edge* from $t_i$ to $t_j$ has the same length as the shortest path from $t_i$ to $t_j$ in $\mathcal{S}$.

**Lemma 4.42** *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be an STN with distance matrix $\mathcal{D}$. Let $\mathcal{T}'$ be an arbitrary set of time-points such that $z \in \mathcal{T}' \subseteq \mathcal{T}$. Then the distance matrix for the d-subnetwork $\mathcal{S}_d|_{\mathcal{T}'}$ is identical to the distance submatrix of $\mathcal{D}_d$ relevant to $\mathcal{T}'$. Thus, the notation $\mathcal{D}_d|_{\mathcal{T}'}$ is unambiguous.*

**Proof** Let $\mathcal{D}_{[\mathcal{S}_d|_{\mathcal{T}'}]}$ be the distance matrix for $\mathcal{S}_d|_{\mathcal{T}'}$. Since any solution to $\mathcal{S}$ must satisfy the constraints in $\mathcal{C}_d|_{\mathcal{T}'}$, the constraints in $\mathcal{C}$ entail the constraints in $\mathcal{C}_d|_{\mathcal{T}'}$. Thus, by Corollary 4.34, $\mathcal{D}(t_i, t_j) \leq \mathcal{D}_{[\mathcal{S}_d|_{\mathcal{T}'}]}(t_i, t_j)$, for all $t_i$ and $t_j$ in $\mathcal{T}'$. However, for each $t_i, t_j \in \mathcal{T}'$, the constraint $t_j - t_i \leq \mathcal{D}(t_i, t_j)$ is in $\mathcal{C}_d|_{\mathcal{T}'}$, which implies that the opposite inequality, $\mathcal{D}_{[\mathcal{S}_d|_{\mathcal{T}'}]}(t_i, t_j) \leq \mathcal{D}(t_i, t_j)$, also holds. Thus, $\mathcal{D}_{[\mathcal{S}_d|_{\mathcal{T}'}]}(t_i, t_j) = \mathcal{D}(t_i, t_j)$ holds for all $t_i$ and $t_j$ in $\mathcal{T}'$. Finally, since $\mathcal{D} = \mathcal{D}_d$ (by Proposition 4.29), we have that $\mathcal{D}_{[\mathcal{S}_d|_{\mathcal{T}'}]} = (\mathcal{D}_d)|_{\mathcal{T}'}$. Thus, the notation $\mathcal{D}_d|_{\mathcal{T}'}$ may be used to refer to both the submatrix of $\mathcal{D}_d$ with respect to $\mathcal{T}'$ and the distance matrix for the d-subnetwork of $\mathcal{S}$ with respect to $\mathcal{T}'$ (i.e., what has been called $\mathcal{D}_{[\mathcal{S}_d|_{\mathcal{T}'}]}$ in this proof). ■

In terms of d-subnetworks, Theorem 4.27 may be restated as follows:

> Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be a consistent STN and $\mathcal{T}'$ a set of time-points such that $z \in \mathcal{T}' \subseteq \mathcal{T}$. Any solution to the d-subnetwork $\mathcal{S}_d|_{\mathcal{T}'}$ of $\mathcal{S}$ can be extended to a solution for $\mathcal{S}$.

We now provide a slight generalization of this theorem that will be quite useful in the next chapter.

**Theorem 4.43** *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be a consistent STN. Let $\mathcal{C}'$ be any set of constraints over the time-points in $\mathcal{T}'$, where $z \in \mathcal{T}' \subseteq \mathcal{T}$. If the constraints in $\mathcal{C}'$ are consistent with $\mathcal{S}_d|_{\mathcal{T}'}$, then they are also consistent with $\mathcal{S}$.*

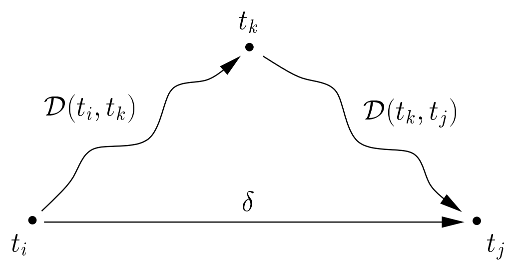
**Proof** Suppose that the constraints in $\mathcal{C}'$ are consistent with $\mathcal{S}_d|_{\mathcal{T}'} = (\mathcal{T}', \mathcal{C}_d|_{\mathcal{T}'})$. By Definition 4.9, this implies that the STN $\mathcal{S}' = (\mathcal{T}', \mathcal{C}_d|_{\mathcal{T}'} \cup \mathcal{C}')$ is consistent. Thus, there is a solution $\mathcal{X}'$ to $\mathcal{S}'$ (i.e., a set of assignments to the time-points in $\mathcal{T}'$ satisfying the constraints in $\mathcal{C}_d|_{\mathcal{T}'} \cup \mathcal{C}'$). In particular, the solution $\mathcal{X}'$ satisfies all the constraints in $\mathcal{C}_d|_{\mathcal{T}'}$. However, this implies that the solution $\mathcal{X}'$ also satisfies all of the constraints in $\mathcal{C}_d$ (since all of the constraints in $\mathcal{C}_d$ relevant to $\mathcal{T}'$ are contained in $\mathcal{C}_d|_{\mathcal{T}'}$). Thus, by Theorem 4.27, the set of variable assignments in $\mathcal{X}'$ must be extendible to a solution $\mathcal{X}$ for $\mathcal{S}$. Since the solution $\mathcal{X}'$ satisfies all the constraints in $\mathcal{C}'$, and since the constraints in $\mathcal{C}'$ only involve time-points in $\mathcal{T}'$, the solution $\mathcal{X}$ for $\mathcal{S}$ must also be a solution for $(\mathcal{T}, \mathcal{C} \cup \mathcal{C}')$. Thus, $(\mathcal{T}, \mathcal{C} \cup \mathcal{C}')$ is consistent (i.e., the constraints in $\mathcal{C}'$ are consistent with $\mathcal{S}$). ■

We now consider a notion of *dominance* among constraints in an STN. Our definition of dominance, our characterization of dominance in an STN without rigid components, and our algorithm for finding the *undominated constraints* in an STN are related to pre-existing work (Tsamardinos, 2000; Tsamardinos, Muscettola, and Morris, 1998). The key difference is that the definition of dominance in the pre-existing work is tied to the real-time execution of tasks in an STN, whereas ours is not.

**Definition 4.44 (Dominance)** *An explicit constraint (or edge)* $t_j - t_i \leq \delta$ *is* dominated *in a lean STN* $\mathcal{S}$ *if removing that edge would result in no change to the distance matrix for* $\mathcal{S}$.

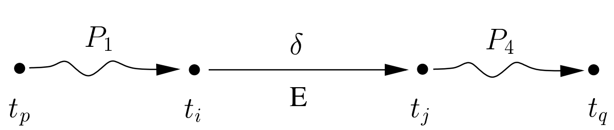
**Theorem 4.45 (Characterizing Dominance)** *Consider an edge* $E$ *of the form* $t_j - t_i \leq \delta$ *in a consistent STN* $\mathcal{S} = (\mathcal{T}, \mathcal{C})$.

- *If* $\mathcal{D}(t_i, t_j) < \delta$ *(i.e., if* $E$ *is not tight in* $\mathcal{S}$*), then* $E$ *is dominated in* $\mathcal{S}$.
- *If* $\delta = \mathcal{D}(t_i, t_j)$ *(i.e., if* $E$ *is tight in* $\mathcal{S}$*)* and there are no rigid components in $\mathcal{S}$*, then the edge* $t_j - t_i \leq \delta$ *is dominated in* $\mathcal{S}$ *if and only if there is some* other *point* $t_k \in \mathcal{T}$ *such that:*
$$\delta \;=\; \mathcal{D}(t_i, t_k) + \mathcal{D}(t_k, t_j),$$
*as illustrated below.*



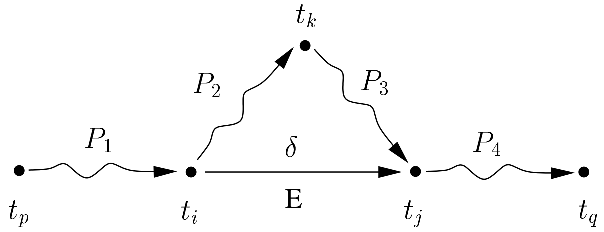

**Proof** For the first part, if $E$ is not tight, then (by Proposition 4.20) $E$ cannot lie along any shortest path. Hence, removing $E$ could not cause a change to the distance matrix.

For the second part, suppose that $\mathcal{S}$ has no rigid components and that the *tight* edge $E$, $t_j - t_i \leq \delta$, is *not* dominated in $\mathcal{S}$. Then $E$ must be part of some shortest path $P$ from some $t_p$ to some $t_q$ such that removing $E$ from $\mathcal{S}$ would cause $\mathcal{D}(t_p, t_q)$ to increase (i.e., the subsequent shortest path from $t_p$ to $t_q$ would be longer than $P$). $P$ has the form shown below:




where the squiggly arcs denote (possibly empty) *paths*. (Without loss of generality, $E$ may be assumed to occur only once on the path $P$.)

Suppose there exists some $t_k$ such that $\mathcal{D}(t_i, t_k) + \mathcal{D}(t_k, t_j) = \delta$. Then there exist shortest paths $P_2$ (from $t_i$ to $t_k$) and $P_3$ (from $t_k$ to $t_j$), as shown below




where $|P_2| + |P_3| = \delta$.

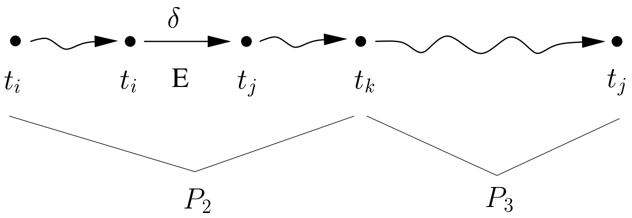
Let $P'$ be the concatenation of the paths $P_1, P_2, P_3$ and $P_4$. Then $|P'|$ is given by:

$$|P'| \; = \; |P_1| + |P_2| + |P_3| + |P_4| \; = \; |P_1| + \delta + |P_4| \; = \; |P| \; = \; \mathcal{D}(t_p, t_q).$$

Thus, $P'$ is also a shortest path from $t_p$ to $t_q$.

$E$ must lie along the path $P'$ since otherwise removing $E$ from $\mathcal{S}$ would not change the value of $\mathcal{D}(t_p, t_q)$. Since $E$ does not lie along $P_1$ or $P_4$, it must lie along $P_2$ or $P_3$.

Suppose $E$ lies along $P_2$. Then $P_2$ and $P_3$ look like this:




Thus,

$$\begin{array}{rrcl} & \delta & = & |P_2| + |P_3| \\ \Rightarrow & \delta & = & [\mathcal{D}(t_i, t_i) + \delta + \mathcal{D}(t_j, t_k)] + \mathcal{D}(t_k, t_j) \\ \Rightarrow & \delta & = & [0 + \delta + \mathcal{D}(t_j, t_k)] + \mathcal{D}(t_k, t_j) \\ \Rightarrow & 0 & = & \mathcal{D}(t_j, t_k) + \mathcal{D}(t_k, t_j) \end{array}$$

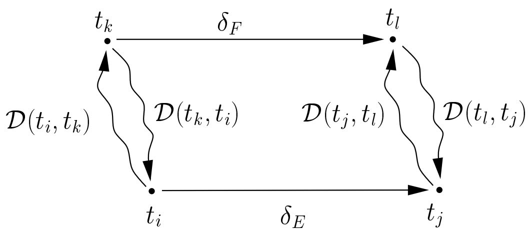
contradicting that $\mathcal{S}$ has no rigid components (cf. Definition 4.36). The case of $E$ lying along $P_3$ results in a similar contradiction. Thus, the assumption that there exists some $t_k$ such that $\mathcal{D}(t_i, t_k) + \mathcal{D}(t_k, t_j) = \delta$ must have been wrong.

On the other hand, if $E$ is a tight, dominated edge and $\mathcal{S}$ has no rigid components, then removing $E$ must not cause a change to the distance matrix of $\mathcal{S}$. Thus, there must exist some shortest path $P$ from $t_i$ to $t_j$ such that $|P| = \delta$. Since the edge from $t_i$ to $t_j$ has been removed, $P$ must contain some other point $t_k$. Thus, $\mathcal{D}(t_i, t_k) + \mathcal{D}(t_k, t_j) = \delta$. ■

**Theorem 4.46** *Suppose $E$ is a dominated edge in a consistent STN $\mathcal{S}$ in which there are no rigid components. Then removing any set $\{E_1, E_2, \ldots, E_n\}$ of dominated edges from $\mathcal{S}$ will not change the property of $E$ being a dominated edge in $\mathcal{S}$. Thus, the set of* undominated constraints *in $\mathcal{S}$ is a well-defined notion.*

**Proof** Let $E = (t_i, \delta_E, t_j)$ be a dominated edge in an STN $\mathcal{S}$ that has no rigid components. Suppose $F = (t_k, \delta_F, t_l)$ is some *other* dominated edge in $\mathcal{S}$, and that removing $F$ would cause $E$ to become undominated in $\mathcal{S}$. Since $E$ and $F$ are both dominated in $\mathcal{S}$, removing either edge alone would not cause a change in the distance matrix of $\mathcal{S}$. However, by assumption, removing $F$ would cause $E$ to become undominated. Thus, removing *both* $E$ and $F$ *would* cause a change to the distance matrix. Thus, removing only $E$ would cause $F$ to become undominated as well.

As a result, it must be that $F$ is part of every shortest path from $t_i$ to $t_j$ that does not include $E$. Similarly, $E$ is part of every shortest path from $t_k$ to $t_l$ that does not include $F$. This situation is pictured below.




Considering the paths $(t_i, t_k, t_l, t_j)$ and $(t_k, t_i, t_j, t_l)$, which are shortest paths, we get the following equations:

$$\begin{array}{rcl} \mathcal{D}(t_i, t_k) + \delta_F + \mathcal{D}(t_l, t_j) & = & \delta_E \\ \mathcal{D}(t_k, t_i) + \delta_E + \mathcal{D}(t_j, t_l) & = & \delta_F. \end{array}$$

Adding these together and canceling like quantities yields:

$$[\mathcal{D}(t_i, t_k) + \mathcal{D}(t_k, t_i)] \; + \; [\mathcal{D}(t_l, t_j) + \mathcal{D}(t_j, t_l)] \; = \; 0.$$

Since both of the bracketed quantities are necessarily non-negative in a consistent STN, we get that both bracketed quantities must in fact be equal to 0, contradicting that $\mathcal{S}$ has no rigid components.

Finally, suppose that removing the finite *sequence* of dominated edges $E_1, E_2, \ldots, E_n$ caused the status of some remaining edge $E$ to change from dominated to undominated. Let $E_i$ be the *first* edge in that sequence such that $E$ was dominated before $E_i$'s removal, but undominated thereafter. The preceding argument shows that this cannot happen. ■

**Definition 4.47** *The* undominated constraints *in an STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ are denoted $\mathcal{C}^u$.*

By Theorem 4.45, the following simple $O(|\mathcal{C}||\mathcal{T}|)$ algorithm will find the undominated constraints in an STN that has no rigid components: For each tight edge, $t_j - t_i \leq \delta,$ check whether there is some $t_k$ such that $\mathcal{D}(t_i, t_k) + \mathcal{D}(t_k, t_j) = \delta.$ If there is no such $t_k$, then the edge is undominated.

**Theorem 4.48** *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be an STN in which there are no rigid components; let $\mathcal{S}_d = (\mathcal{T}, \mathcal{C}_d)$ be its d-STN. Then the set $(\mathcal{C}_d)^u$ of undominated constraints in $\mathcal{C}_d$ is the unique, minimal set of constraints equivalent to the constraints in $\mathcal{C}$. (Here, "minimal" refers to the number of constraints in the set.)*

**Proof** *(Equivalence of $\mathcal{C}$ and $(\mathcal{C}_d)^u$)* By Proposition 4.29, $\mathcal{S}$ and $\mathcal{S}_d$ are equivalent (i.e., $\mathcal{S} \equiv \mathcal{S}_d$). Thus, $\mathcal{C} \equiv \mathcal{C}_d$. Furthermore, since $\mathcal{S}$ has no rigid components, neither does

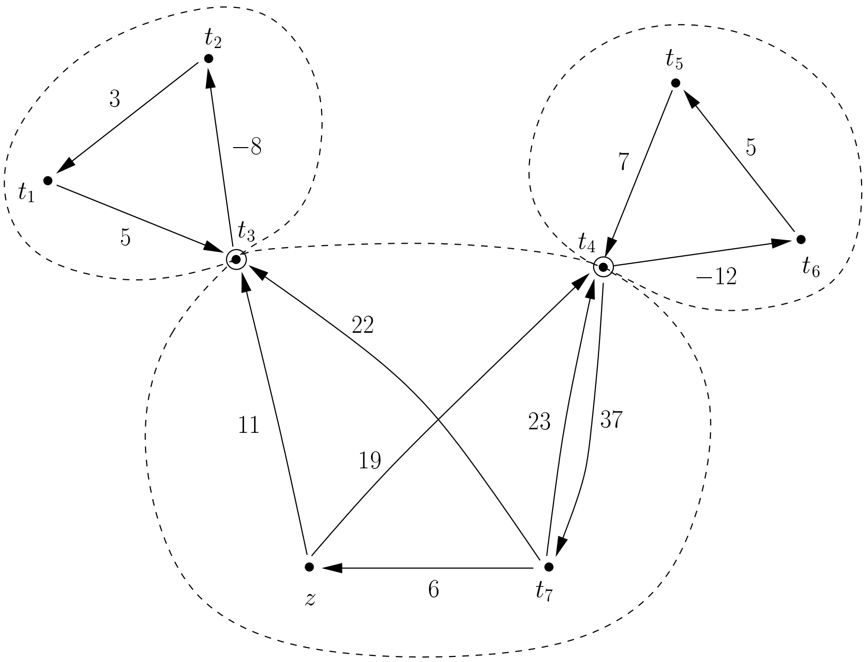



Figure 4.4: The STN from Figure 4.2 with its decoupled rigid components

$\mathcal{S}_d$. Thus, by Theorem 4.46, the set $(\mathcal{C}_d)^u$ of undominated constraints in $\mathcal{C}_d$ may be constructed by removing the dominated constraints from $\mathcal{C}_d$. Since removing dominated constraints does not affect the distance matrix (Definition 4.44), the distance matrices for $\mathcal{S}_d$ and $(\mathcal{T}, (\mathcal{C}_d)^u)$ are identical. Hence, by Corollary 4.28, $\mathcal{C}_d \equiv (\mathcal{C}_d)^u$. Thus, $(\mathcal{C}_d)^u \equiv \mathcal{C}$.

*(Minimality of* $(\mathcal{C}_d)^u$*)* Since removing any undominated constraint from $(\mathcal{C}_d)^u$ *would* change the distance matrix (Definition 4.44), and hence would cause the resulting constraint set to lose its equivalence with $\mathcal{C}$, the set $(\mathcal{C}_d)^u$ is a minimal constraint set equivalent to $\mathcal{C}$.

*(Uniqueness)* By Theorem 4.45, each edge $E = (t_j - t_i \leq \delta)$ in $(\mathcal{C}_d)^u$ constitutes the *only* shortest path from $t_i$ to $t_j$ in the fully-connected d-graph $\mathcal{G}_d$. Thus, replacing the edge $E$ with some alternative shortest path from $t_i$ to $t_j$ would necessarily require changing the distance matrix. ∎

Theorem 4.48, above, specifies the minimum number of constraints needed to entail the constraints in a network having no rigid components. Theorem 4.53, below, provides the analogous result for networks that have rigid components. It shows that the minimum number of edges needed to entail a network having rigid components can be found by first decoupling the rigid components from the rest of the network, as described in Section 4.1, and then providing minimal sets of edges to represent each of the resulting subnetworks.

Figure 4.3 from Section 4.1 shows an STN after its rigid components have been collapsed down to single points. Figure 4.4 shows the same network, but with the rigid components shown as decoupled subnetworks, henceforth called *rc-subnetworks*. Notice that rc-subnetworks, unlike proper STNs (cf. Definition 4.1), need not include the zero time-point variable, $z$. Notice also that the edges in each rc-subnetwork have been arranged into a loop, which the following result shows to be the most concise representation for a rigid component.

**Lemma 4.49** *If all of the time-points in a consistent STN* $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ *are rigidly connected and* $|\mathcal{T}| > 1$, *then* $|\mathcal{C}| \geq |\mathcal{T}|$. *Furthermore, there exists an equivalent STN* $\mathcal{S}' = (\mathcal{T}, \mathcal{C}')$ *such that* $|\mathcal{C}'| = |\mathcal{T}|$.

**Proof** Let $t_i \in \mathcal{T}$ be arbitrary. Since $|\mathcal{T}| > 1$, there exists some $t_j \in \mathcal{T}$ such that $t_i \neq t_j$. Since $t_i$ and $t_j$ are rigidly connected in $\mathcal{S}$, there is a shortest path in $\mathcal{S}$ from $t_i$ to $t_j$. Thus, there is at least one edge leaving $t_i$. Since each edge originates at a single point and each time-point has at least one edge emanating from it, there must be at least $|\mathcal{T}|$ edges in $\mathcal{C}$.

Now suppose $\mathcal{T} = \{t_0, t_1, \ldots, t_n\}$. Let $\mathcal{C}'$ be the following set of constraints:

$$\{\ (t_1 - t_0 \leq \mathcal{D}(t_0, t_1)),\ \ldots,\ (t_n - t_{n-1} \leq \mathcal{D}(t_{n-1}, t_n)),\ (t_0 - t_n \leq \mathcal{D}(t_n, t_0))\ \},$$

where $\mathcal{D}$ is the distance matrix for $\mathcal{S}$. Let $\mathcal{S}' = (\mathcal{T}, \mathcal{C}')$. Let $t_i, t_j \in \mathcal{T}$ be arbitrary such that $i < j$. The only path from $t_i$ to $t_j$ in $\mathcal{S}'$ that does not contain a loop is the path, $P_{ij} = (t_i, t_{i+1}, \ldots, t_{j-1}, t_j)$. Thus, $P_{ij}$ must be a shortest path in $\mathcal{S}'$. Similarly, the only path from $t_j$ to $t_i$ that does not contain a loop is the path, $P_{ji} = (t_j, t_{j+1}, \ldots, t_0, \ldots, t_{i-1}, t_i)$, which must therefore also be a shortest path. Next, notice that

$$|P_{ij}| = \mathcal{D}(t_i, t_{i+1}) + \ldots + \mathcal{D}(t_{j-1}, t_j),$$

which, by repeated application of the Triangle Inequality yields that $|P_{ij}| \geq \mathcal{D}(t_i, t_j)$. Similarly, $|P_{ji}| \geq \mathcal{D}(t_j, t_i)$.

Next, notice that

$$|P_{ij}| + |P_{ji}| = \mathcal{D}(t_0, t_1) + \mathcal{D}(t_1, t_2) + \ldots + \mathcal{D}(t_{n-1}, t_n) + \mathcal{D}(t_n, t_0).$$

Since each pair of time-points, $(t_p, t_q)$, is rigidly connected in $\mathcal{S}$, it follows that $\mathcal{D}(t_p, t_q) = -\mathcal{D}(t_q, t_p)$ (cf. Definition 4.36). Thus,

$$\begin{aligned} -|P_{ij}| - |P_{ji}| &= -\mathcal{D}(t_0, t_1) - \mathcal{D}(t_1, t_2) - \ldots - \mathcal{D}(t_{n-1}, t_n) - \mathcal{D}(t_n, t_0) \\ &= \mathcal{D}(t_1, t_0) + \mathcal{D}(t_2, t_1) + \ldots + \mathcal{D}(t_n, t_{n-1}) + \mathcal{D}(t_0, t_n). \end{aligned}$$

Since the right-hand side of each of the above equations represents the length of a loop in $\mathcal{S}$, which is a consistent STN, each must be non-negative (cf. Theorem 4.30). Thus, $|P_{ij}| + |P_{ji}|$ must be equal to zero. Thus,

$$\begin{array}{ll} |P_{ij}| = -|P_{ji}| & \\ \Rightarrow\ |P_{ij}| \leq -\mathcal{D}(t_j, t_i), & \text{Since } |P_{ji}| \geq \mathcal{D}(t_j, t_i) \\ \Rightarrow\ |P_{ij}| \leq \mathcal{D}(t_i, t_j), & \text{Since } \mathcal{D}(t_i, t_j) + \mathcal{D}(t_j, t_i) = 0 \\ \Rightarrow\ |P_{ij}| = \mathcal{D}(t_i, t_j), & \text{Since } \mathcal{D}(t_i, t_j) \leq |P_{ij}|. \end{array}$$

Similarly, $|P_{ji}| = \mathcal{D}(t_j, t_i)$. Since $P_{ij}$ is a shortest path in $\mathcal{S}'$, $\mathcal{D}'(t_i, t_j) = |P_{ij}|$; hence $\mathcal{D}'(t_i, t_j) = \mathcal{D}(t_j, t_i)$. Similarly, $\mathcal{D}'(t_j, t_i) = \mathcal{D}(t_j, t_i)$. Since $t_i$ and $t_j$ were arbitrary in $\mathcal{T}$ with $i < j$, $\mathcal{S}' \equiv \mathcal{S}$. Since the number of edges in $\mathcal{C}'$ is equal to $|\mathcal{T}|$, the result is proven. ∎

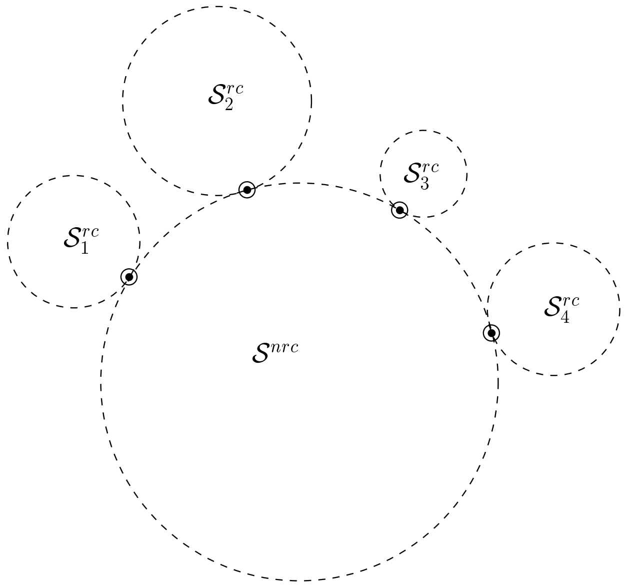



Figure 4.5: An STN with four decoupled rigid components

**Definition 4.50 (rc-Subnetworks)** *Let $\mathcal{S}$ be a consistent STN. Let $\mathcal{S}^*$ be* any *STN resulting from decoupling the rigid components of $\mathcal{S}$ as described in Section 4.1. Let*

$$\mathcal{S}_1^{rc} = (\mathcal{T}_1^{rc}, \mathcal{C}_1^{rc}), \quad \ldots, \quad \mathcal{S}_n^{rc} = (\mathcal{T}_n^{rc}, \mathcal{C}_n^{rc})$$

*be the subnetworks in $\mathcal{S}^*$ corresponding to the decoupled rigid components; and let $\mathcal{S}^{nrc} = (\mathcal{T}^{nrc}, \mathcal{C}^{nrc})$ be the remaining subnetwork of $\mathcal{S}$ from which the rigid components have been decoupled. The subnetworks, $\mathcal{S}_1^{rc} = (\mathcal{T}_1^{rc}, \mathcal{C}_1^{rc}), \ldots, \mathcal{S}_n^{rc} = (\mathcal{T}_n^{rc}, \mathcal{C}_n^{rc})$, are called* rc-subnetworks *of $\mathcal{S}^*$; $\mathcal{S}^{nrc}$ is called the* nrc-subnetwork *of $\mathcal{S}^*$.*

Figure 4.5 illustrates the notation from Definition 4.50. Notice that the nrc-subnetwork, $\mathcal{S}^{nrc}$, contains no rigid components, but that for each $i$, $\mathcal{T}^{nrc} \cap \mathcal{T}_i^{rc}$ is a singleton set containing the representative time-point for the rc-subnetwork $\mathcal{S}_i^{rc}$.

**Proposition 4.51** *If $P$ is a loop with length zero in a consistent STN $\mathcal{S}$, then the set of time-points occurring in $P$ forms a rigid component in $\mathcal{S}$.*

**Proof** Let $t_i$ and $t_j$ be arbitrary time-points in $P$. Let $P_{ij}$ be the subpath of $P$ from $t_i$ to $t_j$. Let $P_{ji}$ be the subpath in $P$ from $t_j$ to $t_i$. (If $t_i$ or $t_j$ occur more than once in $P$, choose $P_{ij}$ and $P_{ji}$ such that they concatenate to form $P$.) Since replacing $P_{ij}$ in $P$ with a strictly shorter path would create a loop in $\mathcal{S}$ with negative path-length, which would contradict that $\mathcal{S}$ is consistent, $P_{ij}$ must be a shortest path in $\mathcal{S}$. Similarly, $P_{ji}$ must be a shortest path in $\mathcal{S}$. Thus,

$$0 \;=\; |P_{ij}| + |P_{ji}| \;=\; \mathcal{D}(t_i, t_j) + \mathcal{D}(t_j, t_i).$$

Hence, $t_i$ and $t_j$ are rigidly connected. Since $t_i$ and $t_j$ were arbitrary time-points in $P$, the result is proven. ■

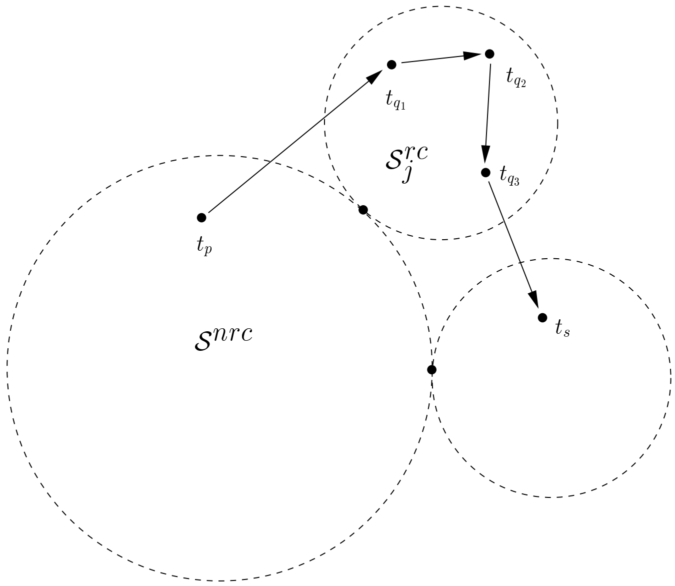



Figure 4.6: A sample path $P''$ from the proof of Theorem 4.53

**Corollary 4.52** *If $P$ is a shortest path from $t_i$ to $t_j$ in a consistent STN, where $t_i$ and $t_j$ are rigidly connected, then* all *of the time-points in $P$ belong to the rigid component containing $t_i$ and $t_j$.*

**Proof** Since $P$ is a shortest path from $t_i$ to $t_j$, $|P| = \mathcal{D}(t_i, t_j)$. Since $t_i$ and $t_j$ are rigidly connected, there is some shortest path $P'$ from $t_j$ to $t_i$ such that $|P'| = \mathcal{D}(t_j, t_i) = -\mathcal{D}(t_i, t_j)$. But then concatenating $P$ and $P'$ forms a loop with path-length zero. Thus, by Proposition 4.51, all of the time-points in $P$, including $t_i$ and $t_j$, belong to the same rigid component. ■

**Theorem 4.53** *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be a consistent STN. Let $\mathcal{S}_1^{rc}, \ldots, \mathcal{S}_n^{rc}$ and $\mathcal{S}^{nrc}$ be the rc-subnetworks of $\mathcal{S}$ (cf. Definition 4.50) such that each rc-subnetwork $\mathcal{S}_i^{rc}$ has the minimum number of edges (cf. Lemma 4.49). Let $\mathcal{S}' = (\mathcal{T}, \mathcal{C}')$ be an arbitrary STN that is equivalent to $\mathcal{S}$. Then*

$$|\mathcal{C}'| \;\geq\; |\mathcal{T}_1^{rc}| + \ldots + |\mathcal{T}_n^{rc}| + |(\mathcal{C}^{nrc})_d^u|.$$

**Proof** First, consider an arbitrary rigid component $\mathcal{T}_i^{rc}$ in $\mathcal{S}$. Let $P$ be an arbitrary shortest path in $\mathcal{S}'$ both of whose endpoints are in $\mathcal{T}_i^{rc}$. By Corollary 4.52, all of the time-points in $P$ are necessarily in $\mathcal{T}_i^{rc}$. As a result, the edges in $\mathcal{C}'$ that entail the constraints in the rigid components $\mathcal{T}_1^{rc}, \mathcal{T}_2^{rc}, \ldots, \mathcal{T}_n^{rc}$, necessarily fall into *disjoint sets*.

Next, let $P'$ be a shortest path in $\mathcal{S}'$ both of whose endpoints are in $\mathcal{S}^{nrc}$. Suppose $P'$ contains a subpath, $P'' = (t_p, t_{q_1}, t_{q_2}, \ldots, t_{q_k}, t_s)$, where $t_p \in \mathcal{T}^{nrc}$, $k \geq 1$, $t_{q_1}, t_{q_2}, \ldots, t_{q_k} \in \mathcal{T}_j^{rc}$, for some $j$, and $t_s \notin \mathcal{T}_j^{rc}$. Figure 4.6 illustrates such a path in the case where $k = 3$ and $t_s$ happens to lie in some other rc-subnetwork. Let $E_1$ and $E_2$ be the first and last edges in $P''$. Then, $E_1$ and $E_2$ have the following form:

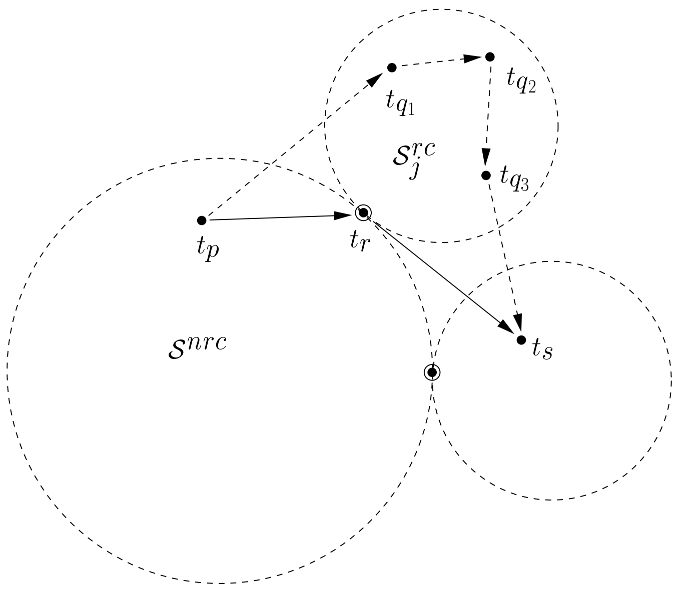



Figure 4.7: Modifying the constraint set $\mathcal{C}'$ in the proof of Theorem 4.53

$$E_1\colon t_{q_1} - t_p \leq \delta_1 \quad \text{and} \quad E_2\colon t_s - t_{q_k} \leq \delta_2.$$

If $t_{q_1}$ is not the representative $t_r$ of the rigid component, $\mathcal{T}_j^{rc}$, then $E_1$ may be replaced by the edge,

$$E_1'\colon t_r - t_p \leq (\delta_1 + \mathcal{D}(t_q, t_r)),$$

as described in Section 4.1. Similarly, if $t_{q_k}$ is not the representative of the component, then $E_2$ may be replaced in $\mathcal{C}'$ by the edge,

$$E_2'\colon t_s - t_r \leq (\delta_2 - \mathcal{D}(t_q, t_r)).$$

Modifying the contents of the set $\mathcal{C}'$ in this way, as illustrated in Figure 4.7, does not change the set of solutions to the network; nor does it change the number of edges in $\mathcal{C}'$. However, the subpath $P''$ may be replaced (in $P'$) by the subpath $(t_p, t_r, t_s)$, without affecting the length of $P'$. However, $t_p$ and $t_r$ are time-points in $\mathcal{T}^{nrc}$. Furthermore, if $t_s$ is not in $\mathcal{T}^{nrc}$, then the above argument may be applied to the subpath beginning at $t_r$. Thus, by repeated applications of the above argument, the original path $P'$ may be converted to an equivalent path.

Similar arguments apply to any shortest path from a time-point in $\mathcal{T}^{nrc}$ to a non-representative time-point in some rigid component, and to any shortest path joining non-representative time-points from two different rigid components. In each case, any edge that joins a non-representative time-point from some rigid component to a time-point lying outside that rigid component is replaced by an edge involving the representative of that rigid component, as described in Section 4.1. Since doing so does not affect the intra-component edges in $\mathcal{C}'$, does not affect the set of solutions to the network, and does not change the number of edges in $\mathcal{C}'$, the net result is that the edges in the modified version of $\mathcal{C}'$ fall into disjoint sets: one set for each rigid component and one set for the time-points in $\mathcal{T}^{nrc}$. Since the minimum number of edges required to entail the constraints in each rigid component $\mathcal{T}_i^{rc}$ is $|\mathcal{T}_i^{rc}|$ (by Lemma 4.49), and since the minimum number of edges required to entail the constraints in $\mathcal{S}^{nrc}$ is $|(\mathcal{C}^{nrc})_d^u|$ (by Theorem 4.48), the result is proven. ■

# Chapter 5

# The Temporal Decoupling Problem

This chapter addresses an important family of temporal reasoning problems centered on finding ways of decoupling portions of a temporal network. Variations on the basic theme of temporal decoupling apply to several important problems in collaborative, multi-agent planning. For each variation, the problem is formally defined, theorems are presented characterizing solutions, and sound and complete algorithms are presented. Direct applications of these algorithms to problems related to the combinatorial auction-based ICDP mechanism described in Chapter 3 are discussed throughout this chapter.

## 5.1 Introduction

**The Basic Temporal Decoupling Problem.** The basic idea of temporal decoupling is to partition a temporal network into independent subnetworks. The characteristic property of such a decoupling is the Mergeable Solutions Property which says that any solutions to the independent subnetworks may be merged into a solution for the global network. If a given network is not initially decoupled into independent subnetworks, constraints must be added to achieve the decoupling. The temporal decoupling problem (TDP) is to find a set of constraints that achieves such a decoupling without needlessly constraining the network. If agents working on temporally dependent tasks in some group activity temporally decouple those tasks, then they may work independently, without requiring further coordination, thereby drastically reducing the need for negotiation about temporal dependencies, while costing the agents some amount of flexibility in their individual schedules. Thus, the basic TDP algorithm is directly applicable to the Post-Auction Coordination Problem described in Section 3.5.

**The Allocative TDP.** A slight variant of the basic TDP, called the Allocative TDP, is applicable to the situation faced by an agent wanting to participate in a task-allocation auction of the sort used in the ICDP mechanism described in Chapter 3. To keep its bid-generation computations manageable, the agent needs to restrict the portion of its schedule of pre-existing commitments that it uses as the basis for its bid-generation computations. To do so, the agent must decouple that portion of its schedule. The decoupled portion is called the agent's Auction-Participation Context. For the problem of generating a suitable

APC, there are two subnetworks to consider: $\mathcal{S}_X$, which corresponds to the portion of its schedule to be associated with the APC, and $\mathcal{S}_Y$, which corresponds to the rest of the agent's schedule. Decoupling these subnetworks enables the agent to deal with the rest of its schedule independently of the auction. For example, it need not consider the complicated issues associated with having outstanding bids when considering whether to add a new task to the rest of its schedule.

In the Allocative TDP, the contents of the subnetworks to be decoupled (i.e., the sets of time-points associated with $\mathcal{S}_X$ and $\mathcal{S}_Y$) are not known in advance. An algorithm for the Allocative TDP can be applied directly to the APC-Generation Problem described above. The Allocative TDP algorithm is an iterative algorithm that interleaves the allocation of time-points to the subnetworks with the addition of new constraints aimed at decoupling the subnetworks. The Allocative TDP algorithm is *lazy* in the sense that it does not establish the Mergeable Solutions Property at the end of each iteration; instead, it establishes it only at the end of the final iteration.

**Relative Temporal Decoupling.** When agents are working on temporally dependent tasks in a group activity, they may not want to accept the degree of extra constrainedness necessary to decouple the global network into completely independent subnetworks. One way to give agents greater flexibility, while still reducing the need for coordination, is to fully decouple the tasks being done by *some* of the agents. In this case, the subnetworks being decoupled do not cover all the time-points in the global network. The time-points that do not belong to any of the subnetworks being decoupled are collectively called the *relativizing set* $\mathcal{T}_W$. The resulting decoupling is called a temporal decoupling *relative* to $\mathcal{T}_W$. The characteristic Mergeable Solutions Property for a relative temporal decoupling says that the merger of any solutions for the decoupled subnetworks may be extended into a solution for the global network.

Another application of the relative temporal decoupling problem is to the problem of bid generation in a task allocation auction. When an agent places a bid on a set of tasks in such an auction, it is allowed to include temporal constraints with that bid. The purpose of the bid's temporal constraints is to allow the agent to protect the feasibility of its private schedule of pre-existing commitments should the bid eventually be awarded. The temporal-constraint-generation problem is, in effect, an instance of the relative temporal decoupling problem in which there is only one subnetwork $\mathcal{S}_1$ being decoupled relative to a leftover set of time-points. This case is not trivial given that the Mergeable Solutions Property must hold. The subnetwork $\mathcal{S}_1$ represents the time-points associated with the group activity (including the tasks on which the agent is bidding). The relativizing set $\mathcal{T}_W$ covers time-points associated with the rest of the agent's auction-participation context.

**Adding new constraints to the relativizing set.** An agent controlling a decoupled subnetwork may operate independently; however, this implies that an agent controlling the leftover time-points in the relativizing set $\mathcal{T}_W$ cannot be allowed to add any constraints that, upon propagation through the network, would add any extra constrainedness to any of the decoupled subnetworks. For example, suppose an agent generates temporal constraints for a bid in a task-allocation auction as described above. While the bid is outstanding, if that agent wants to add temporal constraints to the rest of the time-points in its auction-participation context (e.g., corresponding to a new commitment it might want to adopt), it must be sure that those new constraints do not jeopardize its ability to honor the outstanding bid should that bid eventually be awarded. Similarly, in the case of multiple, decoupled subnetworks corresponding to tasks being done by different agents, if $G_W$ is an agent controlling the leftover time-points in the relativizing set $\mathcal{T}_W$, then $G_W$ cannot be allowed to add constraints to the time-points in $\mathcal{T}_W$ if doing so would cause any additional constrainedness in the decoupled subnetworks.

The standard approach for determining the bounds on how much constraints in an STN can be tightened (cf. Theorem 4.33) does not accommodate the extra requirement that the tightening of constraints not impose any additional constrainedness on the decoupled subnetwork(s). This chapter presents *lambda bounds* that an agent can use to accommodate this extra requirement. Although the lambda bounds are more complicated than those in the standard case (which derive from the negative-transpose entries in the distance matrix), they can be computed in polynomial time. The lambda bounds are proven to be necessary and sufficient for the purposes described.

**Hierarchical partial temporal decoupling.** To allow even greater flexibility, a hierarchy of partially decoupled subnetworks may be constructed by recursively applying the relative temporal decoupling algorithm to a given network. This can be used to allow agents to have varying degrees of flexibility and dependence in a group activity. This chapter shows how to construct such hierarchies. In addition, it shows that the lambda bounds described above are applicable to the hierarchical situation as well.

The rest of this chapter is organized as follows. Section 5.2 formally defines the basic Temporal Decoupling Problem in the case of two subnetworks. Section 5.3 treats the Allocative TDP, also restricted to the case of two subnetworks, and shows how an algorithm for solving the Allocative TDP can be applied to the Auction-Participation-Context-Generation Problem. Section 5.4 extends the treatment of the basic Temporal Decoupling Problem to the case of arbitrarily many subnetworks and shows how an algorithm for the General TDP may be applied to the Post-Auction-Coordination Problem. Section 5.5 formally addresses the problem of decoupling subnetworks relative to some leftover set of time-points. That section formally derives the *lambda bounds* needed by agents controlling the leftover time-points in a relative temporal decoupling and discusses applications of the Relative TDP algorithm and the lambda bounds to the Temporal-Constraint-Generation and Post-Auction-Coordination Problems. Section 5.6 discusses related work.

## 5.2 The Temporal Decoupling Problem (Case $n = 2$)

This section presents the temporal decoupling problem in the case of two subnetworks (which shall be called $\mathcal{S}_X$ and $\mathcal{S}_Y$). (The general case of arbitrarily many subnetworks is treated in Section 5.4.) Section 5.2.1 formally defines the temporal decoupling problem. Section 5.2.2 presents theorems specifying necessary and sufficient characterizations

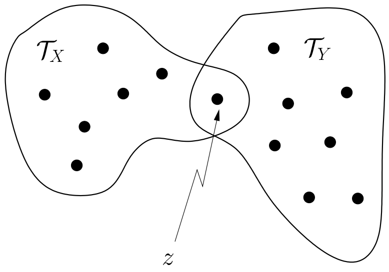



Figure 5.1: A Sample z-Partition

of solutions to instances of the TDP. Sections 5.2.3 and 5.2.4 present sound and complete algorithms for solving instances of the TDP, which are then evaluated experimentally in Section 5.2.5. Section 5.2.6 presents a formulation of the TDP in which solutions are restricted to locally optimal decouplings, called *minimal temporal decouplings*. Section 5.2.7 then recasts the basic TDP as an optimization problem in which a decoupling is sought that maximizes some measure of goodness.

### 5.2.1 Formal Definition of the TDP

We begin by defining a *z-partition* of a set of time-points. A z-partition is similar to an ordinary partition, except that every subset in the z-partition is required to include the special zero time-point $z$ (since every STN includes $z$).

**Definition 5.1 (z-Partition)** *Let $\mathcal{T}$, $\mathcal{T}_X$ and $\mathcal{T}_Y$ be sets of time-point variables. We say that $\mathcal{T}_X$ and $\mathcal{T}_Y$* z-partition *$\mathcal{T}$ if:*

- $\mathcal{T}_X \cap \mathcal{T}_Y = \{z\}$ *and*
- $\mathcal{T}_X \cup \mathcal{T}_Y = \mathcal{T}$.

Figure 5.1 shows a sample z-partition of a set of time-point variables.

**Definition 5.2 (Temporal Decoupling)** *A* temporal decoupling *of the STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ is a pair of STNs, $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$, such that:*

- *$\mathcal{S}_X$ and $\mathcal{S}_Y$ are consistent;*
- *$\mathcal{T}_X$ and $\mathcal{T}_Y$ z-partition $\mathcal{T}$; and*
- *(Mergeable Solutions Property) Merging* any *solutions for $\mathcal{S}_X$ and $\mathcal{S}_Y$ necessarily yields a solution for $\mathcal{S}$.*

*To highlight the particular z-partition, we may say that $\mathcal{S}_X$ and $\mathcal{S}_Y$ are a temporal decoupling of $\mathcal{S}$* using *the z-partition $(\mathcal{T}_X, \mathcal{T}_Y)$.*

**Lemma 5.3** *For $\mathcal{S}_X$ and $\mathcal{S}_Y$ to be a temporal decoupling of $\mathcal{S}$, it is necessary for $\mathcal{S}$ to be consistent.*

**Proof** If $\mathcal{S}_X$ and $\mathcal{S}_Y$ are a temporal decoupling of $\mathcal{S}$, then, by Definition 5.2, both $\mathcal{S}_X$ and $\mathcal{S}_Y$ are consistent. Thus, each has at least one solution. By the Mergeable Solutions Property, the merging of any such solutions yields a solution to $\mathcal{S}$. Thus, $\mathcal{S}$ is consistent. ■

**Lemma 5.4** *If $\mathcal{S}_X$ and $\mathcal{S}_Y$ are a temporal decoupling of $(\mathcal{T}, \mathcal{C} \cup \mathcal{C}')$, then $\mathcal{S}_X$ and $\mathcal{S}_Y$ are also a temporal decoupling of $(\mathcal{T}, \mathcal{C})$.*

**Proof** Any solution for $(\mathcal{T}, \mathcal{C} \cup \mathcal{C}')$ is necessarily a solution for $(\mathcal{T}, \mathcal{C})$. ■

**Lemma 5.5** *If $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$ are a temporal decoupling of $(\mathcal{T}, \mathcal{C})$, then $\mathcal{S}_X$ and $\mathcal{S}_Y$ are also a temporal decoupling of $(\mathcal{T}, \mathcal{C} \cup \mathcal{C}_X \cup \mathcal{C}_Y)$.*

**Proof** Let $\mathcal{X}$ and $\mathcal{Y}$ be arbitrary solutions for $\mathcal{S}_X$ and $\mathcal{S}_Y$, respectively. Thus, the variable assignments in $\mathcal{X}$ and $\mathcal{Y}$ satisfy the constraints in $\mathcal{C}_X$ and $\mathcal{C}_Y$, respectively. If $\mathcal{S}_X$ and $\mathcal{S}_Y$ are a temporal decoupling of $(\mathcal{T}, \mathcal{C})$, then $\mathcal{X} \cup \mathcal{Y}$ must be a solution for $(\mathcal{T}, \mathcal{C})$. Thus, the variable assignments in $\mathcal{X} \cup \mathcal{Y}$ satisfy the constraints in $\mathcal{C}$. Since the constraints in $\mathcal{C}_X$ do not constrain the variables in $\mathcal{T}_Y$, and the constraints in $\mathcal{C}_Y$ do not constrain the variables in $\mathcal{T}_X$, the variable assignments in $\mathcal{X} \cup \mathcal{Y}$ satisfy the constraints in $\mathcal{C} \cup \mathcal{C}_X \cup \mathcal{C}_Y$. In other words, $\mathcal{X} \cup \mathcal{Y}$ is a solution for $(\mathcal{T}, \mathcal{C} \cup \mathcal{C}_X \cup \mathcal{C}_Y)$. Since the solutions $\mathcal{X}$ and $\mathcal{Y}$ were arbitrary, $\mathcal{S}_X$ and $\mathcal{S}_Y$ are a temporal decoupling of $(\mathcal{T}, \mathcal{C} \cup \mathcal{C}_X \cup \mathcal{C}_Y)$. ■

**Note.** In some applications (e.g., in the temporal-constraint-generation problem discussed in Section 5.5.6), it is convenient to replace the Mergeable Solutions Property with an equivalent property called the *Mergeable Constraints Property*. The Mergeable Constraints Property says that no matter how the agents subsequently constrain their decoupled subnetworks, the global network $\mathcal{S}$ will remain consistent.

**Definition 5.6 (Mergeable Constraints Property)** *Let $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$ be consistent STNs such that $\mathcal{T}_X$ and $\mathcal{T}_Y$ z-partition a set of time-points $\mathcal{T}$. We say that $\mathcal{S}_X$ and $\mathcal{S}_Y$ have the* Mergeable Constraints Property *in the STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ if for any sets of constraints $\Delta\mathcal{C}_X$ and $\Delta\mathcal{C}_Y$ over $\mathcal{T}_X$ and $\mathcal{T}_Y$, respectively, if $\Delta\mathcal{C}_X$ and $\Delta\mathcal{C}_Y$ are consistent with $\mathcal{S}_X$ and $\mathcal{S}_Y$, respectively, then their union $\Delta\mathcal{C}_X \cup \Delta\mathcal{C}_Y$ is consistent with $\mathcal{S}$.*

**Theorem 5.7 (MSP $\equiv$ MCP)** *Let $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$ be consistent STNs such that $\mathcal{T}_X$ and $\mathcal{T}_Y$ z-partition a set of time-points $\mathcal{T}$. $\mathcal{S}_X$ and $\mathcal{S}_Y$ have the Mergeable Solutions Property (MSP) in the STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ if and only if they have the Mergeable Constraints Property (MCP) in $\mathcal{S}$.*

**Proof** ($\Rightarrow$) Suppose $\mathcal{S}_X$ and $\mathcal{S}_Y$ have the Mergeable Solutions Property in $\mathcal{S}$. Let $\Delta\mathcal{C}_X$ and $\Delta\mathcal{C}_Y$ be arbitrary sets of constraints over the time-points in $\mathcal{T}_X$ and $\mathcal{T}_Y$, respectively, such that $\Delta\mathcal{C}_X$ and $\Delta\mathcal{C}_Y$ are consistent with $\mathcal{S}_X$ and $\mathcal{S}_Y$, respectively. By Definition 4.9, $\mathcal{S}_X^+ = (\mathcal{T}_X, \mathcal{C}_X \cup \Delta\mathcal{C}_X)$ and $\mathcal{S}_Y^+ = (\mathcal{T}_Y, \mathcal{C}_Y \cup \Delta\mathcal{C}_Y)$ are consistent STNs. Let $\mathcal{X}$ and $\mathcal{Y}$

be arbitrary solutions for $\mathcal{S}_X^+$ and $\mathcal{S}_Y^+$, respectively. Since the constraints in $\Delta\mathcal{C}_X$ do not constrain the variables in $\mathcal{T}_Y$ and, similarly, the constraints in $\Delta\mathcal{C}_Y$ do not constrain the variables in $\mathcal{T}_X$, we have that the variable assignments in $\mathcal{X} \cup \mathcal{Y}$ satisfy all the constraints in $\Delta\mathcal{C}_X \cup \Delta\mathcal{C}_Y$.

Now, since $\mathcal{X}$ and $\mathcal{Y}$ are (also) solutions for $\mathcal{S}_X$ and $\mathcal{S}_Y$, respectively, the Mergeable Solutions Property gives us that $\mathcal{X} \cup \mathcal{Y}$ must be a solution for $\mathcal{S}$, whence the variable assignments in $\mathcal{X} \cup \mathcal{Y}$ also satisfy all the constraints in $\mathcal{C}$. Thus, the variable assignments in $\mathcal{X} \cup \mathcal{Y}$ satisfy all the constraints in $\mathcal{C} \cup \Delta\mathcal{C}_X \cup \Delta\mathcal{C}_Y$, which implies that $(\mathcal{T}, \mathcal{C} \cup \Delta\mathcal{C}_X \cup \Delta\mathcal{C}_Y)$ is a consistent STN. Thus, by Definition 4.9, the constraints in $\Delta\mathcal{C}_X \cup \Delta\mathcal{C}_Y$ are consistent with $\mathcal{S} = (\mathcal{T}, \mathcal{C})$.

($\Leftarrow$) Suppose $\mathcal{S}_X$ and $\mathcal{S}_Y$ have the Mergeable Constraints Property in $\mathcal{S}$. Let $\mathcal{X}$ and $\mathcal{Y}$ be arbitrary solutions for $\mathcal{S}_X$ and $\mathcal{S}_Y$, respectively. These solutions may be represented as sets of constraints $\Delta\mathcal{C}_X$ and $\Delta\mathcal{C}_Y$, respectively, by transforming each variable assignment $t_i = w_i$ into a pair of constraints: $t_i - z \leq w_i$ and $z - t_i \leq -w_i$. Thus, by the Mergeable Constraints Property, the union $\Delta\mathcal{C}_X \cup \Delta\mathcal{C}_Y$ is consistent with $\mathcal{S}$. But then the variable assignments in $\mathcal{X} \cup \mathcal{Y}$ constitute a solution for $\mathcal{S}$. ■

The following definition of the Temporal Decoupling Problem does not include a metric of solution *quality*. Section 5.2.7 presents a formulation of the TDP as an optimization problem in terms of such a metric.

**Definition 5.8 (The Temporal Decoupling Problem)** *Given an STN $\mathcal{S}$ whose time-points are z-partitioned by $\mathcal{T}_X$ and $\mathcal{T}_Y$, find sets of constraints $\mathcal{C}_X$ and $\mathcal{C}_Y$ such that the STNs $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$ temporally decouple $\mathcal{S}$.*

**Note.** Upon finding sets $\mathcal{C}_X$ and $\mathcal{C}_Y$ such that $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$ temporally decouple $\mathcal{S} = (\mathcal{T}, \mathcal{C})$, it is typically desirable to add the constraint sets $\mathcal{C}_X$ and $\mathcal{C}_Y$ to $\mathcal{S}$. After all, if the time-points in $\mathcal{T}_X$ and $\mathcal{T}_Y$ are in fact constrained by $\mathcal{C}_X$ and $\mathcal{C}_Y$, there is nothing to gain from failing to represent this information in $\mathcal{S}$. Lemma 5.5 shows that if $\mathcal{S}_X$ and $\mathcal{S}_Y$ temporally decouple $(\mathcal{T}, \mathcal{C})$, then they also temporally decouple $(\mathcal{T}, \mathcal{C} \cup \mathcal{C}_X \cup \mathcal{C}_Y)$.

**Lemma 5.9** *Any instance of the TDP in which $\mathcal{S}$ is consistent has a solution.*

**Proof** Let $\mathcal{S}$ be a consistent STN whose time-points are z-partitioned by $\mathcal{T}_X$ and $\mathcal{T}_Y$. Let $\mathcal{X}$ be an arbitrary solution for $\mathcal{S}$. Then the following specifies a temporal decoupling of $\mathcal{S}$ using $\mathcal{T}_X$ and $\mathcal{T}_Y$ in which each time-point in $\mathcal{S}$ is fixed to its value in the solution $\mathcal{X}$.

$$\begin{aligned}
\mathcal{C}_X &= \{\, (x - z \leq v) : (x = v) \in \mathcal{X}, x \in \mathcal{T}_X \,\} \cup \{\, (z - x \leq -v) : (x = v) \in \mathcal{X}, x \in \mathcal{T}_X \,\} \\
\mathcal{C}_Y &= \{\, (y - z \leq w) : (y = w) \in \mathcal{X}, y \in \mathcal{T}_Y \,\} \cup \{\, (z - y \leq -w) : (y = w) \in \mathcal{X}, y \in \mathcal{T}_Y \,\}
\end{aligned}$$

■

We call such decouplings *rigid decouplings*. One problem with rigid decouplings is that the subnetworks $\mathcal{S}_X$ and $\mathcal{S}_Y$ are completely rigid (i.e., completely inflexible). Theorems 5.10 and 5.12 below provide necessary and sufficient characterizations of solutions to the TDP that point the way to TDP algorithms that yield more flexible decoupled subnetworks.

### 5.2.2 Necessary and Sufficient Characterizations of TDP Solutions

This section presents theorems specifying necessary and sufficient conditions for a temporal decoupling. The conditions are expressed in terms of inequalities involving the distance matrices for the subnetworks $\mathcal{S}_X$ and $\mathcal{S}_Y$, and for the global network $\mathcal{S}$. Each entry in the distance matrix for $\mathcal{S}$ serves as an upper bound for various expressions involving the distance matrices of the subnetworks. Thus, a wide range of TDP solutions is possible. In contrast, Section 5.2.6 defines a *minimal* temporal decoupling which is optimal in the sense that any non-trivial weakening of an intra-subnetwork constraint (i.e., a constraint in $\mathcal{C}_X$ or $\mathcal{C}_Y$) would ruin the decoupling.

**Theorem 5.10 (Necessary Conditions)** *If the STNs $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$ are a temporal decoupling of the STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$, then the following properties must hold:*

*(Property $1_X$)* $\mathcal{D}_X(x_i, x_j) \leq \mathcal{D}(x_i, x_j)$, *for all* $x_i, x_j \in \mathcal{T}_X$;

*(Property $1_Y$)* $\mathcal{D}_Y(y_i, y_j) \leq \mathcal{D}(y_i, y_j)$, *for all* $y_i, y_j \in \mathcal{T}_Y$;

*(Property $2_{XY}$)* $\mathcal{D}_X(x, z) + \mathcal{D}_Y(z, y) \leq \mathcal{D}(x, y)$, *for all* $x \in \mathcal{T}_X$ *and* $y \in \mathcal{T}_Y$; *and*

*(Property $2_{YX}$)* $\mathcal{D}_Y(y, z) + \mathcal{D}_X(z, x) \leq \mathcal{D}(y, x)$, *for all* $x \in \mathcal{T}_X$ *and* $y \in \mathcal{T}_Y$,

*where $\mathcal{D}_X$, $\mathcal{D}_Y$, and $\mathcal{D}$ are the distance matrices for $\mathcal{S}_X$, $\mathcal{S}_Y$ and $\mathcal{S}$, respectively.*

*If, in addition, Properties $1_X$ and $1_Y$ hold with equality in all instances (i.e., $\mathcal{S}_X \equiv \mathcal{S}_d|_{\mathcal{T}_X}$ and $\mathcal{S}_Y \equiv \mathcal{S}_d|_{\mathcal{T}_Y}$), then Properties $2_{XY}$ and $2_{YX}$ also hold with equality in all instances.*

Properties $1_X$ and $1_Y$ are illustrated in Figure 5.2. Properties $2_{XY}$ and $2_{YX}$ are illustrated in Figures 5.3 and 5.4.

**Proof** Let $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$ be an arbitrary temporal decoupling of the STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$. By definition of a temporal decoupling, $\mathcal{S}_X$ and $\mathcal{S}_Y$ are both consistent.

(Property $1_X$) Let $x_i, x_j \in \mathcal{T}_X$ be arbitrary. By Theorem 4.33, there is a solution $\mathcal{X}$ for $\mathcal{S}_X$ in which $x_j - x_i = \mathcal{D}_X(x_i, x_j)$. Let $\mathcal{Y}$ be an arbitrary solution for $\mathcal{S}_Y$. Since $\mathcal{S}_X$ and $\mathcal{S}_Y$ are a temporal decoupling of $\mathcal{S}$, merging the solutions $\mathcal{X}$ and $\mathcal{Y}$ yields a solution for $\mathcal{S}$. In that solution, $x_j - x_i = \mathcal{D}_X(x_i, x_j)$. In addition, being a solution for $\mathcal{S}$ implies that $x_j - x_i \leq \mathcal{D}(x_i, x_j)$. Thus,

$$\mathcal{D}_X(x_i, x_j) \;=\; x_j - x_i \;\leq\; \mathcal{D}(x_i, x_j).$$

(Property $2_{XY}$) Let $x \in \mathcal{T}_X$ and $y \in \mathcal{T}_Y$ be arbitrary. Let $\mathcal{X}$ be a solution for $\mathcal{S}_X$ in which $z - x = \mathcal{D}_X(x, z)$. Similarly, let $\mathcal{Y}$ be a solution for $\mathcal{S}_Y$ in which $y - z = \mathcal{D}_Y(z, y)$. Since $\mathcal{S}_X$ and $\mathcal{S}_Y$ are a temporal decoupling of $\mathcal{S}$, merging the solutions $\mathcal{X}$ and $\mathcal{Y}$ yields a solution for $\mathcal{S}$. In this solution,

$$y - x \;=\; (z - x) + (y - z) \;=\; \mathcal{D}_X(x, z) + \mathcal{D}_Y(z, y).$$

In addition, being a solution for $\mathcal{S}$ implies that $y - x \leq \mathcal{D}(x, y)$. Thus,

$$\mathcal{D}_X(x, z) + \mathcal{D}_Y(z, y) \;=\; y - x \;\leq\; \mathcal{D}(x, y).$$

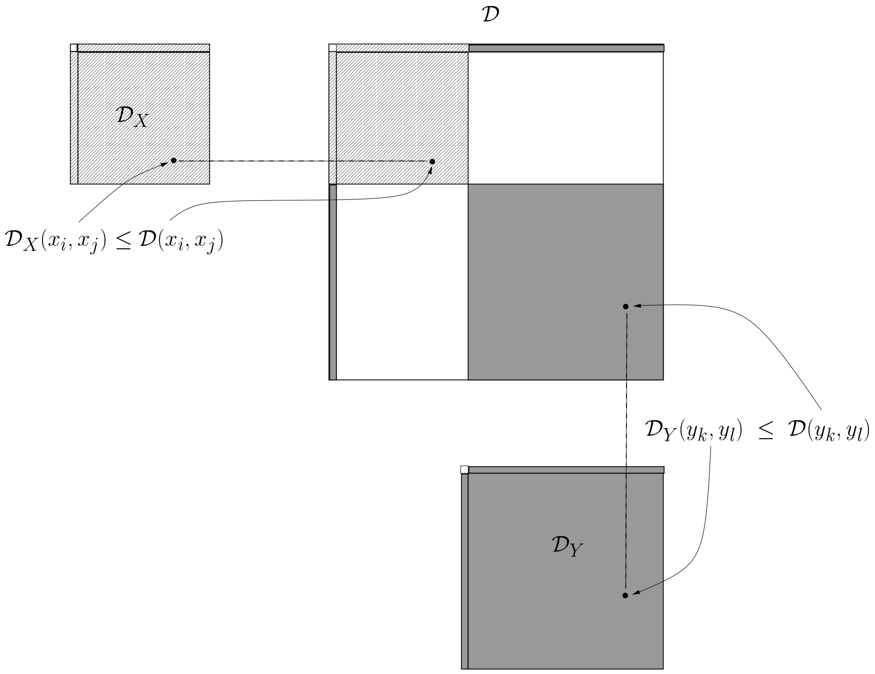



Figure 5.2: Illustration of Properties $1_X$ and $1_Y$ from Theorem 5.10

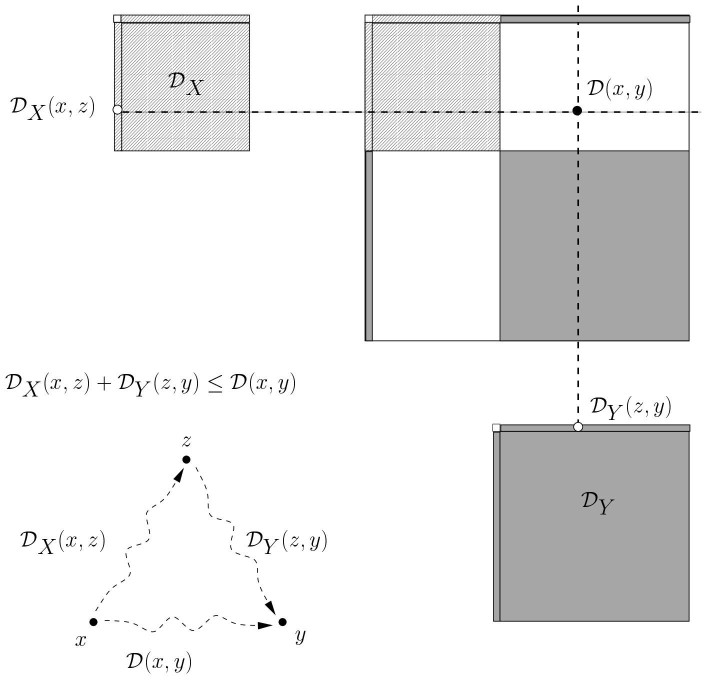



Figure 5.3: Illustration of Property $2_{XY}$ from Theorem 5.10

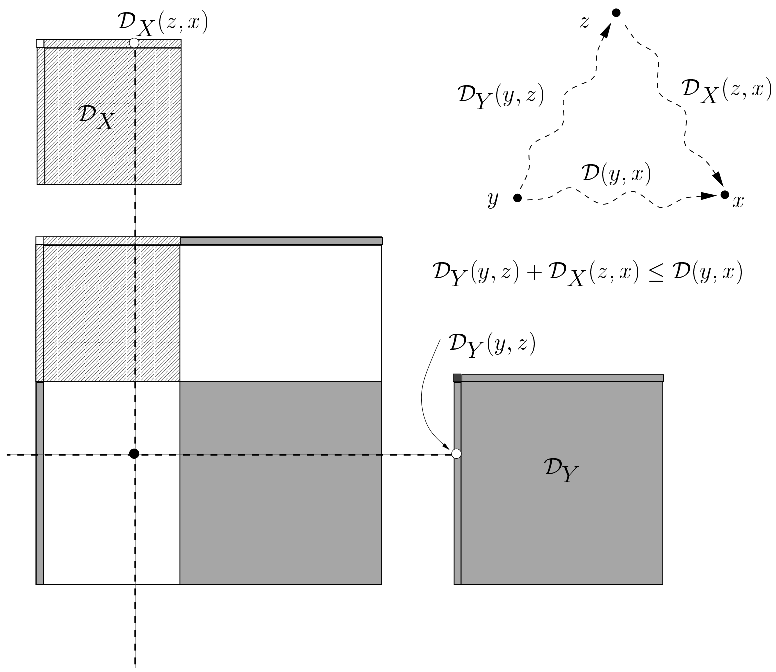



Figure 5.4: Illustration of Property $2_{YX}$ from Theorem 5.10

Properties $1_Y$ and $2_{YX}$ are handled analogously.

If Properties $1_X$ and $1_Y$ hold with equality in all instances, then Property $2_{XY}$ becomes:

$$\mathcal{D}(x,z) + \mathcal{D}(z,y) \;\le\; \mathcal{D}(x,y), \text{ for all } x \in \mathcal{T}_X \text{ and } y \in \mathcal{T}_Y.$$

However, by the triangle inequality:

$$\mathcal{D}(x,z) + \mathcal{D}(z,y) \ge \mathcal{D}(x,y), \text{ for all } x \in \mathcal{T}_X \text{ and } y \in \mathcal{T}_Y.$$

Thus, equality obtains in Property $2_{XY}$. Property $2_{YX}$ is similarly affected.

Finally, if Property $1_X$ holds with equality in all instances then $\mathcal{D}_X = \mathcal{D}|_{\mathcal{T}_X}$. Since $\mathcal{D} = \mathcal{D}_d$ (by Proposition 4.29), $\mathcal{D}|_{\mathcal{T}_X} = \mathcal{D}_d|_{\mathcal{T}_X}$. Thus, $\mathcal{D}_X = \mathcal{D}_d|_{\mathcal{T}_X}$. Since $\mathcal{D}_d|_{\mathcal{T}_X}$ is the distance matrix for the d-subnetwork of $\mathcal{S}$ with respect to $\mathcal{T}_X$ (Lemma 4.42), $\mathcal{S}_X \equiv \mathcal{S}_d|_{\mathcal{T}_X}$. Similarly, $\mathcal{S}_Y \equiv \mathcal{S}_d|_{\mathcal{T}_Y}$. ■

**Corollary 5.11** *If $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$ temporally decouple $(\mathcal{T}, \mathcal{C})$, then $\mathcal{S}_X$ and $\mathcal{S}_Y$ temporally decouple $(\mathcal{T}, \mathcal{C} \cup \mathcal{C}_X \cup \mathcal{C}_Y)$ such that Properties $1_X$ and $1_Y$ (and hence Properties $2_{XY}$ and $2_{YX}$) from Theorem 5.10 hold with equality in all instances.*

**Proof** Let $\mathcal{D}^+$ be the distance matrix for the STN, $\mathcal{S}^+ = (\mathcal{T}, \mathcal{C} \cup \mathcal{C}_X \cup \mathcal{C}_Y)$. By Lemma 5.5, if $\mathcal{S}_X$ and $\mathcal{S}_Y$ temporally decouple $(\mathcal{T}, \mathcal{C})$, then they also temporally decouple $\mathcal{S}^+$. Thus, it suffices to show that Properties $1_X$ and $1_Y$ hold with equality in all instances for this temporal decoupling.

Let $x_i, x_j \in \mathcal{T}_X$ be arbitrary. By Theorem 4.33, there is a solution for $\mathcal{S}^+$ in which $x_j - x_i = \mathcal{D}^+(x_i, x_j)$. However, any solution to $\mathcal{S}^+$ must satisfy the constraints in $\mathcal{C}_X$. Thus, $x_j - x_i \leq \mathcal{D}_X$. Thus,

$$\mathcal{D}^+(x_i, x_j) \;=\; x_j - x_i \;\leq\; \mathcal{D}_X(x_i, x_j).$$

Since the opposite inequality is guaranteed by Property $1_X$ of a temporal decoupling of $\mathcal{S}^+$, we have that $\mathcal{D}^+(x_i, x_j) = \mathcal{D}_X(x_i, x_j)$. Since $x_i$ and $x_j$ were arbitrary in $\mathcal{T}_X$, Property $1_X$ holds with equality in all instances for this decoupling. Similarly remarks apply to Property $1_Y$. ■

**Theorem 5.12 (Sufficient Conditions)** *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$, $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$ be consistent STNs such that $\mathcal{T}_X$ and $\mathcal{T}_Y$ z-partition $\mathcal{T}$. If Properties $1_X$, $1_Y$, $2_{XY}$ and $2_{YX}$ of Theorem 5.10 hold, then $\mathcal{S}_X$ and $\mathcal{S}_Y$ are a temporal decoupling of $\mathcal{S}$.*

**Proof** Suppose $\mathcal{S}$, $\mathcal{S}_X$ and $\mathcal{S}_Y$ satisfy the above conditions. The only part of the definition of a temporal decoupling (Definition 5.2) that is non-trivial to verify in this setting is the Mergeable Solutions Property. Let

$$\begin{array}{rcl} \mathcal{X} & = & \{\, z = 0,\ x_1 = v_1,\ \ldots,\ x_m = v_m \,\} \text{ and} \\ \mathcal{Y} & = & \{\, z = 0,\ y_1 = w_1,\ \ldots,\ y_n = w_n \,\} \end{array}$$

be arbitrary solutions for $\mathcal{S}_X$ and $\mathcal{S}_Y$, respectively. To show that $\mathcal{X} \cup \mathcal{Y}$ is a solution for $\mathcal{S}$, it suffices to show that the variable assignments in $\mathcal{X} \cup \mathcal{Y}$ satisfy all of the constraints in $\mathcal{C}$. Let $E$: $\tau_j - \tau_i \leq \delta$ be an arbitrary constraint in $\mathcal{C}$, where $\tau_i$ and $\tau_j$ are meta-variables ranging over the time-point variables in $\mathcal{T} = \mathcal{T}_X \cup \mathcal{T}_Y$.

Case 1: $\tau_i, \tau_j \in \mathcal{T}_X$. Thus, $\tau_i = x_p$ and $\tau_j = x_q$ for some $x_p$ and $x_q$ in $\mathcal{T}_X$. (These equalities do not represent temporal constraints; they simply state which time-point variables the meta-variables $\tau_i$ and $\tau_j$ represent.) $\mathcal{X}$ being a solution for $\mathcal{S}_X$ implies that $v_q - v_p \leq \mathcal{D}_X(x_p, x_q)$. Since Property $1_X$ holds, $\mathcal{D}_X(x_p, x_q) \leq \mathcal{D}(x_p, x_q)$. Finally, since $E$ is a constraint in $\mathcal{C}$, we have that $\mathcal{D}(x_p, x_q) \leq \delta$. Thus,

$$v_q - v_p \;\leq\; \mathcal{D}_X(x_p, x_q) \;\leq\; \mathcal{D}(x_p, x_q) \;\leq\; \delta$$

(i.e., the constraint $E$ is satisfied by $x_p = v_p$ and $x_q = v_q$).

Case 2: $\tau_i, \tau_j \in \mathcal{T}_Y$. Handled analogously to Case 1.

Case 3: $\tau_i \in \mathcal{T}_X$ and $\tau_j \in \mathcal{T}_Y$. Thus, $\tau_i = x_p$ and $\tau_j = y_q$ for some $x_p \in \mathcal{T}_X$ and $y_q \in \mathcal{T}_Y$. Since $x_p = v_p$ is part of a solution for $\mathcal{S}_X$, we have that $0 - v_p \leq \mathcal{D}_X(x_p, z)$. Similarly, $w_q - z \leq \mathcal{D}_Y(z, y_q)$. Thus,

$$w_q - v_p \;=\; (z - v_p) + (w_q - z) \;\leq\; \mathcal{D}_X(x_p, z) + \mathcal{D}_Y(z, y_q).$$

Since Property $2_{XY}$ holds, $\mathcal{D}_X(x_p, z) + \mathcal{D}_Y(z, y_q) \leq \mathcal{D}(x_p, y_q)$. Finally, since $E$ is a constraint in $\mathcal{C}$, we have that $\mathcal{D}(x_p, y_q) \leq \delta$. Thus,

$$w_q - v_p \ \leq\ \mathcal{D}_X(x_p, z) + \mathcal{D}_Y(z, y_q) \ \leq\ \mathcal{D}(x_p, y_q) \ \leq \delta$$

(i.e., the constraint $E$ is satisfied by $x_p = v_p$ and $y_q = w_q$).

Case 4: $\tau_i \in \mathcal{T}_Y$ and $\tau_j \in \mathcal{T}_X$. Handled analogously to Case 3.

Since the constraint $E$ was arbitrary in $\mathcal{C}$, $\mathcal{X} \cup \mathcal{Y}$ is a solution for $\mathcal{S}$. ■

### 5.2.3 Toward a TDP Algorithm

Properties $2_{XY}$ and $2_{YX}$ of Theorem 5.10 specify numerous inequalities that must be satisfied by any temporal decoupling. The analysis in this section, culminating in Lemma 5.18, shows that it suffices to satisfy only the inequalities pertaining to what are called *tight, proper xy-edges*. Based on this analysis, the TDP algorithm in Section 5.2.4 operates exclusively on tight, proper xy-edges.

**Definition 5.13 (xy-Pairs, xy-Edges)** *Let $\mathcal{T}_X$, $\mathcal{T}_Y$ and $\mathcal{T}$ be sets of time-points such that $\mathcal{T}_X$ and $\mathcal{T}_Y$ z-partition $\mathcal{T}$. Let $t_i$ and $t_j$ be arbitrary time-points in $\mathcal{T}$. The pair $(t_i, t_j)$ is called an* xy-pair *if:*

$$(t_i \in \mathcal{T}_X \textit{ and } t_j \in \mathcal{T}_Y) \quad \textit{or} \quad (t_i \in \mathcal{T}_Y \textit{ and } t_j \in \mathcal{T}_X).$$

*If, in addition, neither $t_i$ nor $t_j$ is the zero time-point variable, then $(t_i, t_j)$ is called a* proper *xy-pair. A constraint (or edge), $(t_j - t_i \leq \delta)$, is called an* xy-edge *if $(t_i, t_j)$ is an xy-pair. An xy-edge is called a* proper *xy-edge if the corresponding pair is a proper xy-pair.*

Given some proper xy-edge, the *zero-path shortfall* associated with that edge specifies a lower bound on the amount by which the corresponding inequality in Property $2_{XY}$ or $2_{YX}$ in Theorem 5.10 *fails* to hold. For a *tight*, proper xy-edge, the zero-path shortfall specifies the *exact* amount by which the corresponding inequality fails to hold. The TDP algorithm in the next section operates on tight, proper xy-edges until the zero-path shortfall associated with each such edge is equal to zero (and hence the corresponding inequality from Property $2_{XY}$ or $2_{YX}$ *holds*).

**Definition 5.14 (Zero-Path Shortfall)** *The* zero-path shortfall *(ZPS) associated with the proper xy-edge, $E$: $(t_j - t_i \leq \delta)$, is defined by:*

$$\mathrm{ZPS}(E) \ = \ [\mathcal{D}(t_i, z) + \mathcal{D}(z, t_j)] - \delta.$$

**Lemma 5.15** *If $E$ is a tight, proper xy-edge, then $\mathrm{ZPS}(E) \geq 0$.*

**Proof** Let $E$: $t_j - t_i \leq \delta$ be a tight, proper xy-edge. Since the edge is tight, $\mathcal{D}(t_i, t_j) = \delta$. Hence, from the Triangle Inequality:

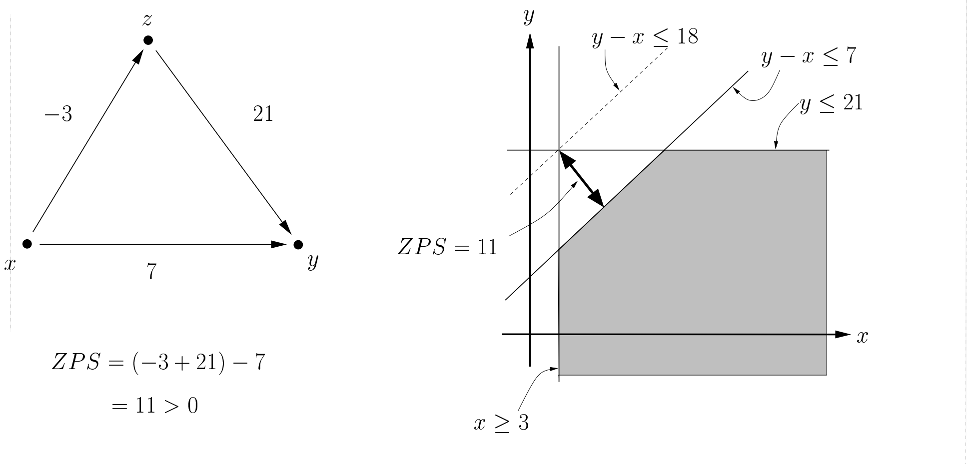



Figure 5.5: An xy-edge that is *not* dominated by a path through zero

$$\delta \ = \ \mathcal{D}(t_i, t_j) \ \leq \ \mathcal{D}(t_i, z) + \mathcal{D}(z, t_j).$$

Thus, ZPS($E$) $\geq 0$. ■

**Definition 5.16 (Dominated by a Path through Zero)** *If the ZPS value for a tight, proper xy-edge is zero, we say that that edge is* dominated by a path through zero.[1]

Figure 5.5 shows a tight, proper xy-edge, $y - x \leq 7$, that is *not* dominated by a path through zero (i.e., it has a *positive* ZPS value) along with an xy-plane representation of the involved constraints (in which the set of solutions is shaded). The ZPS value for a given edge $y - x \leq \delta$ may be reduced by strengthening one or both of the xz- and zy-edges. Figure 5.6 shows the xy-edge from Figure 5.5, but with the xz- and zy-edges strengthened such that the xy-edge is now dominated by a path through zero. Notice that in the xy-plane representation for this situation, the dominated xy-constraint no longer plays a role in shaping the space of solutions.

**Lemma 5.17** *If adding a set of constraints to a consistent STN $\mathcal{S}$ does not make $\mathcal{S}$ inconsistent, then the ZPS values for any pre-existing tight, proper xy-edges in $\mathcal{S}$ cannot increase.*

[1]As will be seen below, if each tight, proper xy-edge is dominated by a path through zero, then $\mathcal{S}_X$ and $\mathcal{S}_Y$ will necessarily form a temporal decoupling of $\mathcal{S}$. If we ignore these dominated (and hence redundant) constraints in $\mathcal{S}$, we have that $z$ is a *separation vertex* of $\mathcal{S}$, according to the definition given by Dechter et al. (1991). Whereas Dechter et al. focus on finding non-separable components of a given STN, our focus is on finding a set of constraints to impose on an STN to achieve a temporal decoupling of that STN into two (or more) subnetworks. See Section 5.6.1.

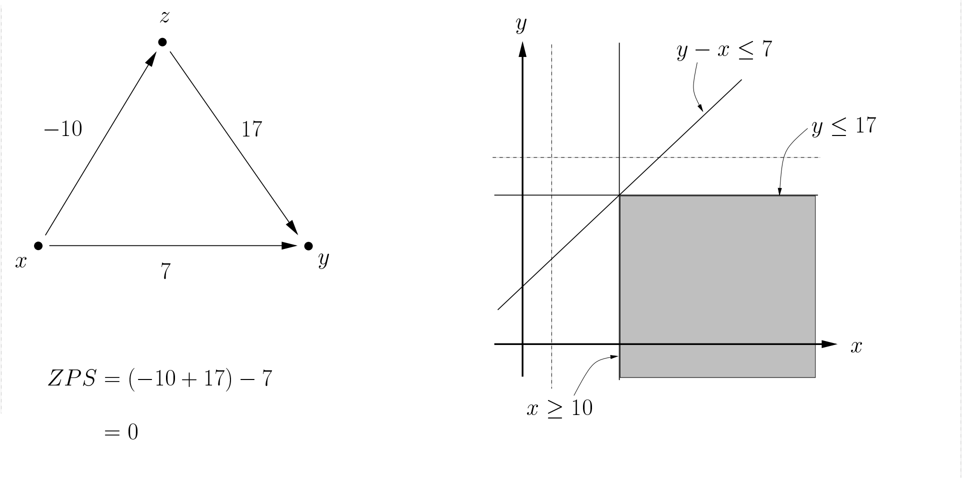



Figure 5.6: An xy-edge that *is* dominated by a path through zero

**Proof** Adding constraints to an STN causes its shortest paths to become shorter or stay the same, and hence its distance matrix entries to decrease or stay the same. Given that $\delta$ is a constant, this implies that $[\mathcal{D}(t_i, z) + \mathcal{D}(z, t_j)] - \delta$ can only decrease or stay the same. ■

Lemma 5.18 shows that it suffices to consider only tight, proper xy-edges when seeking a solution to an instance of the TDP.

**Lemma 5.18** *If* $\mathcal{D}(t_p, z) + \mathcal{D}(z, t_q) \leq \delta$ *holds for every* tight, proper *xy*-edge $(t_q - t_p \leq \delta)$ *in a consistent STN* $\mathcal{S}$*, then* $\mathcal{D}(t_i, z) + \mathcal{D}(z, t_j) \leq \mathcal{D}(t_i, t_j)$ *holds for every xy*-pair $(t_i, t_j)$ *in* $\mathcal{S}$.

**Proof** Suppose that the inequality $\mathcal{D}(t_p, z) + \mathcal{D}(z, t_q) \leq \delta$ holds for every tight, proper xy-*edge* $(t_q - t_p \leq \delta)$ in $\mathcal{S}$. Let $(t_i, t_j)$ be an arbitrary xy-*pair* in $\mathcal{S}$. It suffices to show that the inequality $\mathcal{D}(t_i, z) + \mathcal{D}(z, t_j) \leq \mathcal{D}(t_i, t_j)$ holds.

If $t_i$ or $t_j$ is the zero time-point $z$, then the inequality holds trivially (since $\mathcal{D}(z, z) = 0$ in a consistent STN). Thus, suppose that $(t_i, t_j)$ is a *proper* xy-pair. Without loss of generality, suppose $t_i \in \mathcal{T}_X$ and $t_j \in \mathcal{T}_Y$. Let $P$ be an arbitrary *shortest* path from $t_i$ to $t_j$ in $\mathcal{S}$. If $z$ is on the path $P$, then the inequality holds (since the subpaths from $t_i$ to $z$ and from $z$ to $t_j$ must also be shortest paths, by Proposition 4.20).

Now suppose $z$ is not on $P$. Then $P$ must contain at least one proper xy-edge $E_{xy}$ of the form $(y - x \leq \delta_{xy})$, where $x \in \mathcal{T}_X$ and $y \in \mathcal{T}_Y$. Since $E_{xy}$ is on a shortest path, it must be tight (cf. Proposition 4.20); hence $\delta_{xy} = \mathcal{D}(x, y)$. Thus, the Lemma's premise, applied to the tight, proper xy-edge $E_{xy}$, implies that: $\mathcal{D}(x, z) + \mathcal{D}(z, y) \leq \delta_{xy}$. However, the Triangle Inequality provides the opposite inequality: $\delta_{xy} = \mathcal{D}(x, y) \leq \mathcal{D}(x, z) + \mathcal{D}(z, y)$. Thus, $\delta_{xy} = \mathcal{D}(x, z) + \mathcal{D}(z, y)$. Thus, we may replace $E_{xy}$ in $P$ by a pair of shortest paths, one from $x$ to $z$, one from $z$ to $y$, without changing the length of $P$. But then this version

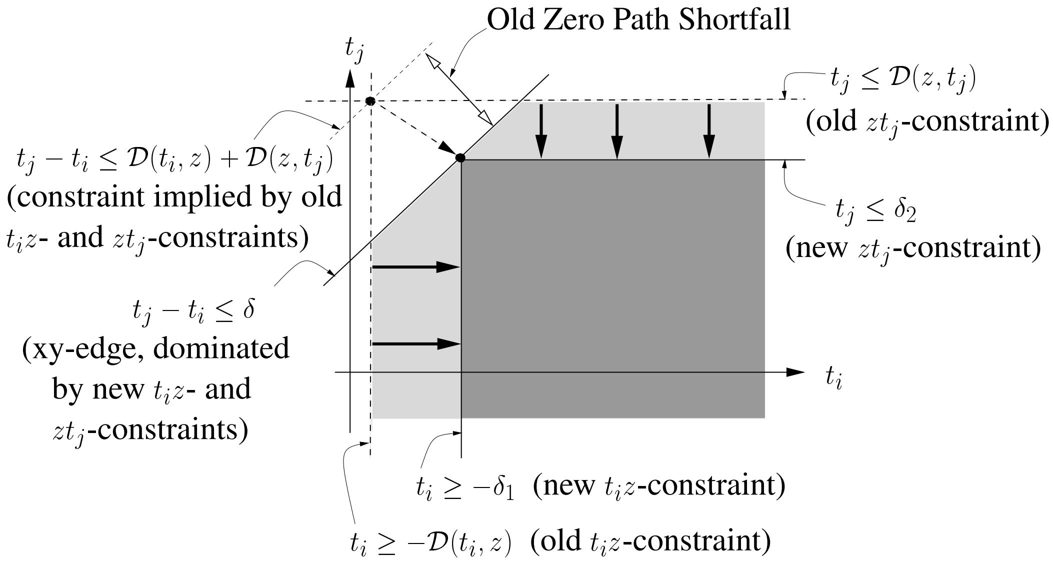



Figure 5.7: Reducing the zero-path shortfall for an xy-edge

of $P$ is a shortest path from $t_i$ to $t_j$ that contains $z$. As argued earlier, this implies that the desired inequality holds. ∎

### 5.2.4 Algorithms for Solving the TDP

This section presents a family of sound and complete algorithms for solving the Temporal Decoupling Problem. The basic TDP algorithm, which is directly motivated by Theorem 5.12 and Lemma 5.18, is sound, but not complete: because it is not guaranteed to terminate. However, Theorems 5.26 and 5.28 (below) specify a variety of simple strategies for strengthening the basic algorithm to ensure that it terminates and, hence, that it is complete.

#### The Basic TDP Algorithm

The main tool of the basic TDP algorithm is to reduce the zero-path shortfall for each tight, proper xy-edge, $t_j - t_i \leq \delta$, by strengthening the corresponding $t_iz$- and/or $zt_j$-edges. Figure 5.7 illustrates the case of an xy-edge's ZPS value being reduced to zero through the addition of edges $z - t_i \leq \delta_1$ (i.e., $t_i \geq -\delta_1$) and $t_j - z \leq \delta_2$ (i.e., $t_j \leq \delta_2$). Adding weaker constraints may reduce the zero-path shortfall but not eliminate it entirely.

The basic TDP algorithm is given in pseudo-code in Figure 5.8. It takes as input an STN $\mathcal{S}$ whose time-points are z-partitioned by the sets $\mathcal{T}_X$ and $\mathcal{T}_Y$. During the course of the algorithm, constraints are added to $\mathcal{S}$ and the distance matrix $\mathcal{D}$ is updated accordingly.

At Step 1, the algorithm checks whether $\mathcal{S}$ is consistent. If $\mathcal{S}$ is inconsistent, the algorithm returns `NIL` and halts because, by Lemma 5.3, only consistent STNs can be decoupled. Otherwise, the algorithm initializes $\mathcal{E}$ to the set of tight, proper xy-edges in $\mathcal{S}$,

**Given:** An STN $\mathcal{S}$ whose time-points $\mathcal{T}$ are z-partitioned by the sets $\mathcal{T}_X$ and $\mathcal{T}_Y$.

(1) Compute the distance matrix $\mathcal{D}$ for $\mathcal{S}$. If $\mathcal{S}$ is inconsistent, return `NIL` and halt; otherwise, initialize $\mathcal{E}$ to the set of tight, proper xy-edges in $\mathcal{S}$, and continue.

(2) Select a tight, proper xy-edge $E = (t_j - t_i \leq \delta)$ from $\mathcal{E}$ whose ZPS value is positive. (If, in the process, any edges in $\mathcal{E}$ are discovered that are no longer tight or that have a ZPS value of zero, remove those edges from $\mathcal{E}$.) If no such edges exist (i.e., if $\mathcal{E}$ has become empty), go to Step 6; otherwise, continue.

(3) Pick values $\delta_1$ and $\delta_2$ such that:

$$\begin{array}{rcccl} -\mathcal{D}(z,t_i) & \leq & \delta_1 & \leq & \mathcal{D}(t_i,z), \\ -\mathcal{D}(t_j,z) & \leq & \delta_2 & \leq & \mathcal{D}(z,t_j), \quad \text{and} \\ \delta & \leq & \delta_1+\delta_2 & < & \mathcal{D}(t_i,z)+\mathcal{D}(z,t_j). \end{array}$$

(4) Add the constraints, $E_1$: $z - t_i \leq \delta_1$ and $E_2$: $t_j - z \leq \delta_2$, to $\mathcal{S}$, updating $\mathcal{D}$ accordingly.

(5) Go to Step 2.

(6) Return:

$$\begin{array}{rcccl} \mathcal{C}_X & = & \mathcal{C}_d|_{\mathcal{T}_X} & = & \{(t_j - t_i \leq \mathcal{D}(t_i,t_j)) : t_i, t_j \in \mathcal{T}_X\}; \\ \mathcal{C}_Y & = & \mathcal{C}_d|_{\mathcal{T}_Y} & = & \{(t_j - t_i \leq \mathcal{D}(t_i,t_j)) : t_i, t_j \in \mathcal{T}_Y\}, \end{array}$$

where $\mathcal{C}$ includes all constraints added in passes of Step 4, and $\mathcal{D}$ has been updated accordingly.

Figure 5.8: Pseudo-code for the basic TDP algorithm

which merely requires checking each edge against the corresponding entries in the distance matrix. The algorithm then iteratively operates on edges drawn from $\mathcal{E}$ until each such edge is dominated by a path through zero (recall Definition 5.16).

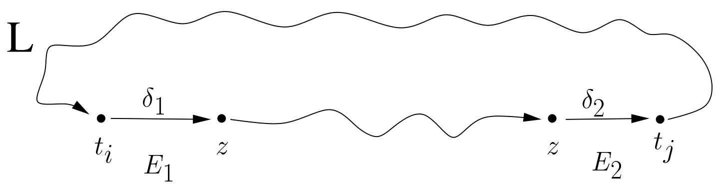
For each iteration, the algorithm does the following. In Step 2, a tight, proper xy-edge $E$: $(t_j - t_i \leq \delta)$ with a positive zero-path shortfall, $\zeta$, is selected from the set $\mathcal{E}$. In Steps 3 and 4, new constraints $E_1$: $z - t_i \leq \delta_1$ and $E_2$: $t_j - z \leq \delta_2$ are added to $\mathcal{S}$. (Theorem 5.19 (below) ensures that adding $E_1$ and $E_2$ to $\mathcal{S}$ will not cause $\mathcal{S}$ to become inconsistent.) After updating the distance matrix $\mathcal{D}$ to reflect the new constraints, the new values of $\mathcal{D}(t_i, z)$ and $\mathcal{D}(z, t_j)$ will be $\delta_1$ and $\delta_2$, respectively (as shown in Corollary 5.20 below). Thus, the *updated* ZPS value for $E$, which we denote $\zeta^*$, will be: $\zeta^* = \delta_1 + \delta_2 - \delta$. The third Step 3 constraint guarantees that $\zeta^*$ will be strictly smaller than $\zeta$ (i.e., that the ZPS value of $E$ has been reduced).

Upon adding the Step 4 edges to $\mathcal{S}$, it may be that some of the edges in $\mathcal{E}$ are no longer tight or no longer have positive ZPS values. However, the algorithm need not check for that in Step 4. Instead, if any such edges are ever encountered during the selection process in Step 2, they are simply removed from $\mathcal{E}$ at that time.

If it ever happens that every tight, proper xy-edge is dominated by a path through zero, as evidenced by the set $\mathcal{E}$ becoming empty, then the algorithm terminates (see Steps 2 and 6). Theorem 5.23 guarantees that the algorithm is sound: if the algorithm terminates at Step 6, then the constraint sets $\mathcal{C}_X$ and $\mathcal{C}_Y$ will be such that $(\mathcal{T}_X, \mathcal{C}_X)$ and $(\mathcal{T}_Y, \mathcal{C}_Y)$ are a temporal decoupling of $\mathcal{S}$.

**Theorem 5.19** *Let $\mathcal{S}$ be a consistent STN; let $E$: $t_j - t_i \leq \delta$ be a tight, proper xy-edge whose ZPS value is positive; and let $\delta_1$ and $\delta_2$ be arbitrary values satisfying the requirements of Step 3 of the basic TDP algorithm. Then adding the* pair *of corresponding Step 4 constraints (i.e., $E_1$ and $E_2$ in Figure 5.8) will not threaten the consistency of $\mathcal{S}$.*

**Proof** Let $\zeta > 0$ be the ZPS value for the edge $E$. Suppose that adding the corresponding Step 4 constraints $E_1$ and $E_2$ caused $\mathcal{S}$ to become inconsistent. Then, by Theorem 4.30, there must be a loop with negative path-length in the distance graph $\mathcal{G}$. By Theorem 4.33, the first two Step 3 requirements (from Figure 5.8) imply that $E_1$ and $E_2$ are individually consistent with $\mathcal{S}$. Thus, any loop with negative path length in $\mathcal{G}$ must contain *both* $E_1$ and $E_2$. Of all such loops, let $L$ be one that has the minimum number of edges. $L$ is illustrated below.




Let $L'$ be the subpath of $L$ from $t_i$ (at the beginning of $E_1$) to $t_j$ (at the end of $E_2$). Then,

$$|L'| \;\geq\; \delta_1 + \mathcal{D}(z, z) + \delta_2 \;\geq\; \delta_1 + 0 + \delta_2 \;=\; \delta_1 + \delta_2.$$

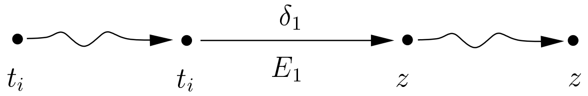
However, part of the third Step 3 requirement (from Figure 5.8) says that $\delta_1 + \delta_2 \geq \delta$. Thus, $|L'| \geq \delta$. As a result, the subpath $L'$, which contains at least two edges, could be replaced in $L$ by the single edge $E$ (which has length $\delta$), resulting in a loop $L''$ such that $|L''| \leq |L| < 0$. But $L''$ has fewer edges than $L$, which contradicts the choice of $L$. Thus, no such $L$ exists. Thus, adding both $E_1$ and $E_2$ to $\mathcal{S}$ leaves $\mathcal{S}$ consistent. ■

**Corollary 5.20** *Let $\mathcal{S}$ be a consistent STN. Let $E_1$: $z - t_i \leq \delta_1$ and $E_2$: $t_j - z \leq \delta_2$ be constraints satisfying the Step 3 requirements of the basic TDP algorithm. Let $\mathcal{S}^+$ be the STN resulting from adding $E_1$ and $E_2$ to $\mathcal{S}$. Let $\mathcal{D}^+$ be the corresponding distance matrix. Then,*

$$\mathcal{D}^+(t_i, z) = \delta_1 \quad \text{and} \quad \mathcal{D}^+(z, t_j) = \delta_2.$$

**Proof** Let $P$ be a shortest path from $t_i$ to $z$ in $\mathcal{S}^+$. Thus, $|P| = \mathcal{D}^+(t_i, z)$. Since $E_1$ by itself constitutes a path from $t_i$ to $z$ in $\mathcal{S}^+$ with length $\delta_1$, we have that $|P| = \mathcal{D}^+(t_i, z) \leq \delta_1$.

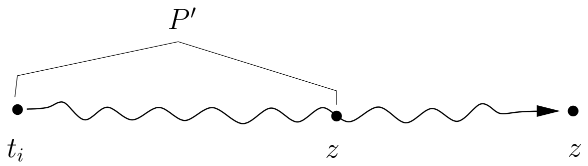
Case 1: $E_1$ is contained in $P$, as shown below.




Since any subpaths of a shortest path must themselves be shortest paths (and since $\mathcal{S}^+$ is guaranteed to be consistent by Theorem 5.19),

$$|P| \;=\; \mathcal{D}(t_i, t_i) + \delta_1 + \mathcal{D}(z, z) \;=\; 0 + \delta_1 + 0 \;=\; \delta_1.$$

Case 2: $E_1$ is not contained in $P$. Let $P'$ be the subpath of $P$ from the beginning of $P$ to the first occurrence of $z$ in $P$, as shown below.




(Since $z$ appears at the end of $P$, the subpath $P'$ is well-defined. Furthermore, it may be that $P' = P$.) Since $P'$ is a subpath of a shortest path, $P'$ must itself be a shortest path. However, $P'$ is also a path from $t_i$ to $z$. Thus, $|P'| = |P|$. Furthermore, $P'$ does not contain either $E_1$ (by assumption) or $E_2$ (since the only occurrence of $z$ in $P'$ is at the end of $P'$). Thus, $|P'| = \mathcal{D}(t_i, z)$ (where $\mathcal{D}$ is the distance matrix for $\mathcal{S}$, not $\mathcal{S}^+$). But the first Step 3 requirement says that $\mathcal{D}(t_i, z) \geq \delta_1$. Thus, $|P| = |P'| = \mathcal{D}(t_i, z) \geq \delta_1$. Since it has already been established that $|P| \leq \delta_1$, it must be that $|P| = \delta_1$.

In either case, it has been shown that a shortest path from $t_i$ to $z$ in $\mathcal{S}^+$ has length $\delta_1$; thus $\mathcal{D}^+(t_i, z) = \delta_1$. That $\mathcal{D}^+(z, t_j) = \delta_2$ follows in a similar fashion. ■

The values $\delta_1$ and $\delta_2$ chosen in Step 3 of the algorithm specify the strengths of the constraints $E_1$ and $E_2$ added in Step 4. It is also useful to think in terms of the amount $R$ by which the ZPS value for the edge under consideration is thereby reduced, as well as the fractions of this ZPS reduction corresponding to the amounts by which the $t_iz$- and $zt_j$-edges were tightened (specified by $\alpha$ and $1-\alpha$). Lemma 5.21 (below) specifies the relationship between the pairs of values $(\delta_1, \delta_2)$ and $(R, \alpha)$. Lemma 5.21 is subsequently used to demonstrate various properties of the TDP algorithm (e.g., its soundness).

**Lemma 5.21** *Let $E$ be some tight, proper xy-edge $t_j - t_i \leq \delta$ whose ZPS value $\zeta = \mathrm{ZPS}(E)$ is positive. Let $\Omega$ be the set of ordered pairs $(\delta_1, \delta_2)$ satisfying the Step 3 requirements of the basic TDP algorithm (recall Figure 5.8). Let $\Theta$ be the set of ordered pairs $(R, \alpha)$ such that $R \in (0, \zeta]$ and $\alpha \in [0, 1]$. Let $T_1$ and $T_2$ be the following 2-by-2 transformations:*

$$T_1\text{:}\ \begin{cases} \delta_1 = f(R,\alpha) = \mathcal{D}(t_i, z) - \alpha R \\ \delta_2 = g(R,\alpha) = \mathcal{D}(z, t_j) - (1-\alpha)R \end{cases}$$

$$T_2\text{:}\ \begin{cases} R = u(\delta_1,\delta_2) = \mathcal{D}(t_i, z) + \mathcal{D}(z, t_j) - (\delta_1 + \delta_2) \\ \alpha = v(\delta_1,\delta_2) = \frac{\mathcal{D}(t_i,z)-\delta_1}{\mathcal{D}(t_i,z)+\mathcal{D}(z,t_j)-(\delta_1+\delta_2)} \end{cases}$$

*Then $T_1$ and $T_2$ are invertible transformations between $\Omega$ and $\Theta$ (with $T_1^{-1} = T_2$) such that for any pair $(\delta_1, \delta_2) \in \Omega$, the corresponding pair $(R, \alpha) \in \Theta$ satisfies that:*

- *$R$ is the amount by which the pair of corresponding Step 4 constraints, $E_1$: $z - t_i \leq \delta_1$ and $E_2$: $t_j - z \leq \delta_2$, reduce the ZPS value for the edge $E$, and*
- *$\alpha$ and $(1-\alpha)$ represent the fractions of this ZPS reduction due to the tightening of the $t_iz$- and $zt_j$-edges, respectively.*

**Proof** The Step 3 requirements (from Figure 5.8) correspond to the boundaries of the region $\Omega$ in Figure 5.9. Note that the point $(\mathcal{D}(t_i, z), \mathcal{D}(z, t_j))$ is *not* part of $\Omega$ due to the strict inequality: $\delta_1 + \delta_2 < \mathcal{D}(t_i, z) + \mathcal{D}(z, t_j)$. Also, by Lemma 4.21,

$$\mathcal{D}(t_i, z) - \mathcal{D}(t_j, z) \leq \delta \quad \text{and} \quad \mathcal{D}(z, t_j) - \mathcal{D}(z, t_i) \leq \delta.$$

These inequalities ensure that the diagonal boundary of $\Omega$, which corresponds to the constraint $\delta \leq \delta_1 + \delta_2$, lies above and to the right of the lines $\mathcal{D}(t_i, z) - \mathcal{D}(t_j, z) = \delta_1 + \delta_2$ and $\mathcal{D}(z, t_j) - \mathcal{D}(z, t_i) = \delta_1 + \delta_2$ (shown as dashed lines in the $\delta_1\delta_2$-plane in Figure 5.9).

The amount of reduction in the ZPS value resulting from adding the Step 4 constraints, $E_1$: $z - t_i \leq \delta_1$ and $E_2$: $t_j - z \leq \delta_2$, is given by:

$$\begin{aligned} &(\text{old ZPS value}) - (\text{new ZPS value}) \\ &\quad = \zeta - \zeta^* \\ &\quad = [\mathcal{D}(t_i, z) + \mathcal{D}(z, t_j) - \delta] - [\delta_1 + \delta_2 - \delta] \\ &\quad = \mathcal{D}(t_i, z) + \mathcal{D}(z, t_j) - (\delta_1 + \delta_2) \end{aligned}$$

which is precisely the value $R = u(\delta_1, \delta_2)$. The amount by which the $t_iz$-edge is strengthened is given by:

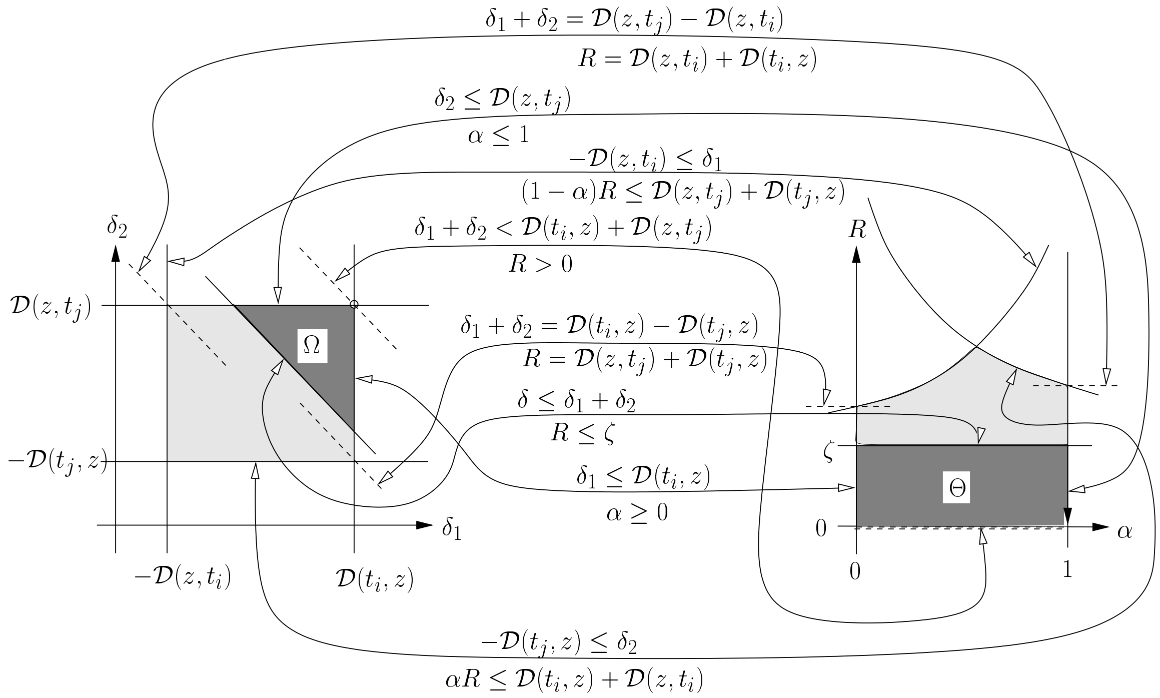



Figure 5.9: The regions $\Omega$ and $\Theta$ from Lemma 5.21

$$\text{(old value)} - \text{(new value)} \;=\; \mathcal{D}(t_i, z) - \delta_1.$$

Hence, the fraction of the total ZPS reduction produced by strengthening the $t_i z$-edge is precisely $\alpha = v(\delta_1, \delta_2)$, from which it also follows that $\delta_1 = f(R, \alpha) = \mathcal{D}(t_i, z) - \alpha R$. Similarly, the fraction of the total ZPS reduction produced by strengthening the $z t_j$-edge is precisely $(1 - \alpha)$, and $\delta_2 = g(R, \alpha) = \mathcal{D}(z, t_j) - (1 - \alpha)R$.

That the transformations $T_1$ and $T_2$ are inverses of one another is easily verified by showing that the following equations hold for any $(\delta_1, \delta_2) \in \Omega$ and any $(R, \alpha) \in \Theta$:

$$\left\{ \begin{array}{l} f(u(\delta_1, \delta_2), v(\delta_1, \delta_2)) = \delta_1 \\ g(u(\delta_1, \delta_2), v(\delta_1, \delta_2)) = \delta_2 \end{array} \right\} \text{ and } \left\{ \begin{array}{l} u(f(R, \alpha), g(R, \alpha)) = R \\ v(f(R, \alpha), g(R, \alpha)) = \alpha \end{array} \right\}.$$

Also, given any $(\delta_1, \delta_2) \in \Omega$, we may verify that $(R, \alpha) = (u(\delta_1, \delta_2), v(\delta_1, \delta_2)) \in \Theta$ by showing that $u(\delta_1, \delta_2) \in (0, \zeta]$ and $v(\delta_1, \delta_2) \in [0, 1]$, as follows:

- $\mathcal{D}(t_i, z) + \mathcal{D}(z, t_j) \;>\; \delta_1 + \delta_2$ (Step 3 Requirement)
  $\mathcal{D}(t_i, z) + \mathcal{D}(z, t_j) - (\delta_1 + \delta_2) \;>\; 0$ (Rearrange terms)
  $u(\delta_1, \delta_2) \;>\; 0$ (Defn. of $u(\delta_1, \delta_2)$)

- $\delta \;\leq\; \delta_1 + \delta_2$ (Step 3 Requirement)
  $-(\delta_1 + \delta_2) \;\leq\; -\delta$ (Rearrange terms)
  $\mathcal{D}(t_i, z) + \mathcal{D}(z, t_j) - (\delta_1 + \delta_2)$
  $\quad\leq\; \mathcal{D}(t_i, z) + \mathcal{D}(z, t_j) - \delta$ (Add $\mathcal{D}(t_i, z) + \mathcal{D}(z, t_j)$ to both sides)
  $u(\delta_1, \delta_2) \;\leq\; \zeta$ (Defns. of $u(\delta_1, \delta_2)$ and $\zeta$)

- $\mathcal{D}(t_i, z) \;\geq\; \delta_1$ (Step 3 Requirement)
  $\mathcal{D}(t_i, z) - \delta_1 \;\geq\; 0$ (Rearrange terms)
  $\frac{\mathcal{D}(t_i,z)-\delta_1}{\mathcal{D}(t_i,z)+\mathcal{D}(z,t_j)-(\delta_1+\delta_2)} \;\geq\; 0$ (Divide by positive number)
  $v(\delta_1, \delta_2) \;\geq\; 0$ (Defn. of $v(\delta_1, \delta_2)$)

- $\delta_2 \;\leq\; \mathcal{D}(z, t_j)$ (Step 3 Requirement)
  $\mathcal{D}(t_i, z) - \delta_1$
  $\quad \leq\; \mathcal{D}(t_i, z) + \mathcal{D}(z, t_j) - (\delta_1 + \delta_2)$ (Add $\mathcal{D}(t_i, z) - \delta_1 - \delta_2$ to both sides)
  $\frac{\mathcal{D}(t_i,z)-\delta_1}{\mathcal{D}(t_i,z)+\mathcal{D}(z,t_j)-(\delta_1+\delta_2)} \;\leq\; 1$ (Divide by RHS, which is positive)
  $v(\delta_1, \delta_2) \;\leq\; 1$ (Defn. of $v(\delta_1, \delta_2)$)

Furthermore, since all these steps are reversible, we get that

$$(\delta_1, \delta_2) = (f(R, \alpha), g(R, \alpha)) \in \Omega, \text{ for any } (R, \alpha) \in \Theta.$$

Finally, from Theorem 4.33, the quantity $\mathcal{D}(z, t_j) + \mathcal{D}(t_j, z)$ specifies the maximum amount that the $zt_j$-edge can be tightened without threatening the consistency of the STN. The following establishes that $\zeta$ (i.e., the ZPS value for the edge $E$) is no more than this amount:

$\mathcal{D}(t_i, z) - \mathcal{D}(t_j, z) \;\leq\; \delta$ (Lemma 4.21)
$\mathcal{D}(t_i, z) + \mathcal{D}(z, t_j) - \delta \;\leq\; \mathcal{D}(z, t_j) + \mathcal{D}(t_j, z)$ (Add like quantities and rearrange)
$\zeta \;\leq\; \mathcal{D}(z, t_j) + \mathcal{D}(t_j, z)$ (Defn. of $\zeta$).

Thus, the entire zero-path-shortfall for $E$ may be eliminated by tightening only the $zt_j$-constraint. Similarly, the entire zero-path-shortfall may be eliminated by instead tightening only the $t_iz$-constraint. In Figure 5.9, these constraints on $\zeta$ ensure that the top horizontal boundary of the region $\Theta$ lies below the curves $\alpha R = \mathcal{D}(t_i, z) + \mathcal{D}(z, t_i)$ and $(1 - \alpha)R = \mathcal{D}(z, t_j) + \mathcal{D}(t_j, z)$. Thus, the values for $R$ and $\alpha$ may be independently chosen. ■

**Corollary 5.22** *Let $E$: $t_j - t_i \leq \delta$ be a tight, proper xy-edge with ZPS value $\zeta > 0$. Let $R \in (0, \zeta]$ and $\alpha \in [0, 1]$ be arbitrary. It is always possible to choose $\delta_1$ and $\delta_2$ satisfying the Step 3 requirements such that adding the Step 4 constraints ($E_1$ and $E_2$) results in the ZPS value for $E$ being reduced by $R$ and the $t_iz$- and $zt_j$-edges being* tightened *by the amounts $\alpha R$ and $(1 - \alpha)R$, respectively.*

**Theorem 5.23 (Soundness)** *If the temporal decoupling algorithm terminates at Step 6, then the constraint sets $\mathcal{C}_X$ and $\mathcal{C}_Y$ returned by the TDP algorithm are such that $(\mathcal{T}_X, \mathcal{C}_X)$ and $(\mathcal{T}_Y, \mathcal{C}_Y)$ are a temporal decoupling of the input STN $\mathcal{S}$.*

**Proof** During each pass of Step 4, the TDP algorithm modifies the input STN by adding new constraints. To distinguish the input STN $\mathcal{S}$ from the modified STN existing at the end of the algorithm's execution (i.e., at Step 6), we shall refer to the latter as $\mathcal{S}^+$. If

$\mathcal{C}^+$ is the set of all Step 4 constraints added during the execution of the algorithm, then $\mathcal{S}^+ = (\mathcal{T}, \mathcal{C} \cup \mathcal{C}^+)$. Let $\mathcal{D}^+$ be the distance matrix for $\mathcal{S}^+$. (Thus, using this notation, it is $\mathcal{D}^+$ that is used to construct the constraint sets $\mathcal{C}_X$ and $\mathcal{C}_Y$ in Step 6.) Since every constraint in $\mathcal{S}$ is present in $\mathcal{S}^+$, $\mathcal{D}^+(t_i, t_j) \leq \mathcal{D}(t_i, t_j)$ for all $t_i, t_j \in \mathcal{T}$.

To show that $\mathcal{S}_X$ and $\mathcal{S}_Y$ are a temporal decoupling of $\mathcal{S}$, it suffices (by Theorem 5.12) to show that $\mathcal{S}_X$ and $\mathcal{S}_Y$ are each consistent and that Properties $1_X$, $1_Y$, $2_{XY}$ and $2_{YX}$ from Theorem 5.10 hold. (It is given that the sets $\mathcal{T}_X$ and $\mathcal{T}_Y$ z-partition $\mathcal{T}$.)

Theorem 5.19 guarantees that $\mathcal{S}^+$ is consistent. Furthermore, since any solution for $\mathcal{S}^+$ must satisfy the constraints represented in $\mathcal{D}^+$, which cover all the constraints in $\mathcal{C}_X$ and $\mathcal{C}_Y$, $\mathcal{S}_X$ and $\mathcal{S}_Y$ must also be consistent.

By construction, $\mathcal{D}_X(x_p, x_q) = \mathcal{D}^+(x_p, x_q)$, for every $x_p$ and $x_q$ in $\mathcal{T}_X$. Thus, since $\mathcal{D}^+(x_p, x_q) \leq \mathcal{D}(x_p, x_q)$, Property $1_X$ of Theorem 5.10 holds. Similarly, Property $1_Y$ also holds.

To prove that Property $2_{XY}$ holds, first notice that the premise of Lemma 5.18 is equivalent to saying that ZPS $\leq 0$ for each tight, proper xy-*edge*, which is precisely what the exit clause of Step 2 requires. Thus, the algorithm will not terminate at Step 6 unless the premise of Lemma 5.18 holds—with respect to $\mathcal{S}^+$. Hence, from the conclusion of Lemma 5.18, we have that for any xy-*pair* in $\mathcal{S}^+$: $\mathcal{D}^+(t_i, z) + \mathcal{D}^+(z, t_j) \leq \mathcal{D}^+(t_i, t_j)$. In the case where $t_i \in \mathcal{T}_X$ and $t_j \in \mathcal{T}_Y$, we have that $\mathcal{D}^+(t_i, z) = \mathcal{D}_X(t_i, z)$ and $\mathcal{D}^+(z, t_j) = \mathcal{D}_Y(z, t_j)$. Since $\mathcal{D}^+(t_i, t_j) \leq \mathcal{D}(t_i, t_j)$, we get that $\mathcal{D}_X(t_i, z) + \mathcal{D}_Y(z, t_j) \leq \mathcal{D}(t_i, t_j)$, which is Property $2_{XY}$. Similarly, Property $2_{YX}$ holds. ∎

**Ensuring Completeness for the TDP Algorithm**

The Step 3 requirement that $\delta_1 + \delta_2 < \mathcal{D}(t_i, z) + \mathcal{D}(z, t_j)$ ensures that the ZPS value for the edge under consideration will decrease. However, it does not ensure that *substantial* progress will be made. As a result, the basic TDP algorithm, as shown in Figure 5.8, is not guaranteed to terminate. Theorems 5.26 and 5.28, below, specify two ways of strengthening the Step 3 requirements, each sufficient to ensure that the TDP algorithm will terminate and, hence, that it is complete. Each strategy involves a method for choosing $R$, the amount by which the ZPS value for the edge currently under consideration is to be reduced (recall Lemma 5.21). Each strategy leaves the distribution of additional constrainedness among the $t_i z$- and $z t_j$-edges (i.e., the choice of $\alpha$) unrestricted.

**Definition 5.24 (Greedy Strategy)** *At each pass of Step 3, the entire zero-path shortfall for the edge under consideration is eliminated (i.e., choose: $R = \zeta$).*

Proposition 5.25 will be used in the proofs of the theorems.

**Proposition 5.25** *If an xy-edge ever loses its tightness, it cannot ever regain it. Thus, since the algorithm never adds any proper xy-edges, the pool of tight, proper xy-edges relevant to Step 2 can never grow. Furthermore, by Lemma 5.17, the ZPS values cannot ever increase. Thus, any progress made by the algorithm is never lost.*

**Theorem 5.26** *Using the Greedy Strategy (cf. Definition 5.24), the TDP algorithm is guaranteed to terminate after at most $2|\mathcal{T}_X||\mathcal{T}_Y|$ iterations.*

**Proof** By Corollary 5.22, it is always possible to choose $\delta_1$ and $\delta_2$ in Step 3 such that the entire zero-path shortfall will be eliminated. Thus, the Greedy Strategy is feasible. By Proposition 5.25, the algorithm needs to do Step 3 processing of each tight, proper xy-edge at most once. There are at most $2|\mathcal{T}_X||\mathcal{T}_Y|$ such edges. ■

Barring some extravagant selection process in Step 2, the computation in each iteration of the TDP algorithm is dominated by the propagation of the temporal constraints added in Step 4. This is no worse than $O(|\mathcal{T}|^3)$, by Proposition 4.22.

Although the Greedy Strategy leads to quick results, a less greedy approach can lead to decoupled networks having greater flexibility, as will be seen in the experimental evaluation section (Section 5.2.5). Below, a Less-Greedy Strategy is defined in which the ZPS value of the edge $E$ under consideration in Step 3 is reduced by a fractional amount (unless it is already below some threshold). By doing so, the algorithm enables the processing of other edges to contribute to reducing the ZPS value for the edge $E$ in between its visits to $E$, thereby reducing the possibility of overconstraining the network.

Unlike the Greedy Strategy, the Less-Greedy Strategy requires all of the initial ZPS values to be finite—which is always the case in practice.

**Definition 5.27 (Less Greedy Strategy)** *Let $Z$ be the maximum of the initial ZPS values among all the tight, proper xy-edges in $\mathcal{S}$. Let $\epsilon > 0$ and $r \in (0, 1)$ be arbitrary constants. At each pass of Step 3 in the TDP algorithm, choose $R$ (the amount by which the ZPS value $\zeta$ for the edge currently under consideration is reduced) as follows:*

$$R = \begin{cases} r\zeta, & \text{if } \zeta > \epsilon \\ \zeta, & \text{otherwise} \end{cases}$$

**Theorem 5.28** *If the initial ZPS values among all of the tight, proper xy-edges in a consistent STN $\mathcal{S}$ are finite, then, using the Less-Greedy Strategy (cf. Definition 5.27), the TDP algorithm is guaranteed to terminate after at most $2|\mathcal{T}_X||\mathcal{T}_Y|\left[\lceil\frac{\log(Z/\epsilon)}{\log(1/(1-r))}\rceil + 1\right]$ iterations.*

**Proof** Corollary 5.22 ensures that the value of $R$ may always be chosen in accordance with the Less-Greedy Strategy.

Let $\mathcal{E}_0$ be the initial set of tight, proper xy-edges having positive ZPS values. Let $E \in \mathcal{E}_0$ be arbitrary. Let $\zeta_0$ be $E$'s initial ZPS value. Suppose $E$ has been processed by the algorithm (in Step 3) $n$ times so far. Given the above strategy for choosing $R$, $E$'s current ZPS value $\zeta$ is necessarily bounded above by $\zeta_0(1-r)^n$, and hence also by $Z(1-r)^n$. If $n > \frac{\log(Z/\epsilon)}{\log(1/(1-r))}$, we get that $Z(1-r)^n < \epsilon$. Thus, after at most $\left[\lceil\frac{\log(Z/\epsilon)}{\log(1/(1-r))}\rceil + 1\right]$ appearances of $E$ in Step 3, its ZPS value $\zeta$ will be zero. Since $E$ was an arbitrary edge from $\mathcal{E}_0$ and $|\mathcal{E}_0| \leq 2|\mathcal{T}_X||\mathcal{T}_Y|$, the result is proven. ■

| | | |
|---|---|---|
| Step 2 Choice | $N$ | Randomly choose an edge from $\mathcal{E}$. |
| | $K$ | Randomly select a $K$-item subset of $\mathcal{E}$, where $K$ is one of $\{2, 4, 8\}$; choose the edge from that subset whose processing in Step 3 will result in the minimal change to the STN's rigidity. |
| Choice of $R$ | $G$ | Greedy strategy |
| | $L$ | Less-Greedy strategy where $r = 0.5$ and the computation-multiplier is either $6$ or $18$. |
| Choice of $\alpha$ | $B$ | Randomly choose $\alpha$ from $\{0, 1\}$. |
| | $U$ | Randomly choose $\alpha$ from $[0, 1]$ (uniform distribution). |
| | $F$ | Randomly choose $\alpha$ from $[0, 1]$ with distribution weighted by flexibility of $t_i z$- and $zt_j$-edges. |

Figure 5.10: Variations of the TDP Algorithm Tested

The factor $\left[\lceil \frac{\log(Z/\epsilon)}{\log(1/(1-r))} \rceil + 1\right]$ specifies an upper bound on the run-time using the Less Greedy strategy as compared to the Greedy strategy. In practice, this factor may be kept small by choosing $\epsilon$ appropriately. For example, if $r = 0.5$ and $Z/\epsilon = 1000$, this factor is 11.

**Corollary 5.29 (Completeness)** *Using either the Greedy or Less-Greedy strategy, the TDP algorithm is complete.*

**Proof** Suppose a solution exists for an instance of the TDP for an STN $\mathcal{S}$. By Lemma 5.3, $\mathcal{S}$ must be consistent. Thus, the TDP algorithm will not halt at Step 1. Using either of the strategies in Theorems 5.26 or 5.28, the algorithm will eventually terminate at Step 6. By Theorem 5.23, this only happens when the algorithm has found a solution to the TDP. ■

Finally, the subnetworks $\mathcal{S}_X$ and $\mathcal{S}_Y$ generated by the TDP algorithm may be most concisely represented by their canonical forms, as described in Section 4.2. (By Theorem 4.53, the canonical form of an STN $\hat{\mathcal{S}}$ is guaranteed to have the minimum number of edges among all STNs equivalent to $\hat{\mathcal{S}}$.)

### 5.2.5 Experimental Evaluation

This section compares the performance of the TDP algorithm across the following dimensions: (1) the function used in Step 2 to select the next edge to work on; (2) the function used in Step 3 to determine $R$ (i.e., the amount of ZPS reduction); and (3) the function used in Step 3 to determine $\alpha$, which governs the distribution of additional constrainedness among the $t_i z$- and $zt_j$-edges. Figure 5.10 shows the algorithm variations tested in the experiments. Each variation is identified by its parameter settings using the abbreviations in Figure 5.10. The goal of the experiments was to measure the tradeoff between solution

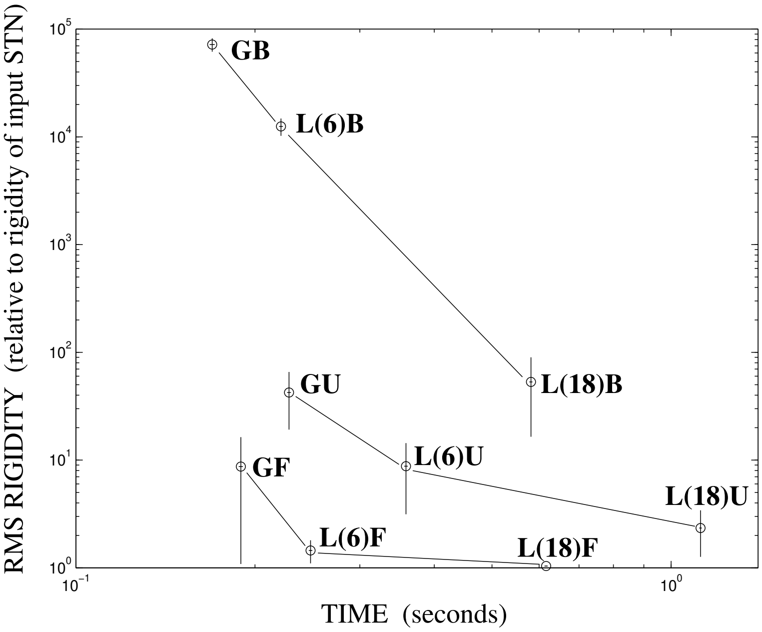



Figure 5.11: Comparing different strategies for choosing $R$ and $\alpha$ in Step 3 of the TDP algorithm

quality (as measured by the rigidity of the decoupled STN) and execution time across the different algorithm variations.

The first experiment fixed the Step 2 choice function to option $N$ (cf. Figure 5.10) while varying the Step 3 strategies for choosing $R$ and $\alpha$. It consisted of 500 trials, each restricted to the time-interval $[0, 100]$. For each trial, the STN contained start and finish time-points for 30 actions (i.e., 60 time-points). Half of the actions/time-points were allocated to $\mathcal{T}_X$, half to $\mathcal{T}_Y$. Constraints were generated randomly, as follows. For each action, a lower bound $d$ was drawn uniformly from the interval $[0, 1]$; an upper bound was drawn from $[d, d{+}1]$. Also, 400 constraints among time-points in $\mathcal{T}_X$, 400 among time-points in $\mathcal{T}_Y$, and 800 xy-edges were generated, the strength of each determined by selecting a random value from $[0, f]$, where $f$ was 30% of the maximum amount the constraint could be tightened.

The results of the first experiment are shown in Figure 5.11. The horizontal axis measures time in seconds. The vertical axis measures the RMS Rigidity (cf. Definition 4.38) of the decoupled STN (as a multiple of the RMS Rigidity of the input STN). 95% confidence intervals are shown for both time and rigidity, but the intervals for time are barely visible. Both scales are logarithmic.

Lines in the plot are used to connect variations using the same $\alpha$-selection function. In each grouping, the Greedy Strategy (G) is faster, but the Less-Greedy Strategy (L) results in decouplings that are substantially more flexible (i.e., less rigid). In addition, using a larger computation-multiplier in the Less-Greedy approach (18 vs. 6), which corresponds to a smaller value of the threshold $\epsilon$ (cf. Definition 5.27), results in decouplings that are more flexible.

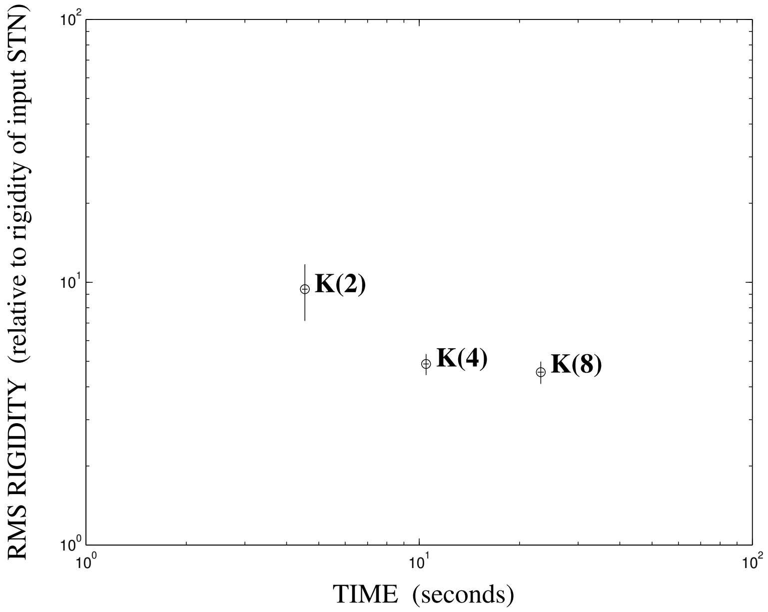



Figure 5.12: Comparing different strategies for the Step 2 choice function for the TDP algorithm

The most surprising result is the dramatic benefit from using either of the two pseudo-continuous methods, U or F, for choosing $\alpha$. Biasing the distribution according to the flexibility in the $t_i z$- and $zt_j$-edges (as is done in the Fair method, F) gives consistently more flexible decouplings, while taking less time to do so. The L(18)F variation produced decouplings that were scarcely more rigid than the input STN.

The second experiment tested the performance of the K-item-subset Step 2 choice function (i.e., option K in Figure 5.10). This Step 2 choice function is computationally intensive since for each edge in the $K$-item subset, constraints must be propagated (and reset) and the rigidity of the STN must be computed. Subset sizes of 2, 4 and 8 were tested in the experiment. The Step 3 choice functions for choosing $R$ and $\alpha$ were fixed to options that performed well in the first experiment: the Less-Greedy Strategy (L) with a computation-multiplier of 6, and the Fair method (F) for choosing $\alpha$.

The experiment consisted of 200 trials. For each trial the STN contained 40 actions (80 time-points), as well as 1600 constraints among time-points in $\mathcal{T}_X$, 1600 among time-points in $\mathcal{T}_Y$ and 3200 xy-edges. The results, shown in Figure 5.12, demonstrate that the K-item-subset Step 2 choice function can generate decoupled networks with greater flexibility. However, it is not immune to the Law of Diminishing Returns. In this case, using an 8-item subset did not appear to be worth the extra computational effort.

### 5.2.6 Minimal Temporal Decouplings

In the previous section, it was shown that the strategies employed in the TDP algorithm can have a significant impact on the quality of the decoupling generated by the algorithm (as measured by the rigidity of the decoupled network). This section defines a *minimal temporal decoupling* of an STN $\mathcal{S}$ to be a temporal decoupling of $\mathcal{S}$ with the property that any non-trivial weakening of the constraints in the decoupled subnetworks would necessarily cause them to no longer decouple $\mathcal{S}$. It then presents a theorem characterizing minimal temporal decouplings and a sound, complete and polynomial-time Iterative Weakening algorithm for constructing minimal temporal decouplings.

**Definition 5.30 (Minimal Temporal Decoupling)** *Let $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$, $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$, and $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be STNs such that $\mathcal{S}_X$ and $\mathcal{S}_Y$ temporally decouple $\mathcal{S}$. The decoupling of $\mathcal{S}$ by $\mathcal{S}_X$ and $\mathcal{S}_Y$ is called a* minimal temporal decoupling *if the following condition holds:*

> *For any STNs $\mathcal{S}'_X = (\mathcal{T}_X, \mathcal{C}'_X)$ and $\mathcal{S}'_Y = (\mathcal{T}_Y, \mathcal{C}'_Y)$ that temporally decouple $\mathcal{S}$: if the constraints in $\mathcal{C}_X$ and $\mathcal{C}_Y$ entail the constraints in $\mathcal{C}'_X$ and $\mathcal{C}'_Y$, respectively, then $\mathcal{C}_X \equiv \mathcal{C}'_X$ and $\mathcal{C}_Y \equiv \mathcal{C}'_Y$.*

**An Example of Finding a Minimal Decoupling**

Consider the STN $\mathcal{S}$ in Figure 5.13. Using the z-partition defined by $\mathcal{T}_X = \{z, x_1, x_2\}$ and $\mathcal{T}_Y = \{z, y\}$, $\mathcal{S}$ has two tight, proper xy-edges: $y - x_1 \leq 4$ and $y - x_2 \leq 4$, each of which has a zero-path shortfall of $3 + 3 - 4 = 2$. If the TDP algorithm from Figure 5.8 in Section 5.2.4 were run on this STN, for example, using the Greedy Strategy (cf. Definition 5.24), it might first process the edge $y - x_1 \leq 4$, eliminating the zero-path shortfall, say, by tightening the edge from $x_1$ to $z$, as shown in Figure 5.14. Next, it might process the edge $y - x_2 \leq 4$, eliminating the zero-path shortfall, say, by tightening the edge from $z$ to $y$, as shown in Figure 5.15, at which point all tight, proper xy-edges are dominated by paths through zero and, hence, the network is decoupled. In particular, $\mathcal{S}$ is decoupled by the subnetworks $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$, where

$$\mathcal{C}_X = \{ (z - x_1 \leq 1), (z - x_2 \leq 3) \} \quad \text{and} \quad \mathcal{C}_Y = \{ (y - z \leq 1) \}.$$

However, $\mathcal{S}_X$ and $\mathcal{S}_Y$ do not form a *minimal* decoupling of $\mathcal{S}$ since the edge $z - x_1 \leq 1$ may be replaced by the weaker, original edge $z - x_1 \leq 3$ without disturbing the decoupling, as shown in Figure 5.16.

The decoupling in Figure 5.16 *is* minimal, since weakening any of the intra-subnetwork constraints would result in subnetworks that do *not* decouple $\mathcal{S}$. Alternative minimal decouplings of the original STN are shown in Figure 5.17. However, the minimal decoupling in Figure 5.16 is the only minimal decoupling for the original STN that can be generated by weakening intra-subnetwork constraints in the non-minimal decoupling generated by the TDP algorithm. In other words, if $\mathcal{C}^m_X$ and $\mathcal{C}^m_Y$ are the constraint sets corresponding to the minimal decoupling shown in Figure 5.16, then the constraints in $\mathcal{C}_X$ and $\mathcal{C}_Y$ entail the constraints in $\mathcal{C}^m_X$ and $\mathcal{C}^m_Y$, respectively. The analogous statements do not hold for the minimal decouplings shown in Figure 5.17.

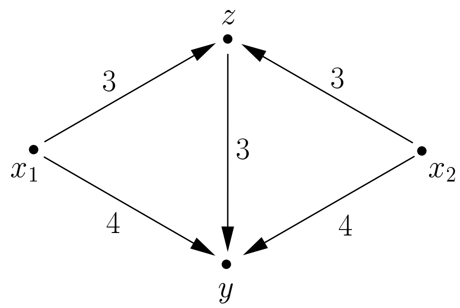



Figure 5.13: An STN for which the TDP algorithm might generate a non-minimal decoupling

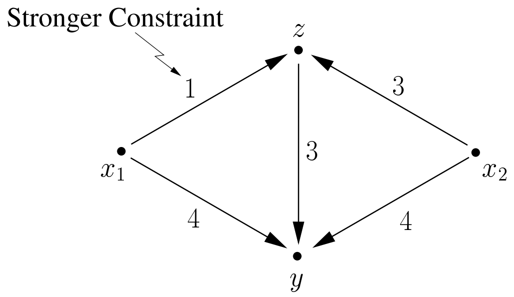



Figure 5.14: The STN from Figure 5.13 after eliminating the zero-path shortfall for one of the xy-edges

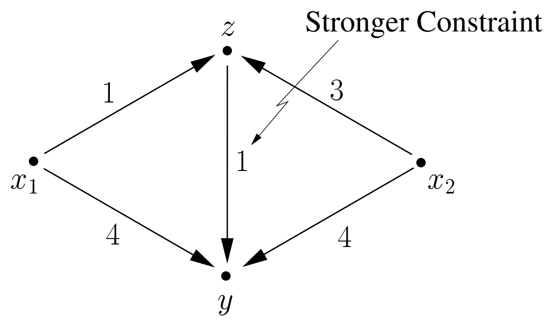



Figure 5.15: The STN from Figure 5.13 after eliminating the zero-path shortfall for both of the xy-edges

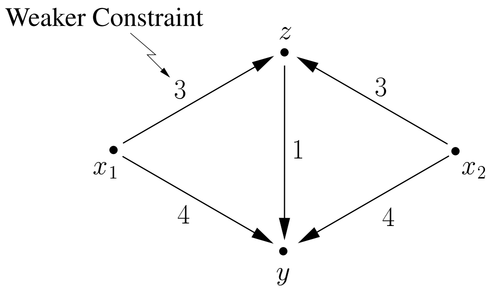



Figure 5.16: A minimal decoupling of the STN from Figure 5.13

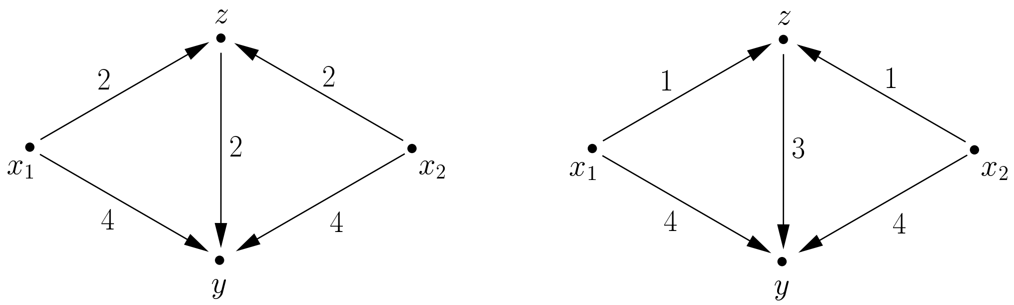



Figure 5.17: Alternative minimal decouplings of the STN from Figure 5.13

### Necessary Conditions for a Minimal Decoupling

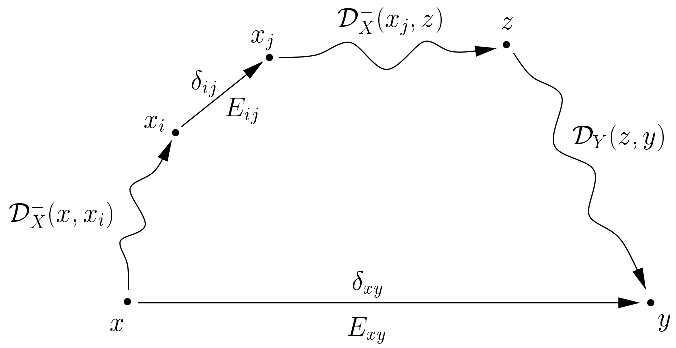
This section presents Theorem 5.33 that specifies necessary conditions for a minimal temporal decoupling. To simplify the analysis, Definition 5.31 specifies a Standard Problem Setup. Since any decoupling of $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ by $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$ requires the constraints in $\mathcal{C}_X$ and $\mathcal{C}_Y$ to entail the constraints in $\mathcal{C}_d|_{\mathcal{T}_X}$ and $\mathcal{C}_d|_{\mathcal{T}_Y}$, respectively, the constraint sets $\mathcal{C}_d|_{\mathcal{T}_X}$ and $\mathcal{C}_d|_{\mathcal{T}_Y}$ are included explicitly in the Standard Setup. However, notice that the Standard Setup does not, in and of itself, require that $\mathcal{S}_X$ and $\mathcal{S}_Y$ decouple $\mathcal{S}$.

**Definition 5.31 (Standard Setup)** *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be an STN whose time-points are z-partitioned by the sets $\mathcal{T}_X$ and $\mathcal{T}_Y$. Let $\mathcal{D}$ be the distance matrix for $\mathcal{S}$. Let $\mathcal{C}_X^0$ be an arbitrary set of constraints over time-points in $\mathcal{T}_X$ such that each $E_{ij}$: $x_j - x_i \leq \delta_{ij}$ in $\mathcal{C}_X^0$ satisfies that $\delta_{ij} < \mathcal{D}(x_i, x_j)$. Similarly, let $\mathcal{C}_Y^0$ be an arbitrary set of constraints over time-points in $\mathcal{T}_Y$ such that each $E_{pq}$: $y_q - y_p \leq \delta_{pq}$ in $\mathcal{C}_Y^0$ satisfies that $\delta_{pq} < \mathcal{D}(y_p, y_q)$. Let $\mathcal{C}_X = \mathcal{C}_d|_{\mathcal{T}_X} \cup \mathcal{C}_X^0$ and $\mathcal{C}_Y = \mathcal{C}_d|_{\mathcal{T}_Y} \cup \mathcal{C}_Y^0$. Finally, let $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$; and let $\mathcal{D}_X$ and $\mathcal{D}_Y$ be the corresponding distance matrices.*

**Definition 5.32 (Necessary Role in a Dominating Path Through Zero)** *Given the Standard Setup (cf. Definition 5.31), let $E_{ij}$: $x_j - x_i \leq \delta_{ij}$ be an arbitrary constraint in $\mathcal{C}_X^0$.*

*Let* $\mathcal{C}_X^- = \mathcal{C}_d|_{\mathcal{T}_X} \cup (\mathcal{C}_X \setminus \{E_{ij}\})$ *be the constraint set that results from removing the edge* $E_{ij}$ *from* $\mathcal{C}_X$*. Let* $\mathcal{S}_X^- = (\mathcal{T}_X, \mathcal{C}_X^-)$*, and let* $\mathcal{D}_X^-$ *be the corresponding distance matrix. Let* $E_{xy}$*:* $y - x \leq \delta_{xy}$ *be some proper xy-edge in* $\mathcal{C}$*. The edge* $E_{ij}$ *is said to play a* necessary role in a dominating path through zero *for the edge* $E_{xy}$ *with respect to* $\mathcal{S}_X$ *and* $\mathcal{S}_Y$ *if:*

- $\mathcal{D}_X^-(x, z) + \mathcal{D}_Y(z, y) > \delta_{xy}$ *(i.e., in the absence of* $E_{ij}$*, the edge* $E_{xy}$ *is not dominated by a path through zero); and*
- $\mathcal{D}_X^-(x, x_i) + \delta_{ij} + \mathcal{D}_X^-(x_j, z) + \mathcal{D}_Y(z, y) \leq \delta_{xy}$ *(i.e., in the presence of* $E_{ij}$*, the edge* $E_{xy}$ is *dominated by a path through zero, as shown below).*




*The definitions for* $E_{ij}$ *playing a necessary role for an edge* $E_{yx}$ *of the form* $x - y \leq \delta_{yx}$*, and for edges in* $\mathcal{C}_Y^0$ *playing necessary roles for xy-edges in* $\mathcal{C}$ *are analogous.*

**Theorem 5.33 (Necessary Conditions for a Minimum Decoupling)** *Given the Standard Setup (cf. Definition 5.31), suppose that* $\mathcal{S}_X$ *and* $\mathcal{S}_Y$ *form a* minimal *temporal decoupling of* $\mathcal{S} = (\mathcal{T}, \mathcal{C})$*. Let* $E_{ij}$*:* $x_j - x_i \leq \delta_{ij}$ *be an arbitrary edge in* $\mathcal{C}_X^0$ *such that the removal of* $E_{ij}$ *from* $\mathcal{C}_X^0$ *would cause a change in the distance matrix for* $\mathcal{S}_X$*. Let* $\mathcal{C}_X^-, \mathcal{S}_X^-$ *and* $\mathcal{D}_X^-$ *be as in Definition 5.32. Then* $E_{ij}$ *plays a necessary role in a dominating path through zero for some tight, proper xy-edge in* $\mathcal{C}$*. The analogous statement for edges in* $\mathcal{C}_Y^0$ *also holds.*

**Proof** Let $E_{ij}$: $x_j - x_i \leq \delta_{ij}$ be an arbitrary edge in $\mathcal{C}_X^0$ whose removal from $\mathcal{C}_X^0$ would cause a change in the distance matrix for $\mathcal{S}_X$. Let $\mathcal{C}_X^-, \mathcal{S}_X^-$ and $\mathcal{D}_X^-$ be as in Definition 5.32.

Since the constraints in $\mathcal{S}_X$ and $\mathcal{S}_Y$ entail the constraints in $\mathcal{S}_X^-$ and $\mathcal{S}_Y$, respectively, but $\mathcal{C}_X^- \not\equiv \mathcal{C}_X$ (by construction), it must be that $\mathcal{S}_X^-$ and $\mathcal{S}_Y$ do *not* temporally decouple $\mathcal{S}$. (Otherwise, it would violate that $\mathcal{S}_X$ and $\mathcal{S}_Y$ are a *minimal* decoupling of $\mathcal{S}$.) Thus, by Theorem 5.12, at least one of the Properties $1_X$, $1_Y$, $2_{XY}$ or $2_{YX}$ from Theorem 5.10 fails to hold (with respect to $\mathcal{S}_X^-$ and $\mathcal{S}_Y$).

Since the constraints in $\mathcal{S}_X^-$ and $\mathcal{S}_Y$ entail the constraints in $\mathcal{C}_d|_{\mathcal{T}_X}$ and $\mathcal{C}_d|_{\mathcal{T}_Y}$, respectively, Properties $1_X$ and $1_Y$ necessarily hold. Thus, some instance of Properties $2_{XY}$ or $2_{YX}$ from Theorem 5.10 fails to hold. Thus, by Lemma 5.18, there must be at least one tight, proper xy-edge $E_{xy}$ in $\mathcal{C}$ that is *not* dominated by a path through zero using edges in $\mathcal{S}_X^-$ and $\mathcal{S}_Y$. Without loss of generality, suppose $E_{xy}$ has the form $y - x \leq \delta_{xy}$, where $x \in \mathcal{T}_X$ and $y \in \mathcal{T}_Y$. Then,

$$\mathcal{D}_X^-(x, z) + \mathcal{D}_Y(z, y) \; > \; \delta_{xy}.$$

Suppose that $E_{ij}$ does *not* play a necessary role in the dominating path through zero (with respect to $\mathcal{S}_X$ and $\mathcal{S}_Y$) for $E_{xy}$. Given the above inequality, which is the first condition in Definition 5.32, it must be that

$$\mathcal{D}_X^-(x, x_i) + \delta_{ij} + \mathcal{D}_X^-(x_j, z) + \mathcal{D}_Y(z, y) \;>\; \delta_{xy},$$

which is the negation of the second condition in Definition 5.32. However, this pair of inequalities together imply that the edge $E_{xy}$ is not dominated by a path through zero using edges in $\mathcal{C}_X$ and $\mathcal{C}_Y$, whether using $E_{ij}$ or not, which contradicts that $\mathcal{S}_X$ and $\mathcal{S}_Y$ decouple $\mathcal{S}$. Thus, the result is proven for the edge $E_{ij}$ in $\mathcal{C}_X^0$.

Edges in $\mathcal{C}_Y^0$ whose removal from $\mathcal{C}_Y$ would cause a change in the distance matrix for $\mathcal{S}_Y$ are handled analogously. ■

**The Iterative Weakening Algorithm for Constructing a Minimal Decoupling**

This section presents a sound and complete algorithm for constructing a minimal temporal decoupling that runs in polynomial time. The algorithm works by weakening intra-subnetwork edges in a non-minimal decoupling (e.g., one produced by the TDP algorithm from Section 5.2.4) until a minimal decoupling is found. The amount by which each intra-subnetwork edge $E$ may be weakened is determined by the *mu bound* for $E$ (cf. Definition 5.34, below). The mu bound for $E$ ensures that the tight, proper xy-edges in $\mathcal{C}$ continue to be dominated by paths through zero despite any weakening of $E$.

**Definition 5.34 (Mu Bound)** *Given the Standard Setup (cf. Definition 5.31), let* $E_{ij}$*:* $x_j - x_i \leq \delta_{ij}$ *be an arbitrary edge in* $\mathcal{C}_X^0$*. Let* $\mathcal{C}_X^-, \mathcal{S}_X^-$ *and* $\mathcal{D}_X^-$ *be as in Definition 5.32.*

*Let* $\mathcal{C}^Z$ *be the set of all tight, proper xy-edges in* $\mathcal{C}$ *that are not dominated by a path through zero in the absence of the edge* $E_{ij}$*. In other words,* $\mathcal{C}^Z = \mathcal{C}_{xy}^Z \cup \mathcal{C}_{yx}^Z$*, where:*

$$\begin{aligned}
\mathcal{C}_{xy}^Z \;&=\; \{\, (y - x \leq \delta_{xy}) \in \mathcal{C} : x \in \mathcal{T}_X,\ y \in \mathcal{T}_Y,\ \delta_{xy} = \mathcal{D}(x, y) \\
&\qquad\qquad \text{and}\ \mathcal{D}_X^-(x, z) + \mathcal{D}_Y(z, y) \;>\; \delta_{xy} \,\};\ \text{ and} \\
\mathcal{C}_{yx}^Z \;&=\; \{\, (x - y \leq \delta_{yx}) \in \mathcal{C} : x \in \mathcal{T}_X,\ y \in \mathcal{T}_Y,\ \delta_{yx} = \mathcal{D}(y, x) \\
&\qquad\qquad \text{and}\ \mathcal{D}_Y(y, z) + \mathcal{D}_X^-(z, x) \;>\; \delta_{yx} \,\}.
\end{aligned}$$

*If* $\mathcal{C}^Z$ *is empty, then the* mu bound *for the edge* $E_{ij}$ *is not defined; otherwise, it is given by the following:*

$$\mu(E_{ij}) \;=\; \min \left\{ \begin{array}{l} \{\, \delta_{xy} - \mathcal{D}_X^-(x, x_i) - \mathcal{D}_X^-(x_j, z) - \mathcal{D}_Y(z, y) : (y - x \leq \delta_{xy}) \in \mathcal{C}_{xy}^Z \,\} \\ \cup\ \{\, \delta_{yx} - \mathcal{D}_Y^-(y, z) - \mathcal{D}_X^-(z, x_i) - \mathcal{D}_X^-(x_j, x) : (x - y \leq \delta_{yx}) \in \mathcal{C}_{yx}^Z \,\} \end{array} \right.$$

*The mu bound for an edge of the form* $E_{pq}$*:* $y_q - y_p \leq \delta_{pq}$ *in* $\mathcal{C}_Y^0$ *is defined analogously.*

**Lemma 5.35 (Properties of the Mu Bound)** *Given the Standard Setup (cf. Definition 5.31) but such that* $\mathcal{S}_X$ *and* $\mathcal{S}_Y$ *temporally decouple* $\mathcal{S}$*, let* $E_{ij}$*:* $x_j - x_i \leq \delta_{ij}$ *be an arbitrary* tight *constraint in* $\mathcal{C}_X^0$*. Let* $\mathcal{C}_X^-, \mathcal{S}_X^-$ *and* $\mathcal{D}_X^-$ *be as in Definition 5.32. Let the set* $\mathcal{C}^Z$ *and the mu bound* $\mu(E_{ij})$ *be as in Definition 5.34. If the set* $\mathcal{C}^Z$ *is non-empty, then* $\delta_{ij} \leq \mu(E_{ij}) < \infty$*. The analogous statement applies to tight edges* $E_{pq}$ *in* $\mathcal{C}_Y^0$*.*

**Proof** Suppose that the set $\mathcal{C}^Z$ is non-empty. Thus, $\mu(E_{ij})$ is well-defined and, furthermore, for some edge $E_{xy}$ in $\mathcal{C}^Z$

$$\mu(E_{ij}) \;=\; \delta_{xy} - \mathcal{D}_X^-(x, x_i) - \mathcal{D}_X^-(x_j, z) - \mathcal{D}_Y(z, y),$$

where, without loss of generality, it has been assumed that $E_{xy}$ has the form, $y - x \leq \; \delta_{xy}$. Since $\mathcal{S}$ is temporally decoupled by $\mathcal{S}_X$ and $\mathcal{S}_Y$, these STNs are all consistent (cf. Definition 5.2 and Lemma 5.3); and since removing edges from a consistent STN cannot make that STN inconsistent, the STN $\mathcal{S}_X^-$ is necessarily consistent, as well. Thus, none of the distance matrix entries in the above equation are negative infinity. Thus, $\mu(E_{ij}) < \infty$.

Now suppose that $\mu(E_{ij}) < \delta_{ij}$. Given the above equation, this implies that

$$\delta_{xy} - \mathcal{D}_X^-(x, x_i) - \mathcal{D}_X^-(x_j, z) - \mathcal{D}_Y(z, y) \;=\; \mu(E_{ij}) \;<\; \delta_{ij},$$

which, in turn, implies that

$$\delta_{xy} \;<\; \mathcal{D}_X^-(x, x_i) + \delta_{ij} + \mathcal{D}_X^-(x_j, z) + \mathcal{D}_Y(z, y).$$

However, this implies that the edge $E_{xy}$ is not dominated by a path through zero containing the edge $E_{ij}$. However, $E_{xy}$ being in $\mathcal{C}^Z$ implies that $E_{xy}$ is not dominated by a path through zero that does *not* contain the edge $E_{ij}$. These statements, taken together, contradict that $E_{xy}$ is dominated by some path through zero (whether containing $E_{ij}$ or not) in the temporal decoupling of $\mathcal{S}$ by $\mathcal{S}_X$ and $\mathcal{S}_Y$. Thus, the assumption that $\mu(E_{ij}) < \delta$ was wrong. ■

The Iterative Weakening algorithm for constructing a minimal temporal decoupling is given in pseudo-code in Figure 5.18. The algorithm takes as input a possibly non-minimal temporal decoupling of $\mathcal{S}$ by $\tilde{\mathcal{S}}_X$ and $\tilde{\mathcal{S}}_Y$. After replacing $\tilde{\mathcal{S}}_X$ and $\tilde{\mathcal{S}}_Y$ by equivalent STNs in the form specified in the Standard Setup (cf. Definition 5.31), the algorithm processes the edges from $\mathcal{C}_X^0 \cup \mathcal{C}_Y^0$ in some unspecified order. (Each edge is processed exactly once.) For each edge, the algorithm computes a helper distance matrix ($\mathcal{D}_X^-$ or $\mathcal{D}_Y^-$) and the set $\mathcal{C}^Z$ (cf. Definition 5.34). If $\mathcal{C}^Z$ is empty, the edge is safely removed from $\mathcal{C}_X^0 \cup \mathcal{C}_Y^0$; otherwise, the corresponding mu bound is computed (cf. Definition 5.34) and the edge is replaced in $\mathcal{C}_X \cup \mathcal{C}_Y$ by a possibly weaker edge whose strength is specified by the mu bound. The algorithm returns a minimal temporal decoupling whose *modified* constraint sets $\mathcal{C}_X$ and $\mathcal{C}_Y$ are entailed by the input constraint sets in $\tilde{\mathcal{S}}_X$ and $\tilde{\mathcal{S}}_Y$, respectively.

**Theorem 5.36** *The Iterative Weakening algorithm for constructing a minimal temporal decoupling is sound and complete.*

**Proof** Since the Iterative Weakening algorithm has at most $|\mathcal{T}_X|^2 + |\mathcal{T}_Y|^2$ iterations, it always terminates. Thus, it suffices to check that subnetworks returned by the algorithm form a minimal temporal decoupling of $\mathcal{S}$.

To avoid confusion between the constraint sets $\mathcal{C}_X$ and $\mathcal{C}_Y$ constructed in Step 0, and those returned by the algorithm, the latter shall be denoted by $\hat{\mathcal{C}}_X$ and $\hat{\mathcal{C}}_Y$, respectively. Similarly, $\hat{\mathcal{S}}_X = (\mathcal{T}_X, \hat{\mathcal{C}}_X)$, $\hat{\mathcal{S}}_Y = (\mathcal{T}_X, \hat{\mathcal{C}}_Y)$, $\hat{\mathcal{D}}_X$ and $\hat{\mathcal{D}}_Y$ denote the corresponding subnetworks and distance matrices existing at the end of the algorithm.

**Given:** STNs $\tilde{\mathcal{S}}_X = (\mathcal{T}_X, \tilde{\mathcal{C}}_X)$, $\tilde{\mathcal{S}}_Y = (\mathcal{T}_Y, \tilde{\mathcal{C}}_Y)$ and $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ such that $\tilde{\mathcal{S}}_X$ and $\tilde{\mathcal{S}}_Y$ temporally decouple $\mathcal{S}$.

(0) Let $\mathcal{C}_X = \mathcal{C}_d|_{\mathcal{T}_X} \cup \mathcal{C}_X^0$ and $\mathcal{C}_Y = \mathcal{C}_d|_{\mathcal{T}_Y} \cup \mathcal{C}_Y^0$ as specified in the Standard Setup (cf. Definition 5.31), but such that $\mathcal{C}_X \equiv \tilde{\mathcal{C}}_X$ and $\mathcal{C}_Y \equiv \tilde{\mathcal{C}}_Y$. Let $\mathcal{S}_X, \mathcal{S}_Y, \mathcal{D}_X$ and $\mathcal{D}_Y$ be the corresponding STNs and distance matrices (again, as in the Standard Setup).

(1) Initialize $\mathcal{E}$ to the set $\mathcal{C}_X^0 \cup \mathcal{C}_Y^0$.

(2) If $\mathcal{E}$ is empty, go to Step 5. Otherwise, *remove* some edge $E$ from $\mathcal{E}$. If $E \in \mathcal{C}_X^0$, go to Step 3a; otherwise, go to Step 4a.

(3a) Let $\mathcal{C}_X^- = \mathcal{C}_d|_{\mathcal{T}_X} \cup (\mathcal{C}_X \setminus \{E\})$ and $\mathcal{S}_X^- = (\mathcal{T}_X, \mathcal{C}_X^-)$ be as in Definition 5.32. Compute the corresponding distance matrix $\mathcal{D}_X^-$.

(3b) Compute the set $\mathcal{C}^Z = \mathcal{C}_{xy}^Z \cup \mathcal{C}_{yx}^Z$ of tight, proper xy-edges in $\mathcal{C}$ that are not dominated by a path through zero in the absence of the edge $E$, as specified in Definition 5.34.

(3c) If $\mathcal{C}^Z$ is empty, remove $E$ from the set $\mathcal{C}_X^0$ (it does not make a necessary contribution to the decoupling of $\mathcal{S}$) and go back to Step 2.

(3d) Compute the mu bound $\mu$ for the edge $E$, as specified in Definition 5.34. Since $E \in \mathcal{C}_X^0$, it has the form $x_j - x_i \leq \delta_{ij}$. If $\mu \geq \mathcal{D}(x_i, x_j)$, then simply *remove* the edge $E$ from $\mathcal{C}_X^0$. If $\mu > \delta_{ij}$, then *replace* the edge $E$ in $\mathcal{C}_X^0$ with the weaker edge $E_\mu$: $x_j - x_i \leq \mu$. (If $\mu = \delta_{ij}$, the edge $E$ cannot be weakened.) Go back to Step 2.

(4a) Let $\mathcal{C}_Y^- = \mathcal{C}_d|_{\mathcal{T}_Y} \cup (\mathcal{C}_Y \setminus \{E\})$ and $\mathcal{S}_Y^- = (\mathcal{T}_Y, \mathcal{C}_Y^-)$ be as in Definition 5.32. Compute the corresponding distance matrix $\mathcal{D}_Y^-$.

(4b) Compute the set $\mathcal{C}^Z = \mathcal{C}_{xy}^Z \cup \mathcal{C}_{yx}^Z$ of tight, proper xy-edges in $\mathcal{C}$ that are not dominated by a path through zero in the absence of the edge $E$, as specified in Definition 5.34.

(4c) If $\mathcal{C}^Z$ is empty, remove $E$ from the set $\mathcal{C}_Y^0$ (it does not make a necessary contribution to the decoupling of $\mathcal{S}$) and go back to Step 2.

(4d) Compute the mu bound $\mu$ for the edge $E$, as specified in Definition 5.34. Since $E \in \mathcal{C}_Y^0$, it has the form $y_q - y_p \leq \delta_{pq}$. If $\mu \geq \mathcal{D}(y_p, y_q)$, then simply *remove* the edge $E$ from $\mathcal{C}_Y^0$. If $\mu > \delta_{pq}$, then *replace* the edge $E$ in $\mathcal{C}_Y^0$ with the weaker edge $E_\mu$: $y_q - y_p \leq \mu$. (If $\mu = \delta_{pq}$, the edge $E$ cannot be weakened.) Go back to Step 2.

(5) **Return:** $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_d|_{\mathcal{T}_X} \cup \mathcal{C}_X^0)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_d|_{\mathcal{T}_Y} \cup \mathcal{C}_Y^0)$, where $\mathcal{C}_X^0$ and $\mathcal{C}_Y^0$ may have been *modified* during the course of the algorithm.

Figure 5.18: The Iterative Weakening algorithm for constructing a minimal temporal decoupling

(Proof that $\hat{\mathcal{S}}_X$ and $\hat{\mathcal{S}}_Y$ temporally decouple $\mathcal{S}$) Suppose not. Then there must have been some point in the algorithm at which the decoupling property first failed to hold. Let $E$ be the edge whose removal or replacement caused the decoupling property to fail for the first time. In other words, prior to removing or replacing $E$, the decoupling property held, but immediately afterward it did not. Without loss of generality, suppose the edge $E = E_{ij}$ has the form, $x_j - x_i \leq \delta_{ij}$, and let $\mathcal{C}^Z$ and, if applicable, $\mu_{ij} = \mu(E_{ij})$ be as specified in Definition 5.34.

First, since $\mathcal{C}_d|_{\mathcal{T}_X} \subseteq \mathcal{C}_X$ and $\mathcal{C}_d|_{\mathcal{T}_Y} \subseteq \mathcal{C}_Y$ throughout the algorithm, Properties $1_X$ and $1_Y$ of Theorem 5.10 hold with respect to $\mathcal{S}_X$ and $\mathcal{S}_Y$ throughout the algorithm. Thus, by Theorem 5.12, the only way the decoupling property can fail to hold is by some instance of Properties $2_{XY}$ or $2_{YX}$ failing to hold. Thus, by Lemma 5.18, the only way the decoupling property can fail to hold is by some tight, proper xy-edge in $\mathcal{S}$ failing to be dominated by some path through zero using edges in $\mathcal{C}_X$ and $\mathcal{C}_Y$.

If $\mathcal{C}^Z$ is empty, then there are no tight, proper xy-edges in $\mathcal{C}$ that require the presence of $E_{ij}$. Thus, removing $E_{ij}$ could not disturb the property that each tight, proper xy-edge in $\mathcal{C}$ is dominated by a path through zero.

On the other hand, suppose $\mathcal{C}^Z$ is non-empty. Let $E_{xy}$ be an arbitrary edge in $\mathcal{C}^Z$. Without loss of generality, $E_{xy}$ has the form, $y - x \leq \delta_{xy}$. By definition of the mu bound,

$$\mu_{ij} \;\leq\; \delta_{xy} - \mathcal{D}^{-}_{X,alg}(x, x_i) - \mathcal{D}^{-}_{X,alg}(x_j, z) - \mathcal{D}_{Y,alg}(z, y),$$

where the *alg* subscripts are used to indicate that the distance matrices are those that existed during the computation of $\mu_{ij}$ by the algorithm (i.e., prior to replacing or removing the edge $E_{ij}$). However, this implies that

$$\mathcal{D}^{-}_{X,alg}(x, x_i) + \mu_{ij} + \mathcal{D}^{-}_{X,alg}(x_j, z) + \mathcal{D}_{Y,alg}(z, y) \;\leq\; \delta_{xy},$$

which implies that the edge $E_{xy}$ is dominated by a path through zero even *after* replacing the edge $E_{ij}$ by the edge $E_\mu$: $x_j - x_i \leq \mu_{ij}$.

Since $E_{xy}$ was arbitrary in $\mathcal{C}^Z$, all of the tight, proper xy-edges in $\mathcal{C}^Z$ must be dominated by paths through zero even after replacing $E_{ij}$ by $E_\mu$. However, all of the tight, proper xy-edges in $\mathcal{C}$ that are *not* in $\mathcal{C}^Z$ do not require $E_{ij}$ for their dominating paths (cf. Definition 5.34). Thus, every tight, proper xy-edge in $\mathcal{C}$ is dominated by a path through zero even after replacing $E_{ij}$ by $E_\mu$. (If $\mu_{ij} \leq \mathcal{D}(x_i, x_j)$, the above argument still applies, but there is no need to insert the edge $E_\mu$ into $\mathcal{C}^0_X$, since the corresponding edge in $\mathcal{C}_d|_{\mathcal{T}_X}$ is at least as strong as $E_\mu$.)

Thus, in either case, the temporal decoupling property holds after the processing of the edge $E_{ij}$, which is a contradiction. Since there is no point in the algorithm at which the decoupling property first fails, the decoupling property must hold all the way through the algorithm, including at the very end.

(Proof that $\hat{\mathcal{S}}_X$ and $\hat{\mathcal{S}}_Y$ are a *minimal* temporal decoupling of $\mathcal{S}$) Let $\mathcal{S}'_X = (\mathcal{T}_X, \mathcal{C}'_X)$ and $\mathcal{S}'_Y = (\mathcal{T}_Y, \mathcal{C}'_Y)$ be arbitrary STNs such that the constraints in $\hat{\mathcal{C}}_X$ and $\hat{\mathcal{C}}_Y$ entail the constraints in $\mathcal{C}'_X$ and $\mathcal{C}'_Y$, respectively, but that $\hat{\mathcal{C}}_X \not\equiv \mathcal{C}'_X$ or $\hat{\mathcal{C}}_Y \not\equiv \mathcal{C}'_Y$. Without loss of generality, assume that $\hat{\mathcal{C}}_X \not\equiv \mathcal{C}'_X$. Without loss of generality, further assume that $\hat{\mathcal{C}}^0_X$ only contains tight edges. Finally, assume that the constraints in $\mathcal{C}'_X$ and $\mathcal{C}'_Y$ entail the constraints in $\mathcal{C}_d|_{\mathcal{T}_X}$ and $\mathcal{C}_d|_{\mathcal{T}_Y}$, respectively, since otherwise, by Theorem 5.10, $\mathcal{S}'_X$ and $\mathcal{S}'_Y$ could not form a temporal decoupling of $\mathcal{S}$.

It is sufficient to show that $\mathcal{S}'_X$ and $\mathcal{S}'_Y$ do *not* temporally decouple $\mathcal{S}$. Thus, it is sufficient to show that there is some tight, proper xy-edge in $\mathcal{C}$ that is *not* dominated by a path through zero with respect to the subnetworks $\mathcal{S}'_X$ and $\mathcal{S}'_Y$.

Since $\hat{\mathcal{C}}_X \not\equiv \mathcal{C}'_X$, and since the constraints in $\mathcal{C}'_X$ entail those in $\mathcal{C}_d|_{\mathcal{T}_X}$, there must exist some edge $E_\mu$: $x_j - x_i \leq \mu_{ij}$ in $\hat{\mathcal{C}}^0_X$, where $x_i, x_j \in \mathcal{T}_X$ and

$$\mu_{ij} \;=\; \hat{\mathcal{D}}_X(x_i, x_j) \;<\; \mathcal{D}'_X(x_i, x_j).$$

Notice that since each edge in $\mathcal{C}^0_X$ is processed exactly once in the algorithm, the value $\mu_{ij}$ must be the value that was computed during the algorithm when the original edge $E_{ij}$ from $x_i$ to $x_j$ in $\mathcal{C}^0_X$ was processed.

Let $\mathcal{C}^Z_{alg}$ be the set of xy-edges in $\mathcal{C}$ that required the presence of the edge $E_{ij}$ for their dominating paths through zero with respect to the constraint sets $\mathcal{C}_{X,alg}$ and $\mathcal{C}_{Y,alg}$, where the $alg$ subscripts indicate that the constraint sets are those that existed during the processing of $E_{ij}$ by the algorithm. Notice that $\mathcal{C}^Z_{alg}$ cannot be empty since otherwise the edge $E_{ij}$ would have been discarded without replacement, contradicting the existence of the edge $E_\mu$ in $\hat{\mathcal{C}}^0_X$.

Let $E_{xy} \in \mathcal{C}^Z_{alg} \subseteq \mathcal{C}$ be such that

$$\mu_{ij} \;=\; \delta_{xy} - \mathcal{D}^-_{X,alg}(x, x_i) - \mathcal{D}^-_{X,alg}(x_j, z) - \mathcal{D}_{Y,alg}(z, y),$$

where it has been assumed, without loss of generality, that $E_{xy}$ has the form, $y - x \leq \delta_{xy}$. Rearranging terms yields the following:

$$\delta_{xy} \;=\; \mathcal{D}^-_{X,alg}(x, x_i) + \mu_{ij} + \mathcal{D}^-_{X,alg}(x_j, z) + \mathcal{D}_{Y,alg}(z, y).$$

Since the replacement edges are never stronger than the edges they replace (cf. Lemma 5.35), the distance matrix entries can only ever stay the same or increase during the course of the algorithm. Therefore, the above equality implies the following:

$$\delta_{xy} \;\leq\; \hat{\mathcal{D}}^-_X(x, x_i) + \mu_{ij} + \hat{\mathcal{D}}^-_X(x_j, z) + \hat{\mathcal{D}}_Y(z, y),$$

where $\hat{\mathcal{C}}^-_X = \mathcal{C}_d|_{\mathcal{T}_X} \cup (\hat{\mathcal{C}}^0_X \setminus \{E_\mu\})$ and $\hat{\mathcal{S}}^-_X = (\mathcal{T}_X, \hat{\mathcal{C}}^-_X)$, and where $\hat{\mathcal{D}}^-_X$ is the corresponding distance matrix. Since $\mu_{ij} < \mathcal{D}'_X(x_i, x_j)$, the above inequality implies the following *strict* inequality:

$$\delta_{xy} \;<\; \hat{\mathcal{D}}^-_X(x, x_i) + \mathcal{D}'_X(x_i, x_j) + \hat{\mathcal{D}}^-_X(x_j, z) + \hat{\mathcal{D}}_Y(z, y).$$

Furthermore, since the constraints in $\hat{\mathcal{C}}_X$ entail the constraints in $\mathcal{C}'_X$, the above inequality yields the following:

$$\delta_{xy} \;<\; \mathcal{D}'^-_X(x, x_i) + \mathcal{D}'_X(x_i, x_j) + \mathcal{D}'^-_X(x_j, z) + \mathcal{D}'_Y(z, y),$$

where ${\mathcal{C}'_X}^-$ is the constraint set $\mathcal{C}'_X$ minus any edge from $x_i$ to $x_j$ that might exist in $\mathcal{C}'_X$ and ${\mathcal{S}'_X}^- = (\mathcal{T}_X, {\mathcal{C}'_X}^-)$, and where ${\mathcal{D}'_X}^-$ is the corresponding distance matrix. However, this implies that the edge $E_{xy}$ is *not* dominated by any path $P$ through zero, using edges in $\mathcal{C}'_X \cup \mathcal{C}'_Y$ and such that $P$ contains a subpath from $x_i$ to $x_j$. However, the definition of $\mathcal{C}^Z$ (of which $E_{xy}$ is a member) implies that $E_{xy}$ is not dominated by any paths through zero using edges only from $\hat{\mathcal{C}}_X^-$ and $\hat{\mathcal{C}}_Y$. Since the constraints in $\hat{\mathcal{C}}_X^-$ and $\hat{\mathcal{C}}_Y$ entail the constraints in ${\mathcal{C}'_X}^-$ and $\mathcal{C}'_Y$, respectively, it follows that $E_{xy}$ is not dominated by any paths through zero using edges only from ${\mathcal{C}'_X}^-$ and $\mathcal{C}'_Y$. Thus, $E_{xy}$ is not dominated by any path through zero, using edges in $\mathcal{C}'_X \cup \mathcal{C}'_Y$, whether including a subpath from $x_i$ to $x_j$ or not, which implies that $\mathcal{S}'_X$ and $\mathcal{S}'_Y$ do *not* temporally decouple $\mathcal{S}$.

Since $\mathcal{S}'_X$ and $\mathcal{S}'_Y$ were arbitrary (such that their constraint sets were entailed by $\hat{\mathcal{C}}_X$ and $\hat{\mathcal{C}}_Y$, and such that $\hat{\mathcal{C}}_X \not\equiv \mathcal{C}'_X$ or $\hat{\mathcal{C}}_Y \not\equiv \mathcal{C}'_Y$), the STNs $\hat{\mathcal{S}}_X$ and $\hat{\mathcal{S}}_Y$ have been shown to be a *minimal* temporal decoupling of $\mathcal{S}$. ■

Returning to the non-minimal decoupling shown in Figure 5.15, notice that the Iterative Weakening algorithm, when applied to the edge from $x_1$ to $z$, would compute the constraint set $\mathcal{C}^Z = \{(y - x \leq 4)\}$ and $\mu = 4 - 0 - 0 - 1 = 3$. Thus, the edge $z - x_1 \leq 1$ would be replaced by the strictly weaker edge $z - x_1 \leq 3$, resulting in the minimal decoupling shown in Figure 5.16.

### 5.2.7 The Optimal Temporal Decoupling Problem

This section defines the Optimal Temporal Decoupling Problem in which a temporal decoupling that maximizes some metric $h$ of solution quality is sought. A candidate metric might measure the flexibility of the decoupled network. Variations on this theme might take into account the flexibility of the decoupled subnetworks as well. In the case of an agent decoupling a portion of its Master STN in preparation for its participation in a task-allocation auction, the metric of solution quality might need to include some measure of how much *space* is available for tasks the agent might want to bid on.

**Definition 5.37 (Temporal Decoupling Metric)** *A function $h$ is a* metric of the quality of temporal decouplings *if for any STNs $\mathcal{S}$, $\mathcal{S}_X$ and $\mathcal{S}_Y$ such that $\mathcal{S}_X$ and $\mathcal{S}_Y$ temporally decouple $\mathcal{S}$, $h(\mathcal{S}, \mathcal{S}_X, \mathcal{S}_Y)$ specifies some real number.*

A sample metric $h_r$, based on the measure of STN rigidity defined in Section 4.1.4, is given by:

$$h_r((\mathcal{T}, \mathcal{C}),\ (\mathcal{T}_X, \mathcal{C}_X),\ (\mathcal{T}_Y, \mathcal{C}_Y)) \;=\; -Rig((\mathcal{T}, \mathcal{C} \cup \mathcal{C}_X \cup \mathcal{C}_Y)),$$

where the negative sign ensures that decouplings with greater flexibility have higher quality.

**Definition 5.38 (Optimal Temporal Decoupling Problem)** *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be an STN whose time-points $\mathcal{T}$ are z-partitioned by the sets $\mathcal{T}_X$ and $\mathcal{T}_Y$. Let $h$ be a metric of the quality of temporal decouplings. Find sets $\mathcal{C}_X$ and $\mathcal{C}_Y$ of constraints over time-points in $\mathcal{T}_X$ and $\mathcal{T}_Y$, respectively, that maximize $h((\mathcal{T}, \mathcal{C}),\ (\mathcal{T}_X, \mathcal{C}_X),\ (\mathcal{T}_Y, \mathcal{C}_Y))$ subject to the constraint that $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$ temporally decouple $\mathcal{S}$.*

**Definition 5.39 (Metric Strongly Biased Toward Flexibility)** *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be an STN whose time-points $\mathcal{T}$ are z-partitioned by the sets $\mathcal{T}_X$ and $\mathcal{T}_Y$. A metric $h$ of the quality of temporal decouplings is said to be* strongly biased toward flexibility *if*

$$h((\mathcal{T}, \mathcal{C}),\ (\mathcal{T}_X, \mathcal{C}_X),\ (\mathcal{T}_Y, \mathcal{C}_Y)) \quad < \quad h((\mathcal{T}, \mathcal{C}),\ (\mathcal{T}_X, \mathcal{C}'_X),\ (\mathcal{T}_Y, \mathcal{C}'_Y))$$

*whenever the constraints in $\mathcal{C}_X$ and $\mathcal{C}_Y$ entail the constraints in $\mathcal{C}'_X$ and $\mathcal{C}'_Y$, respectively.*

Notice that for any metric that is strongly biased toward flexibility, including the metric $h_r$ defined above, any *minimal* temporal decoupling (cf. Definition 5.30) represents a *locally* optimal temporal decoupling.

Finding globally optimal temporal decouplings (i.e., solutions to the Optimal Temporal Decoupling Problem) is beyond the scope of this thesis. That would enable comparing the solution quality attained by the algorithm varieties tested in Section 5.2.5 against the quality of the optimal solution, according to various metrics.

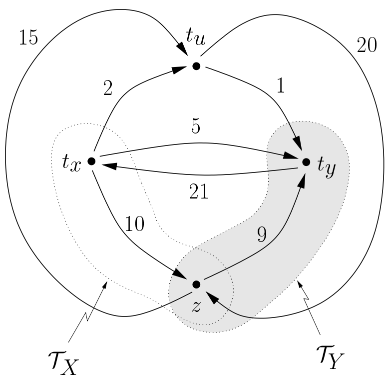
## 5.3 The Allocative Temporal Decoupling Problem

In certain applications (e.g., when an agent needs to decouple a portion of its Master STN in preparation for its participation in a task-allocation auction), the initial contents of the sets $\mathcal{T}_X$ and $\mathcal{T}_Y$ (and hence the z-partition for the subsequent decoupling) may be only partially specified. In such scenarios, the Temporal Decoupling Problem requires not only the addition of intra-subnetwork constraints sufficient to decouple the network, but also the completion of the partially specified z-partition.

This section defines the Allocative Temporal Decoupling Problem in which (1) a partially specified z-partition must be completed, and (2) a temporal decoupling must be constructed using that z-partition. Like the Optimal TDP in Section 5.2.7, the Allocative Temporal Decoupling Problem is cast as an optimization problem, based on a metric $h$ of the quality of temporal decouplings.

This section also presents a greedy, incremental algorithm for finding approximate solutions to the Allocative TDP. The algorithm interleaves (1) the incremental allocation of time-points to $\mathcal{T}_X$ and $\mathcal{T}_Y$ with (2) the incremental addition of intra-subnetwork temporal constraints aimed at providing the eventual decoupling. The incremental-allocation decision-making functionality is encapsulated by an *oracle* function $\mathcal{F}$ which, given a partially decoupled network (based on a partially specified z-partition), generates sets $\Delta\mathcal{T}_X$ and $\Delta\mathcal{T}_Y$ of time-points to add to $\mathcal{T}_X$ and $\mathcal{T}_Y$, respectively. The incremental addition of intra-subnetwork constraints is handled by the basic TDP algorithm, modified to focus on a restricted set of tight, proper xy-edges. For guidance, each process requires the metric $h$ to be able to measure the quality not only of fully decoupled networks, but also of *partially* decoupled networks.

The rest of this section is organized as follows. Section 5.3.1 introduces the Allocative Temporal Decoupling Problem by walking through a sample instance; Section 5.3.2




| | $z$ | $t_x$ | $t_y$ | $t_u$ |
|---|---|---|---|---|
| $z$ | 0 | 30 | 9 | 15 |
| $t_x$ | 10 | 0 | 3 | 2 |
| $t_y$ | 31 | 21 | 0 | 23 |
| $t_u$ | 20 | 22 | 1 | 0 |

$$\begin{aligned} \text{RMS}_X &= 0.024 \\ \text{RMS}_Y &= 0.024 \\ \text{RMS}_{XY} &= 0.040 \\ \text{RMS}_{\text{all}} &= 0.034 \end{aligned}$$

Figure 5.19: A sample STN with an initial allocation of time-points to $\mathcal{T}_X$ and $\mathcal{T}_Y$

formally defines the Allocative TDP; Section 5.3.3 presents an algorithm for finding approximate solutions to the Allocative TDP; and Section 5.3.4 applies the Allocative TDP algorithm to the Auction-Participation-Context-Generation Problem from Chapter 3.

### 5.3.1 A Sample Instance of the Allocative TDP

Figure 5.19 shows a sample STN containing four time-points: $z, t_x, t_y$ and $t_u$. Initially, $\mathcal{T}_X$ contains $z$ and $t_x$; $\mathcal{T}_Y$ contains $z$ and $t_y$; and $t_u$ is unallocated. The only proper xy-edges[2] are:

$$E_{xy}\colon t_y - t_x \leq 5 \quad \text{and} \quad E_{yx}\colon t_x - t_y \leq 21.$$

$E_{yx}$ is a tight edge; $E_{xy}$ is not (because there is a path from $t_x$ to $t_y$ via $t_u$ that has length 3, which is less than the length of $E_{xy}$). The corresponding distance matrix is also shown in the figure, as are several relevant measures of rigidity (cf. Definition 4.38). $\text{RMS}_X$ measures the RMS rigidity of edges among points in $\mathcal{T}_X$; $\text{RMS}_Y$ measures the RMS rigidity of edges among points in $\mathcal{T}_Y$; $\text{RMS}_{XY}$ measures the RMS rigidity of edges joining points in $\mathcal{T}_X$ to points in $\mathcal{T}_Y$; and $\text{RMS}_{\text{all}}$ measures the RMS rigidity of the entire network.

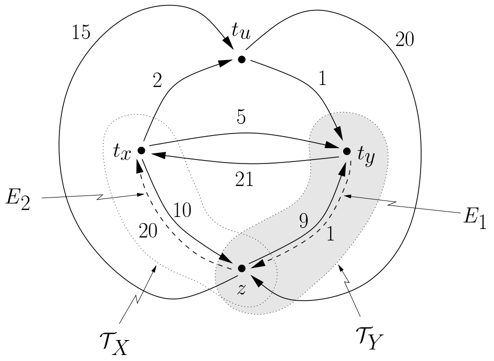
Adding the edges

$$E_1\colon z - t_y \leq 1 \;\; (\text{i.e., } \delta_1 = 1) \quad \text{and} \quad E_2\colon t_x - z \leq 20 \;\; (\text{i.e., } \delta_2 = 20),$$

shown as dashed arcs in Figure 5.20, provides a path through zero that dominates the edge $E_{yx}$.[3] Adding the edges $E_1$ and $E_2$ to the network causes its RMS rigidity to increase from

[2]Throughout this section, it is presumed that the notion of an xy-edge (cf. Definition 5.13) is extended (in the obvious way) to include the case where the sets $\mathcal{T}_X$ and $\mathcal{T}_Y$ do not form a proper z-partition of $\mathcal{T}$.

[3]Notice that the updated distance matrix entries satisfy, $\mathcal{D}(t_y, z) = \delta_1 = 1$ and $\mathcal{D}(z, t_x) = \delta_2 = 20$, as predicted by Corollary 5.20.




| | $z$ | $t_x$ | $t_y$ | $t_u$ |
|---|---|---|---|---|
| $z$ | 0 | 20 | 9 | 15 |
| $t_x$ | 10 | 0 | 3 | 2 |
| $t_y$ | 1 | 21 | 0 | 16 |
| $t_u$ | 2 | 22 | 1 | 0 |

$$\begin{aligned} \text{RMS}_X &= 0.040 \\ \text{RMS}_Y &= 0.091 \\ \text{RMS}_{XY} &= 0.040 \\ \text{RMS}_{\text{all}} &= 0.057 \end{aligned}$$

Figure 5.20: A partial decoupling of the STN from Figure 5.19

$0.034$ to $0.057$.[4]

At this point, there are two ways to proceed: the unallocated time-point $t_u$ may be allocated to either $\mathcal{T}_X$ or $\mathcal{T}_Y$. Allocating $t_u$ to $\mathcal{T}_X$, as shown in Figure 5.21, introduces the new tight, proper xy-edge, $E_{uy}$: $t_y - t_u \leq 1$. Adding the edges

$$E_1'\colon z - t_u \leq -5.5 \quad \text{and} \quad E_2'\colon t_y - z \leq 6.5,$$

shown as dashed arcs in Figure 5.22, provides a path through zero that dominates the edge $E_{uy}$. The subnetworks corresponding to $\mathcal{T}_X$ and $\mathcal{T}_Y$ are now fully decoupled. The RMS rigidity of the entire network has increased to $0.075$.

On the other hand, allocating $t_u$ to $\mathcal{T}_Y$, as shown in Figure 5.23, introduces the new tight, proper xy-edge, $E_{xu}$: $t_u - t_x \leq 2$. Adding the edges

$$E_1''\colon z - t_x \leq 3.15 \quad \text{and} \quad E_2''\colon t_u - z \leq -1.15,$$

shown as dashed arcs in Figure 5.24, provides a path through zero that dominates the edge $E_{xu}$. Once again, the subnetworks corresponding to $\mathcal{T}_X$ and $\mathcal{T}_Y$ are temporally decoupled. This time, the RMS rigidity of the entire network is $0.383$, which is substantially larger.

### 5.3.2 Definition of the Allocative TDP

**Definition 5.40 (The Allocative TDP)** *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be a consistent STN. Let $\mathcal{T}_X^0$ and $\mathcal{T}_Y^0$ be sets of time-points such that $\mathcal{T}_X^0 \cap \mathcal{T}_Y^0 = \{z\}$ and $\mathcal{T}_X^0 \cup \mathcal{T}_Y^0 \subseteq \mathcal{T}$. Find:*

[4] Notice that the inequality $\mathcal{D}(t_x, z) + \mathcal{D}(z, t_y) \leq \mathcal{D}(t_x, t_y)$ (cf. Lemma 5.18) does *not* hold, even though $(t_x, t_y)$ is a proper xy-pair. Thus, the subnetworks corresponding to $\mathcal{T}_X$ and $\mathcal{T}_Y$ are not yet fully decoupled, even though all tight, proper xy-edges (with respect to $\mathcal{T}_X$ and $\mathcal{T}_Y$) are dominated by paths through zero. This does not contradict Lemma 5.18 because that lemma relies on a strict definition of tight, proper xy-edges arising from sets $\mathcal{T}_X$ and $\mathcal{T}_Y$ that properly z-partition $\mathcal{T}$, which is not the case here.

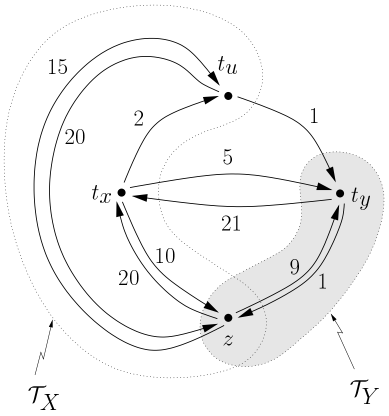
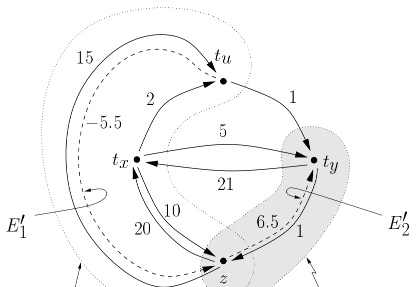



Figure 5.21: Allocating $t_u$ to $\mathcal{T}_X$




| | $z$ | $t_x$ | $t_y$ | $t_u$ |
|---|---|---|---|---|
| $z$ | 0 | 20 | 6.5 | 15 |
| $t_x$ | -3.5 | 0 | 3 | 2 |
| $t_y$ | 1 | 21 | 0 | 16 |
| $t_u$ | -5.5 | 14.5 | 1 | 0 |

$$\begin{aligned}
\text{RMS}_X &= 0.072\\
\text{RMS}_Y &= 0.118\\
\text{RMS}_{XY} &= 0.048\\
\text{RMS}_{\text{all}} &= 0.075
\end{aligned}$$

Figure 5.22: A decoupling in which $t_u \in \mathcal{T}_X$

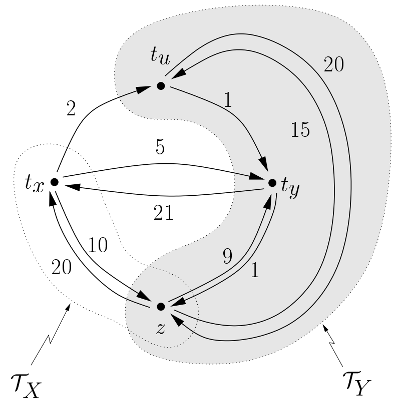
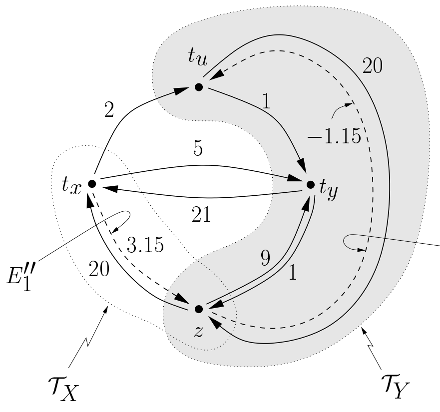



Figure 5.23: Allocating $t_u$ to $\mathcal{T}_Y$




| | $z$ | $t_x$ | $t_y$ | $t_u$ |
|---|---|---|---|---|
| $z$ | 0 | 20 | 0.15 | -1.15 |
| $t_x$ | -3.5 | 0 | 3 | 2 |
| $t_y$ | 1 | 21 | 0 | -0.15 |
| $t_u$ | 2 | 22 | 1 | 0 |

$$\begin{aligned}
\text{RMS}_X &= 0.163 \\
\text{RMS}_Y &= 0.517 \\
\text{RMS}_{XY} &= 0.040 \\
\text{RMS}_{\text{all}} &= 0.383
\end{aligned}$$

Figure 5.24: A decoupling in which $t_u \in \mathcal{T}_Y$

- *sets* $\mathcal{T}_X$ *and* $\mathcal{T}_Y$ *of time-points such that* $\mathcal{T}_X^0 \subseteq \mathcal{T}_X$, $\mathcal{T}_Y^0 \subseteq \mathcal{T}_Y$, *and* $\mathcal{T}_X$ *and* $\mathcal{T}_Y$ *z-partition* $\mathcal{T}$*; and*

- *sets* $\mathcal{C}_X$ *and* $\mathcal{C}_Y$ *of constraints over the time-points in* $\mathcal{T}_X$ *and* $\mathcal{T}_Y$ *such that* $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ *and* $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$ *temporally decouple* $\mathcal{S}$*,*

*such that the quantity* $h(\mathcal{S}, \mathcal{S}_X, \mathcal{S}_Y)$ *is maximized.*

### 5.3.3 An Algorithm for Finding Approximate Solutions to the Allocative TDP

This section presents a greedy, incremental algorithm for finding approximate solutions to the Allocative TDP. The algorithm interleaves two processes: (1) the incremental allocation of time-points to the sets $\mathcal{T}_X$ and $\mathcal{T}_Y$; and (2) the incremental addition of intra-subnetwork temporal constraints. Each process requires the metric $h$ to be able to measure the quality not only of fully decoupled networks, but also of partially decoupled networks. The incremental allocation of time-points to $\mathcal{T}_X$ and $\mathcal{T}_Y$ is encapsulated by an oracle function $\mathcal{F}$. The incremental addition of intra-subnetwork temporal constraints is handled by the basic TDP algorithm from Section 5.2.4, modified to restrict attention to a limited pool of tight, proper xy-edges.

Since the contents of $\mathcal{T}_X$ and $\mathcal{T}_Y$ determine whether any given edge is classified as an xy-edge, every new allocation of a time-point to $\mathcal{T}_X$ or $\mathcal{T}_Y$ has the potential to cause previously unclassified edges to become newly classified as xy-edges. Thus, the pool of tight, proper xy-edges that is the central focus of the basic TDP algorithm (cf. Section 5.2.4) has the potential to grow over time. The algorithm presented in this section lazily adds intra-subnetwork constraints sufficient to ensure that every edge in the *current* pool of tight, proper xy-edges is dominated by a path through zero. This is accomplished by modifying the initialization step of the basic TDP algorithm such that only the most recent additions to the pool of tight, proper xy-edges are included in the set $\mathcal{E}$. No further modification of the basic TDP algorithm is required. By the time the specification of the z-partition is completed, and the latest additions to the pool of tight, proper xy-edges have been dominated by paths through zero, the network is guaranteed to be decoupled.

The greedy, incremental approach to finding approximate solutions to the Allocative TDP requires that the metric $h$ be able to measure the quality not only of fully decoupled networks, but also of *partially* decoupled networks. The metric $h$ can be used not only to guide the Step 2 and Step 3 choice functions of the basic TDP algorithm (cf. Section 5.2.4), but also to guide incremental-allocation decision-making of the oracle function $\mathcal{F}$.

**Definition 5.41 (Requirements for the Oracle Function)** *Given an STN* $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ *and subnetworks* $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ *and* $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$*, where* $\mathcal{T}_X \cap \mathcal{T}_Y = \{z\}$ *and* $\mathcal{T}_X \cup \mathcal{T}_Y \subset \mathcal{T}$*, find sets* $\Delta\mathcal{T}_X$ *and* $\Delta\mathcal{T}_Y$ *such that:*

$$\left.\begin{array}{rcl} \Delta\mathcal{T}_X & \subseteq & (\mathcal{T} \setminus (\mathcal{T}_X \cup \mathcal{T}_Y)) \\ \Delta\mathcal{T}_Y & \subseteq & (\mathcal{T} \setminus (\mathcal{T}_X \cup \mathcal{T}_Y)) \end{array}\right\} \quad \textit{Can only allocate currently unallocated time-points.}$$

$$(\Delta\mathcal{T}_X \cup \Delta\mathcal{T}_Y) \neq \emptyset \qquad \text{Must } \textit{allocate at least one time-point}$$

$$(\Delta\mathcal{T}_X \cap \Delta\mathcal{T}_Y) = \emptyset \qquad \textit{Each time-point must be } \text{uniquely } \textit{allocated}$$

Pseudo-code for the Allocative TDP algorithm is given in Figure 5.25. The algorithm takes as input a metric $h$, an oracle function $\mathcal{F}$, a consistent STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$, and sets $\mathcal{T}_X^0$ and $\mathcal{T}_Y^0$ that partially specify a z-partition of $\mathcal{T}$. The output of the algorithm is a pair of subnetworks, $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$, where $\mathcal{T}_X^0 \subseteq \mathcal{T}_X$ and $\mathcal{T}_Y^0 \subseteq \mathcal{T}_Y$, such that $\mathcal{S}_X$ and $\mathcal{S}_Y$ temporally decouple $\mathcal{S}$.

Each iteration of the algorithm (Steps 1 through 5) assumes that all pre-existing tight, proper xy-edges (i.e., tight, proper xy-edges with respect to the sets $\mathcal{T}_X$ and $\mathcal{T}_Y$) are already dominated by paths through zero. The two-part goal of each iteration is (1) to allocate additional time-points to $\mathcal{T}_X$ and $\mathcal{T}_Y$ and (2) to add intra-subnetwork constraints sufficient to ensure that all tight, proper xy-edges with respect to the sets $(\mathcal{T}_X \cup \Delta\mathcal{T}_X)$ and $(\mathcal{T}_Y \cup \Delta\mathcal{T}_Y)$ are dominated by paths through zero.

Since the input sets $\mathcal{T}_X^0$ and $\mathcal{T}_Y^0$ represent an initial, incremental allocation of time-points, the algorithm prepares for the first iteration by converting this initial allocation into the form expected in Step 2. In particular, $\mathcal{T}_X$ and $\mathcal{T}_Y$ are set to $\{z\}$, $\Delta\mathcal{T}_X$ is set to $(\mathcal{T}_X^0 \setminus \{z\})$, and $\Delta\mathcal{T}_Y$ is set to $(\mathcal{T}_Y^0 \setminus \{z\})$. Notice that, given this setup, there are *no* pre-existing tight, proper xy-edges (with respect to $\mathcal{T}_X$ and $\mathcal{T}_Y$). Thus, the assumption that all pre-existing tight, proper xy-edges are dominated by paths through zero holds trivially. Because the oracle function is not needed for the first iteration, the algorithm then jumps to Step 2. Subsequent iterations begin at Step 1, using the oracle function to generate the sets $\Delta\mathcal{T}_X$ and $\Delta\mathcal{T}_Y$ of time-points to add to $\mathcal{T}_X$ and $\mathcal{T}_Y$.

The rest of an iteration proceeds as follows. In Step 2, $\mathcal{E}$ is assigned to the set of all tight, proper xy-edges in $\mathcal{C}$ with respect to the sets $(\mathcal{T}_X \cup \Delta\mathcal{T}_X)$ and $(\mathcal{T}_Y \cup \Delta\mathcal{T}_Y)$ whose ZPS values are positive and which are *not* pre-existing xy-edges (i.e., are *not* xy-edges with respect to $\mathcal{T}_X$ and $\mathcal{T}_Y$). Thus, $\mathcal{E}$ contains edges only of the form $t_j - t_i \leq \delta$, where:

$$\begin{array}{llll} t_i \in \mathcal{T}_X & \text{and} & t_j \in \Delta\mathcal{T}_Y & ; \\ t_i \in \Delta\mathcal{T}_X & \text{and} & t_j \in \mathcal{T}_Y & ; \text{ or} \\ t_i \in \Delta\mathcal{T}_X & \text{and} & t_j \in \Delta\mathcal{T}_Y & . \end{array}$$

The number of edges in $\mathcal{E}$ is bounded above by

$$|\mathcal{T}_X||\Delta\mathcal{T}_Y| + |\Delta\mathcal{T}_X||\mathcal{T}_Y| + |\Delta\mathcal{T}_X||\Delta\mathcal{T}_Y|$$

which, assuming $|\Delta\mathcal{T}_X| \ll |\mathcal{T}_X|$ and $|\Delta\mathcal{T}_Y| \ll |\mathcal{T}_Y|$, represents a small fraction of the xy-edges with respect to $(\mathcal{T}_X \cup \Delta\mathcal{T}_X)$ and $(\mathcal{T}_Y \cup \Delta\mathcal{T}_Y)$.

Once $\mathcal{E}$ has been initialized, Steps 2 through 6 of the basic TDP algorithm (Figure 5.8) are performed, after which all tight, proper xy-edges with respect to $(\mathcal{T}_X \cup \Delta\mathcal{T}_X)$ and $(\mathcal{T}_Y \cup \Delta\mathcal{T}_Y)$ are guaranteed to be dominated by paths through zero. (Recall that the basic TDP algorithm destructively modifies the STN $\mathcal{S}$ and its distance matrix $\mathcal{D}$.) The final step of the iteration is to add the time-points in $\Delta\mathcal{T}_X$ and $\Delta\mathcal{T}_Y$ to $\mathcal{T}_X$ and $\mathcal{T}_Y$, respectively.

**Given:** A metric $h$, an oracle function $\mathcal{F}$, a consistent STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$, and sets $\mathcal{T}_X^0$ and $\mathcal{T}_Y^0$ such that $\mathcal{T}_X^0 \cap \mathcal{T}_Y^0 = \{z\}$ and $\mathcal{T}_X^0 \cup \mathcal{T}_Y^0 \subseteq \mathcal{T}$.

**Do the following:**

(0) Initialization: 

$$\begin{array}{rclcrcl} \Delta\mathcal{T}_X & = & (\mathcal{T}_X^0 \setminus \{z\}) & \quad & \mathcal{T}_X & = & \{z\} \\ \Delta\mathcal{T}_Y & = & (\mathcal{T}_Y^0 \setminus \{z\}) & \quad & \mathcal{T}_Y & = & \{z\} \end{array}$$

Go to Step 2.

(1) Apply the oracle function $\mathcal{F}$ to $\langle \mathcal{S},\ (\mathcal{T}_X, \mathcal{C}_d|_{\mathcal{T}_X}),\ (\mathcal{T}_Y, \mathcal{C}_d|_{\mathcal{T}_Y}) \rangle$ to generate new sets $\Delta\mathcal{T}_X$ and $\Delta\mathcal{T}_Y$ of time-points that will be added to $\mathcal{T}_X$ and $\mathcal{T}_Y$ in Step 4.

(2) Set $\mathcal{E}$ to the set of *tight* edges in $\mathcal{C}$ having positive ZPS values and restricted to the form, $t_j - t_i \leq \delta$, where:

$$\begin{array}{lcll} t_i \in \mathcal{T}_X & \text{and} & t_j \in \Delta\mathcal{T}_Y & ;\ \text{or} \\ t_i \in \Delta\mathcal{T}_X & \text{and} & t_j \in \mathcal{T}_Y & ;\ \text{or} \\ t_i \in \Delta\mathcal{T}_X & \text{and} & t_j \in \Delta\mathcal{T}_Y & . \end{array}$$

(3) Run the basic TDP algorithm from Figure 5.8 starting at Step 2, using the metric $h$, if desired, to guide the Step 2 and Step 3 choice functions. (In the process, the STN $\mathcal{S}$ and its distance matrix $\mathcal{D}$ may be modified. The outputs $\mathcal{C}_X$ and $\mathcal{C}_Y$ of the basic TDP algorithm may be ignored.)

(4) Add the time-points in $\Delta\mathcal{T}_X$ and $\Delta\mathcal{T}_Y$ to $\mathcal{T}_X$ and $\mathcal{T}_Y$, respectively.

(5) If $(\mathcal{T}_X \cup \mathcal{T}_Y) \neq \mathcal{T}$, go back to Step 1; otherwise …

**Return:** $(\mathcal{T}_X, \mathcal{C}_d|_{\mathcal{T}_X})$ and $(\mathcal{T}_Y, \mathcal{C}_d|_{\mathcal{T}_Y})$, where $\mathcal{C}$ includes all of the constraints added during the various calls to the basic TDP algorithm (from Step 3 above), and $\mathcal{D}$ has been updated accordingly. If desired, the Iterative Weakening algorithm from Section 5.2.6 may be applied to ensure that a minimal temporal decoupling is returned.

Figure 5.25: An algorithm for finding approximate solutions to the Allocative TDP

If some time-points in $\mathcal{T}$ have not yet been allocated, then at least one more iteration needs to be performed; otherwise, the algorithm is done and returns the subnetworks $(\mathcal{T}_X, \mathcal{C}_d|_{\mathcal{T}_X})$ and $(\mathcal{T}_Y, \mathcal{C}_d|_{\mathcal{T}_Y})$, which are guaranteed to temporally decouple $\mathcal{S}$.

**Theorem 5.42 (Soundness, Completeness)** *Given an oracle function $\mathcal{F}$ satisfying the requirements in Definition 5.41, the algorithm for finding approximate solutions to the Allocative Temporal Decoupling Problem is guaranteed to terminate with a temporal decoupling after at most $|\mathcal{T}|$ iterations.*

**Proof** At the end of each iteration of the Allocative TDP algorithm, edges in the current pool of tight, proper xy-edges are dominated by paths through zero. Since ZPS values cannot ever increase (cf. Lemma 5.17), the edges processed during one iteration never need to be revisited during any subsequent iteration. Thus, the main difference between the Allocative TDP algorithm and the basic TDP algorithm (cf. Section 5.2.4) is the *order* in which the tight, proper xy-edges are processed. Thus, the soundness result for the basic TDP algorithm (cf. Theorem 5.23) ensures that the Allocative TDP algorithm is sound.

Since the oracle function is required to allocate at least one time-point per iteration, the Allocative TDP algorithm is guaranteed to terminate after at most $|\mathcal{T}|$ iterations. ■

The greedy, incremental algorithm for finding approximate solutions to the Allocative Temporal Decoupling Problem is guided by the metric $h$ of the quality of possibly-partial temporal decouplings and the oracle function $\mathcal{F}$ for making incremental allocation decisions. The metric $h$ can be used to guide the Step 2 and Step 3 choice functions of the basic TDP algorithm. In addition, it can be used to guide the oracle function. The next section applies the Allocative TDP algorithm to the Auction-Participation-Context-Generation Problem described in Chapter 3. It provides a sample oracle function and discusses the dependence of the metric $h$ on the application domain.

### 5.3.4 The APC-Generation Problem

Prior to participating in the type of task-allocation auction described in Chapter 3, an agent must decide what portion (i.e., subnetwork) of its Master STN to use as the basis for its auction-related computations. To reduce the computational burden of participating in an auction, the selected subnetwork, referred to as the agent's Auction-Participation-Context (APC), should be kept as small as possible. To allow the auction-related computations to be carried out independently, the APC subnetwork must be temporally decoupled from the rest of the Master STN.

In this section, $\mathcal{T}_X$ represents the time-points allocated to the APC subnetwork, $\mathcal{T}_Y$ the time-points allocated to remain with the Master STN, and $\mathcal{T}_U$ the time-points about which an allocation decision has not yet been made.

#### An Oracle Function for the APC-Generation Problem

A sample oracle function, $\mathcal{F}_{APC}$, for an application of the Allocative TDP algorithm to the APC-Generation Problem is given in Figure 5.26. The algorithm takes as input a partially

**Given:** STNs $\mathcal{S} = (\mathcal{T}, \mathcal{C})$, $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ and $\mathcal{S}_Y = (\mathcal{T}_Y, \mathcal{C}_Y)$ such that:

$$\mathcal{T}_X \subseteq \mathcal{T}; \qquad \mathcal{T}_Y \subseteq \mathcal{T}; \qquad \mathcal{T}_X \cap \mathcal{T}_Y = \{z\}; \qquad \text{and} \qquad \mathcal{T}_X \cup \mathcal{T}_Y \subset \mathcal{T},$$

and a metric $h$ that measures the quality of (possibly partial) temporal decouplings.

(0) Let $\mathcal{T}_U = \mathcal{T} \setminus (\mathcal{T}_X \cup \mathcal{T}_Y)$ be the set of currently unallocated time-points in $\mathcal{T}$.

(1) Randomly select a time-point $t_u$ from $\mathcal{T}_U$.

(2) Do Steps 2 and 3 of the Allocative TDP algorithm (Figure 5.25) using $\Delta\mathcal{T}_X = \{t_u\}$ and $\Delta\mathcal{T}_Y = \emptyset$. (Notice that constraints may be added to $\mathcal{C}$ in the process.)

(3) Let $h_X = h(\mathcal{S},\ (\mathcal{T}_X \cup \{t_u\}, \mathcal{C}_d|_{(\mathcal{T}_X \cup \{t_u\})}),\ (\mathcal{T}_Y, \mathcal{C}_d|_{\mathcal{T}_Y}))$.

(4) Restore $\mathcal{S}$ by removing any constraints from $\mathcal{C}$ that may have been added during Step 2 and recomputing the distance matrix $\mathcal{D}$.

(5) Do Steps 2 and 3 of the Allocative TDP algorithm (Figure 5.25) using $\Delta\mathcal{T}_X = \emptyset$ and $\Delta\mathcal{T}_Y = \{t_u\}$. (Once again, constraints may be added to $\mathcal{C}$ in the process.)

(6) Let $h_Y = h(\mathcal{S},\ (\mathcal{T}_X, \mathcal{C}_d|_{\mathcal{T}_X}),\ (\mathcal{T}_Y \cup \{t_u\}, \mathcal{C}_d|_{(\mathcal{T}_Y \cup \{t_u\})}))$.

(7) Restore $\mathcal{S}$ by removing any constraints from $\mathcal{C}$ that may have been added during Step 5 and recomputing the distance matrix $\mathcal{D}$.

**Return:** $\begin{cases} \Delta\mathcal{T}_X = \{t_U\} \ \text{ and } \ \Delta\mathcal{T}_Y = \emptyset \ , & \text{if } h_X \geq h_Y \\ \Delta\mathcal{T}_X = \emptyset \ \text{ and } \ \Delta\mathcal{T}_Y = \{t_U\} \ , & \text{otherwise.} \end{cases}$

Figure 5.26: An oracle function for an application of the Allocative TDP algorithm to the APC-Generation Problem

decoupled network (based on a partially specified z-partition) and returns as output sets, $\Delta\mathcal{T}_X$ and $\Delta\mathcal{T}_Y$, such that $|\Delta\mathcal{T}_X| + |\Delta\mathcal{T}_Y| = 1$. In other words, each time it is called, $\mathcal{F}_{APC}$ allocates a single time-point to either $\Delta\mathcal{T}_X$ or $\Delta\mathcal{T}_Y$.

In Step 1, a time-point $t_u$ is randomly selected from the pool of currently unallocated time-points $\mathcal{T}_U$. In Step 2, an iteration of the Allocative TDP algorithm is executed to determine the effect of allocating $t_u$ to $\Delta\mathcal{T}_X$. Notice that doing so typically involves adding constraints to $\mathcal{C}$ to ensure that any newly created tight, proper xy-edges are dominated by paths through zero. The quality of this (typically partial) decoupling is scored by the metric $h$ and stored in the variable $h_X$ (Step 3). In Step 4, $\mathcal{S}$ is restored to its original state. In Step 5, an iteration of the Allocative TDP algorithm is again executed, this time to determine the effect of allocating $t_u$ to $\mathcal{T}_Y$. Once again, the metric $h$ is used to score the quality of the resulting (typically partial) decoupling, this time storing the result in the variable $h_Y$. In Step 7, $\mathcal{S}$ is once again restored to its original state. Finally, $\mathcal{F}_{APC}$ returns the allocation decision that received a better score (i.e., $t_u \in \Delta\mathcal{T}_X$ or $t_u \in \Delta\mathcal{T}_Y$).

**Choosing a Metric for the APC-Generation Problem**

The purpose of the metric $h$ is to measure the goodness of (typically partial) decouplings based on (possibly partial) allocations represented by the sets $\mathcal{T}_X$ and $\mathcal{T}_Y$. In general, the choice of metric depends on the preferences of the agent. In the context of the APC-Generation Problem, reasonable criteria to be considered by a metric include:

- The flexibility of the decoupled subnetworks; and
- The ability of the subnetwork corresponding to the auction-participation context to accommodate new tasks that the agent might be interested in bidding on.

If the APC subnetwork is highly flexible, then the agent will likely be better able to generate bids containing flexible temporal constraints, thereby increasing the likelihood of its bids being awardable. Similarly, if the rest of its Master STN is highly flexible, then an agent will likely be better able to accept offers to do other activities should they arise. However, the APC subnetwork must also be able to accommodate new tasks that the agent might want to bid on. The following example, illustrated in Figure 5.27, highlights some of the relevant issues.

**Example.** Suppose that $\mathcal{T}_X = \{z, t_x\}$ is the set of points currently allocated to the auction-participation context, and that $\mathcal{T}_Y = \{z, t_{y_1}, t_{y_2}\}$ is the set of points currently allocated to remain with the Master STN. Further suppose that $t_{y_1}$ and $t_{y_2}$ are rigidly connected such that $t_{y_2}$ is constrained to occur precisely 10 minutes after $t_{y_1}$. (In Figure 5.27, the rigid constraint among $t_{y_1}$ and $t_{y_2}$ is represented by a thick, horizontal bar of length 10.) Let $I = [I_1, I_2]$ be a fixed interval on the time-line such that all of the tasks associated with the auction are constrained to lie within the interval $I$. Suppose that $t_{y_1}$ is constrained to lie outside of $I$, but that $t_{y_2}$ (although allocated to $\mathcal{T}_Y$) is constrained to lie within $I$. Let $a$ and $b$ be the lower and upper bounds on $t_{y_1}$. Similarly, let $c$ and $d$ be the lower and upper bounds on $t_{y_2}$. Notice that the rigid constraint among $t_{y_1}$ and $t_{y_2}$ implies that $c = a + 10$

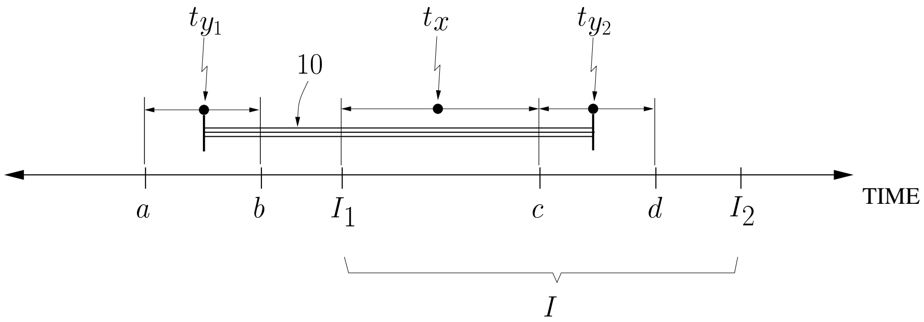



Figure 5.27: Sample scenario for the APC-Generation Problem

and $d = b + 10$. Finally, suppose that the lower and upper bounds for $t_x$ are $I_1$ and $c$, respectively.

Notice that, as shown in Figure 5.27, the subnetworks corresponding to $\mathcal{T}_X$ and $\mathcal{T}_Y$ are temporally decoupled. Notice further that the flexibility afforded to $t_{y_2}$ within the interval $[c, d]$ (subject to $t_{y_1}$ moving in parallel within the interval $[a, b]$) is due to $t_{y_2}$ having been allocated to $\mathcal{T}_Y$. If $t_{y_2}$ had instead been allocated to $\mathcal{T}_X$, then temporally decoupling $\mathcal{T}_X$ and $\mathcal{T}_Y$ would have required fixing both $t_{y_1}$ and $t_{y_2}$ on the time-line. (In general, rigidly connected time-points may be decoupled only by fixing them on the time-line.)

Nonetheless, the flexibility afforded to $t_{y_2}$ by having allocated it to $\mathcal{T}_Y$ comes with a cost, namely: it reduces the amount of space within the interval $I$ for accommodating new tasks that the agent might want to bid on. For example, assuming that the agent cannot do two things at once, any time-point $t^*$ associated with a new task that the agent might want to bid on must be constrained to lie outside the interval $[c, d]$. Otherwise, $t^*$, which is required to belong to $\mathcal{T}_X$, would introduce a dependence among $\mathcal{T}_X$ and $\mathcal{T}_Y$, thus subverting the decoupling. Thus, a metric $h$ must consider the tradeoff between the flexibility of the decoupled subnetworks and the ability of the subnetwork representing the auction-participation context to accommodate new tasks that the agent might want to bid on. The Allocative TDP algorithm provides a structure within which a variety of candidate metrics may be applied.

Given a suitable metric $h$ and the oracle function $\mathcal{F}_{APC}$, the Allocative TDP algorithm enables an agent to incrementally decide which points from its Master STN it will allocate to its auction-participation context. The basis for making each allocation decision may include not only the immediate impact on the flexibility of the decoupled networks, but also the amount of space in the auction-participation context for accommodating tasks that the agent might be interested in bidding on.

## 5.4 The General Temporal Decoupling Problem

This section generalizes the temporal decoupling problem presented in Section 5.2 to the case where the z-partition of $\mathcal{T}$ contains arbitrarily many subsets, $\mathcal{T}_1, \ldots, \mathcal{T}_n$. The treatment of the General TDP in Sections 5.4.1 through 5.4.4 (below) parallels the treatment of the $n = 2$ special case of the TDP presented earlier in Sections 5.2.1 through 5.2.4. For completeness, Sections 5.4.1 and 5.4.2 provide the definitions and theorems in the more general setting. However, the proofs, which are entirely analogous to their counterparts in Sections 5.2.1 and 5.2.2, are omitted. Since most of the definitions, theorems and proofs from Sections 5.2.3 and 5.2.4 (concerning xy-edges and xy-pairs) require only a minor terminological change to carry over to the more general setting, only the required terminological change is presented. Those results from Sections 5.2.3 and 5.2.4 that require more involved translation are presented explicitly, along with the General TDP algorithm itself, in Sections 5.4.3 and 5.4.4.

The General TDP algorithm has applications to the Post-Auction Coordination Problem for the sorts of task-allocation auctions described in Chapter 3. Such applications will be discussed at the end of this section.

### 5.4.1 Formal Definition of the General TDP

To simplify the presentation, the variables $r$ and $s$ are assumed to be restricted to the set $\{1, 2, \ldots, n\}$ in all that follows.

**Definition 5.43 (z-Partition)** *Let $\mathcal{T}, \mathcal{T}_1, \ldots, \mathcal{T}_n$ be sets of time-point variables. We say that $\mathcal{T}_1, \ldots, \mathcal{T}_n$* z-partition *$\mathcal{T}$ if:*

- *$\mathcal{T}_r \cap \mathcal{T}_s = \{z\}$, for all $r \neq s$; and*
- *$\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_n = \mathcal{T}$.*

Figure 5.28 shows a sample z-partition involving four subsets of time-points.

**Definition 5.44 (General Temporal Decoupling)** *$\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1), \ldots, \mathcal{S}_n = (\mathcal{T}_n, \mathcal{C}_n)$ are called a* temporal decoupling *of the STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ if:*

- *$\mathcal{S}_1, \ldots, \mathcal{S}_n$ are consistent STNs;*
- *$\mathcal{T}_1, \ldots, \mathcal{T}_n$ z-partition $\mathcal{T}$; and*
- *(Mergeable Solutions Property)* Any *solutions $\mathcal{X}_1, \ldots, \mathcal{X}_n$ for $\mathcal{S}_1, \ldots, \mathcal{S}_n$, respectively, may be merged to form a solution for $\mathcal{S}$.*

**Lemma 5.45** *If $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are a temporal decoupling of $\mathcal{S}$, then $\mathcal{S}$ is consistent.*

**Lemma 5.46** *If $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are a temporal decoupling of $(\mathcal{T}, \mathcal{C} \cup \mathcal{C}')$, then $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are also a temporal decoupling of $(\mathcal{T}, \mathcal{C})$.*

**Lemma 5.47** *If $\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1), \ldots, \mathcal{S}_n = (\mathcal{T}_n, \mathcal{C}_n)$ are a temporal decoupling of $(\mathcal{T}, \mathcal{C})$, then $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are also a temporal decoupling of $(\mathcal{T}, \mathcal{C} \cup \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_n)$.*

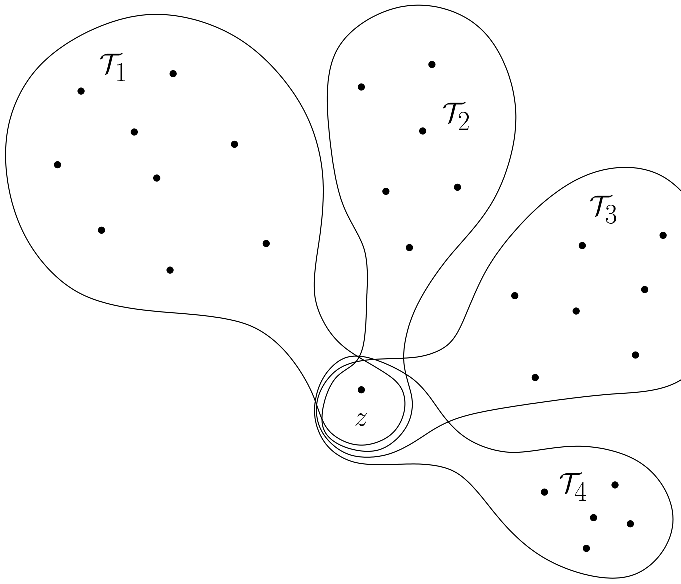



Figure 5.28: A Sample z-Partition

As with the case $n = 2$, the Mergeable Solutions Property is equivalent to the corresponding Mergeable Constraints Property.

**Definition 5.48 (Mergeable Constraints Property)** *Let $\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1)$, $\ldots$, $\mathcal{S}_n = (\mathcal{T}_n, \mathcal{C}_n)$ be consistent STNs such that $\mathcal{T}_1, \ldots, \mathcal{T}_n$ z-partition a set of time-points $\mathcal{T}$. We say that $\mathcal{S}_1, \ldots, \mathcal{S}_n$ have the* Mergeable Constraints Property *in the STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ if for any sets of constraints $\Delta\mathcal{C}_1, \ldots, \Delta\mathcal{C}_n$ over $\mathcal{T}_1, \ldots, \mathcal{T}_n$, respectively, if $\Delta\mathcal{C}_1, \ldots, \Delta\mathcal{C}_n$ are consistent with $\mathcal{S}_1, \ldots, \mathcal{S}_n$, respectively, then their union $\Delta\mathcal{C}_1 \cup \ldots \cup \Delta\mathcal{C}_n$ is consistent with $\mathcal{S}$.*

**Theorem 5.49 (MSP $\equiv$ MCP)** *Let $\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1)$, $\ldots$, $\mathcal{S}_n = (\mathcal{T}_n, \mathcal{C}_n)$ be consistent STNs such that $\mathcal{T}_1, \ldots, \mathcal{T}_n$ z-partition a set of time-points $\mathcal{T}$. $\mathcal{S}_1, \ldots, \mathcal{S}_n$ have the Mergeable Solutions Property in the STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ if and only if they have the Mergeable Constraints Property in $\mathcal{S}$.*

**Definition 5.50 (The General Temporal Decoupling Problem)** *Given an STN $\mathcal{S}$ whose time-points $\mathcal{T}$ are z-partitioned by $\mathcal{T}_1, \ldots, \mathcal{T}_n$, find sets of constraints $\mathcal{C}_1, \ldots, \mathcal{C}_n$ such that the STNs $\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1)$, $\ldots$, $\mathcal{S}_n = (\mathcal{T}_n, \mathcal{C}_n)$ temporally decouple $\mathcal{S}$.*

**Lemma 5.51** *Any instance of the General TDP in which $\mathcal{S}$ is consistent has a solution.*

### 5.4.2 Necessary and Sufficient Characterizations of Solutions to the General TDP

**Theorem 5.52 (Necessary Conditions)** *If the STNs $\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1)$, $\ldots$, $\mathcal{S}_n = (\mathcal{T}_n, \mathcal{C}_n)$ are a temporal decoupling of the STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$, then the following must hold:*

*(Property 1) For each r: $\mathcal{D}_r(x_i, x_j) \leq \mathcal{D}(x_i, x_j)$, for all $x_i, x_j \in \mathcal{T}_r$; and*

*(Property 2) For each $r, s$ such that $r \neq s$: $\mathcal{D}_r(x_i, z) + \mathcal{D}_s(z, x_j) \leq \mathcal{D}(x_i, x_j)$, for all $x_i \in \mathcal{T}_r$ and $x_j \in \mathcal{T}_s$,*

*where $\mathcal{D}_1, \ldots, \mathcal{D}_n$ and $\mathcal{D}$ are the respective distance matrices for $\mathcal{S}_1, \ldots, \mathcal{S}_n$ and $\mathcal{S}$. If, in addition to the above, Property 1 holds with equality in all instances (i.e., $\mathcal{S}_r \equiv \mathcal{S}_d|_{\mathcal{T}_r}$ for each $r$), then Property 2 also holds with equality in all instances.*

**Corollary 5.53** *If $\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1)$, $\ldots$, $\mathcal{S}_n = (\mathcal{T}_n, \mathcal{C}_n)$ temporally decouple $(\mathcal{T}, \mathcal{C})$, then $\mathcal{S}_1, \ldots, \mathcal{S}_n$ temporally decouple $(\mathcal{T}, \mathcal{C} \cup \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_n)$ such that Property 1 (and hence Property 2) from Theorem 5.52 holds with equality in all instances.*

**Theorem 5.54 (Sufficient Conditions)** *Let $\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1)$, $\ldots$, $\mathcal{S}_n = (\mathcal{T}_n, \mathcal{C}_n)$ and $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be consistent STNs such that $\mathcal{T}_1, \ldots, \mathcal{T}_n$ z-partition $\mathcal{T}$. If Properties 1 and 2 of Theorem 5.52 hold, then $\mathcal{S}_1, \ldots, \mathcal{S}_n$ temporally decouple $\mathcal{S}$.*

### 5.4.3 Toward a General TDP Algorithm

In this section, the notions of xy-pairs and xy-edges are generalized to the notions of mixed pairs and mixed edges, respectively. Since the focus of the new definitions remains on pairs and edges, most of the definitions, results and proofs from Sections 5.2.3 and 5.2.4 carry over to the more general setting simply by replacing instances of the terms “xy-pairs” and “xy-edges” with the terms “mixed pairs” and “mixed edges”, respectively.

**Definition 5.55 (Mixed Pairs, Mixed Edges)** *Let $\mathcal{T}_1, \ldots, \mathcal{T}_n$ and $\mathcal{T}$ be sets of time-points such that $\mathcal{T}_1, \ldots, \mathcal{T}_n$ z-partition $\mathcal{T}$. Let $t_i$ and $t_j$ be arbitrary time-points in $\mathcal{T}$. The pair $(t_i, t_j)$ is called a* mixed pair *(with respect to that z-partition) if for some $r \neq s$:*

$$t_i \in \mathcal{T}_r \quad \text{and} \quad t_j \in \mathcal{T}_s.$$

*If, in addition, neither $t_i$ nor $t_j$ is the zero time-point variable, then that pair is called a* proper *mixed pair. A constraint (or edge), $(t_j - t_i \leq \delta)$, is called a* mixed edge *if $(t_i, t_j)$ is a mixed pair. A mixed edge is called a* proper *mixed edge if the corresponding pair is a proper, mixed pair.*

Definition 5.14 (Zero-Path Shortfall), Lemma 5.15 (ZPS $\geq 0$), Definition 5.16 (Dominated by a path through zero), Lemma 5.17 (ZPS values cannot increase), and Lemma 5.18 (sufficient to consider only tight, proper xy-edges) from Section 5.2.3 all carry over to the more general setting simply by replacing instances of the terms “xy-pairs” and “xy-edges” with the terms “mixed pairs” and “mixed edges”, respectively. (This holds for the proofs, too.) Thus, those definitions and results are not repeated here.

### 5.4.4 Algorithms for Solving the General TDP

Pseudo-code for the basic TDP algorithm (general case) is given in Figure 5.29. The algorithm returns sets of constraints $\mathcal{C}_1, \ldots, \mathcal{C}_n$ such that the corresponding STNs $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are d-subnetworks of $\mathcal{S}$ with respect to $\mathcal{T}_1, \ldots, \mathcal{T}_n$, respectively.

**Given:** An STN $\mathcal{S}$ whose time-points $\mathcal{T}$ are z-partitioned by the sets $\mathcal{T}_1, \ldots, \mathcal{T}_n$.

(1) Compute the distance matrix $\mathcal{D}$ for $\mathcal{S}$. If $\mathcal{S}$ is inconsistent, return `NIL` and halt; otherwise, initialize $\mathcal{E}$ to the set of tight, proper, mixed edges in $\mathcal{S}$, and continue.

(2) Select a tight, proper, mixed edge $E = (t_j - t_i \leq \delta)$ from $\mathcal{E}$ whose ZPS value is positive. (If, in the process, any edges in $\mathcal{E}$ are discovered that are no longer tight or that have a ZPS value of zero, remove those edges from $\mathcal{E}$.) If no such edges exist (i.e., if $\mathcal{E}$ has become empty), go to Step 6; otherwise, continue.

(3) Pick values $\delta_1$ and $\delta_2$ such that:

$$\begin{array}{rcccl} -\mathcal{D}(z, t_i) & \leq & \delta_1 & \leq & \mathcal{D}(t_i, z), \\ -\mathcal{D}(t_j, z) & \leq & \delta_2 & \leq & \mathcal{D}(z, t_j), \quad \text{and} \\ \delta & \leq & \delta_1 + \delta_2 & < & \mathcal{D}(t_i, z) + \mathcal{D}(z, t_j). \end{array}$$

(4) Add the constraints $E_1$: $z - t_i \leq \delta_1$ and $E_2$: $t_j - z \leq \delta_2$ to the STN $\mathcal{S}$, updating $\mathcal{D}$ accordingly.

(5) Go to Step 2.

(6) Return $\mathcal{C}_1, \ldots, \mathcal{C}_n$ where for each $r$,

$$\mathcal{C}_r \;=\; \mathcal{C}_d|_{\mathcal{T}_r} \;=\; \{(t_j - t_i \leq \mathcal{D}(t_i, t_j)) : t_i, t_j \in \mathcal{T}_r\},$$

where $\mathcal{C}$ includes all constraints added in passes of Step 4, and $\mathcal{D}$ has been updated accordingly.

Figure 5.29: Pseudo-code for the basic TDP algorithm (general case)

The same terminological change ("xy-pairs/edges" $\mapsto$ "mixed pairs/edges") that enabled most of the definitions, results and proofs from Section 5.2.3 to carry over to Section 5.4.3 also enables the following results from Section 5.2.4 to carry over to this section:

- Theorem 5.19: adding the Step 4 constraints does not threaten the STN's consistency;
- Corollary 5.20: after adding the Step 4 constraints, $\delta_1$ and $\delta_2$ are the updated distance-matrix entries;
- Lemma 5.21: the regions $\Omega$ and $\Theta$ are related by invertible transformations;
- Corollary 5.22: it is always possible to choose values of $\delta_1$ and $\delta_2$ in Step 3 to achieve any legal $R$ and $\alpha$ values; and
- Proposition 5.25: any progress made by the algorithm is never lost.

The theorems stipulating the soundness and completeness of the TDP algorithm in the general setting are given below.

**Lemma 5.56 (Soundness)** *If the General TDP algorithm terminates at Step 6, then the constraint sets* $\mathcal{C}_1, \ldots, \mathcal{C}_n$ *returned by the algorithm are such that* $(\mathcal{T}_1, \mathcal{C}_1), \ldots, (\mathcal{T}_n, \mathcal{C}_n)$ *are a general temporal decoupling of the input STN* $\mathcal{S}$.

The Greedy and Less-Greedy strategies for ensuring that substantial progress is made during each iteration of the algorithm are unchanged. However, the maximum number of iterations are:

| | | |
|---|---|---|
| Greedy Strategy | At most $2\Pi_{i=1}^{n}\lvert\mathcal{T}_i\rvert$ | iterations |
| Less-Greedy Strategy | At most $2\Pi_{i=1}^{n}\lvert\mathcal{T}_i\rvert \left\lceil \left\lceil \dfrac{log(Z/\epsilon)}{log(1/(1-r))} \right\rceil + 1 \right\rceil$ | iterations |

**Corollary 5.57 (Sound and Complete)** *Using the Greedy Strategy (cf. Definition 5.24) or the Less-Greedy Strategy (cf. Definition 5.27), the General TDP algorithm is sound and complete.*

### 5.4.5 An Application of the General TDP Algorithm to the Post-Auction-Coordination Problem

As described in Chapter 3, when bids in a task-allocation auction are awarded, the temporal constraints from all of the bids are guaranteed to be consistent with the temporal constraints associated with the original opportunity; however, there is no requirement that the tasks being awarded to different agents be temporally decoupled. For example, if I have been awarded a bid concerning a dish-washing task constrained to occur in the interval $[0, 20]$ and you have been awarded a bid concerning the corresponding dish-drying task, similarly constrained to occur in the interval $[0, 20]$, but also constrained to occur after the dish-washing task, then our tasks are not independent. Figure 5.30 shows an STN representing our activities. Notice that the edge from $y$ to $x$ is not dominated by a path through zero. (For simplicity, the tasks are assumed to have zero duration.)

The General TDP algorithm may be directly applied in such situations to achieve a complete decoupling of the tasks being done by different agents. The advantages of such an approach include the following:

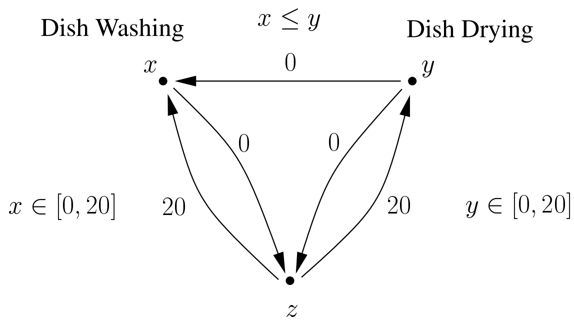



Figure 5.30: An STN representing dependent dish washing and drying activities

- The agents may henceforth operate independently, each confident that no matter what constraints it might choose to add to its local subnetwork, the global constraints will necessarily be satisfied (cf. the Mergeable Solutions Property, Definition 5.44).
- The General TDP algorithm is sound and complete, easy to implement, and runs in polynomial time.

Furthermore, an extension of the Iterative Weakening algorithm (cf. Section 5.2.6) to the case of multiple subnetworks can be applied to any decoupling to generate a *minimal* decoupling.

Despite these advantages, fully decoupling the tasks being done by different agents does require adding constraints to the network. In some cases, agents might prefer to accept some amount of dependence among tasks (and the requisite coordination overhead) in exchange for greater flexibility in their local subnetworks. The Relative Temporal Decoupling Problem described in the next section, formally addresses this issue.

## 5.5 The Relative Temporal Decoupling Problem

In certain applications (e.g., the Post-Auction-Coordination Problem), it can be advantageous to fully decouple a set of subnetworks that, taken together, do not constitute the entire network. For example, if agents $G_1$ and $G_2$ both must finish their tasks $\beta_1$ and $\beta_2$ prior to agent $G_3$ being able to start its task $\beta_3$, it may be advantageous to decouple $\beta_1$ and $\beta_2$ from one another, but not from $\beta_3$. Doing so would leave $G_1$ and $G_2$ able to act independently of one another, while leaving $G_3$ dependent on both of them, as illustrated in Figure 5.31. Such a decoupling is called a *Relative Temporal Decoupling*. In such a decoupling, the subnetworks being decoupled, taken together, do not constitute the entire network. The leftover time-points (i.e., those that do not belong to any of the subnetworks being decoupled) constitute what is herein called the *relativizing set*, denoted $\mathcal{T}_W$. In a relative temporal decoupling, the subnetworks are said to be fully decoupled *relative* to the set $\mathcal{T}_W$ of leftover time-points.

The Relative Temporal Decoupling Problem bears a superficial resemblance to a single iteration of the Allocative TDP algorithm in that each treats a set of subnetworks that, taken

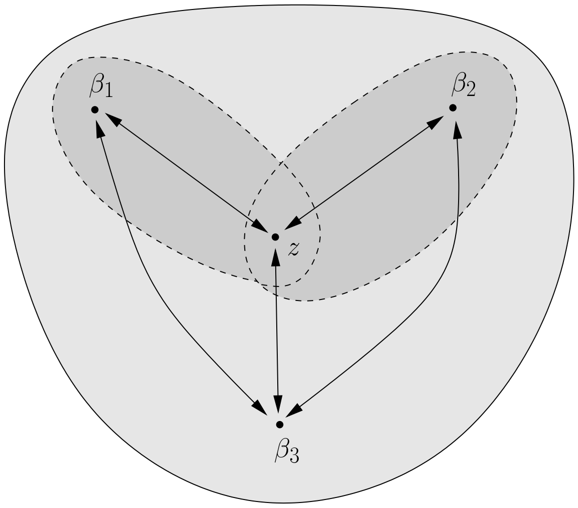



Figure 5.31: A sample relative temporal decoupling

together, do not constitute the entire network. However, after a single iteration of the Allocative TDP algorithm, the subnetworks are typically only *partially* decoupled from one another, whereas a solution to the Relative TDP requires the subnetworks to be *fully* decoupled from one another. Stated differently, a solution to the Relative TDP is required to have a Mergeable Solutions Property (relative to the set $\mathcal{T}_W$), whereas the partially decoupled subnetworks generated by a single iteration of the Allocative TDP algorithm typically have no such property.[5] In addition, the goal of the Allocative TDP algorithm is to incrementally construct a set of fully decoupled subnetworks that, taken together, *do* constitute the entire network, whereas the subnetworks generated by a solution to the Relative TDP, taken together, do *not* constitute the entire network. The set of leftover time-points in $\mathcal{T}_W$ remain "unallocated" in any solution to the Relative TDP.

Because the set of time-points in $\mathcal{T}_W$ are not decoupled from the other subnetworks, there is an inherent asymmetry in a relative temporal decoupling (which mirrors an inherent asymmetry present in many applications). In particular, the agents controlling the subnetworks that are fully decoupled from one another relative to $\mathcal{T}_W$ may act independently of one another—and independently of the agent $G_W$ controlling the time-points in the set $\mathcal{T}_W$; however, the agent $G_W$ does not have the same freedom. In particular, $G_W$ is not allowed to impose any additional constrainedness on any of the other subnetworks. As a result, $G_W$ is not free to blindly apply the ordinary STN rules for adding new constraints (e.g., as specified in Theorem 4.33). $G_W$ must also observe an additional set of bounds, called *lambda bounds*, which are explicated below, in Section 5.5.5.

[5] At the end of an intermediate iteration of the Allocative TDP algorithm, even though every tight, proper xy-edge *that currently exists* is dominated by a path through zero, there might yet be some shortest *path* from some $t_x \in \mathcal{T}_X$ to some $t_y \in \mathcal{T}_Y$ that passes through some of the points not yet allocated to $\mathcal{T}_X$ or $\mathcal{T}_Y$, but that is not dominated by a path through zero (cf. the path from $t_x$ to $t_u$ to $t_y$ in Figure 5.20). In such cases, the Mergeable Solutions Property does not hold.

In addition to its application to the Post-Auction-Coordination Problem, the tighter set of bounds that the agent $G_W$ must observe in a relative temporal decoupling provides a solution to the question of how an agent should accommodate the situation of having outstanding bids. In that case, the auctioneer, who is authorized to apply additional constraints to an agent's bid, plays the role of an independent actor, while the bidder plays the role of the agent $G_W$, who is in part dependent on the decisions of the auctioneer.

In contrast to the General TDP, where it only makes sense to consider decouplings involving two or more subnetworks (i.e., $n \geq 2$), for the Relative TDP, the presence of the relativizing set $\mathcal{T}_W$ makes the case $n = 1$ non-trivial. In fact, the case $n = 1$ is directly applicable to the Temporal-Constraint-Generation problem, as discussed in Section 5.5.6.

Finally, the analysis in this section reduces the problem of finding an algorithm for the Relative Temporal Decoupling Problem to the problem of finding a set of *redundant* constraints to add to the global network to trick the General TDP algorithm into doing the right thing.

The rest of this section is organized as follows. Section 5.5.1 formally defines the Relative TDP. Section 5.5.2 presents theorems characterizing solutions to instances of the Relative TDP. Sections 5.5.3 and 5.5.4 develop a sound and complete algorithm for solving the Relative TDP. Section 5.5.5 presents the *lambda bounds* relevant to the agent controlling the leftover time-points in $\mathcal{T}_W$. Section 5.5.6 shows how the relative TDP algorithm applies to the Temporal-Constraint-Generation problem. Section 5.5.7 shows how the relative TDP algorithm can be used to provide more flexible solutions to the Post-Auction-Coordination Problem.

Although the formal presentation in Sections 5.5.1 through 5.5.4 parallels the corresponding presentations for the TDP and the General TDP, the presence of the relativizing set $\mathcal{T}_W$ introduces additional complications. As a result, full proofs are provided for many of the theorems.

### 5.5.1 Formal Definition of the Relative TDP

We begin by defining a z-partition relative to a set $\mathcal{T}_W$ of time-points. Unlike the other sets involved in a relative z-partition, the relativizing set $\mathcal{T}_W$ does *not* contain $z$. Thus, $\mathcal{T}_W$ is given a distinguished place in the notation.

**Definition 5.58 (Relative z-Partition)** *Let $\mathcal{T}, \mathcal{T}_1, \ldots, \mathcal{T}_n$ and $\mathcal{T}_W$ be sets of time-point variables. A* z-partition *of $\mathcal{T}$* relative to *$\mathcal{T}_W$ is an ordered sequence $(\mathcal{T}_1, \ldots, \mathcal{T}_n;\ \mathcal{T}_W)$ such that the sets $\mathcal{T}_1, \ldots, \mathcal{T}_n$ z-partition the set $\mathcal{T} \setminus \mathcal{T}_W$. The sets $\mathcal{T}_1, \ldots, \mathcal{T}_n$ are said to z-partition $\mathcal{T}$ relative to $\mathcal{T}_W$.*

Figure 5.32 shows a z-partition involving the sets $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ and $\mathcal{T}_W$.

**Definition 5.59 (Extendible Partial Solution)** *Let $\mathcal{S}$ be an STN over the time-points in $\mathcal{T}$. Let $\mathcal{T}' \subseteq \mathcal{T}$ be a set of time-points. An* extendible partial solution *for $\mathcal{S}$* over *$\mathcal{T}'$ is a set of assignments for the time-point variables in $\mathcal{T}'$ that is extendible to a solution for $\mathcal{S}$. In the case where $\mathcal{T}_W = \mathcal{T} - \mathcal{T}'$, then $(\mathcal{T}'; \mathcal{T}_W)$ is a relative z-partition of $\mathcal{T}$ and the above set of*

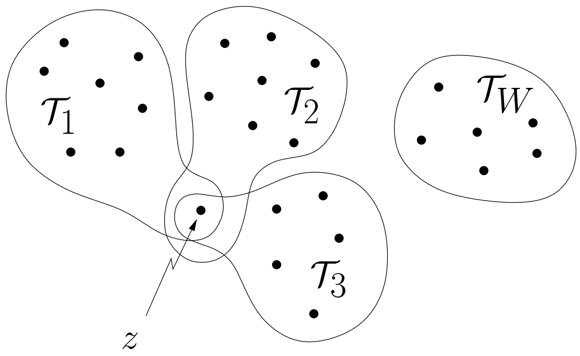



Figure 5.32: A sample relative z-partition

*variable assignments may be called a* partial solution for $\mathcal{S}$ relative to $\mathcal{T}_W$ *(i.e., highlighting the points* not *yet assigned values).*

**Proposition 5.60** *By Theorems 4.26 and 4.33, a set $\mathcal{X}|_{\mathcal{T}'}$ of assignments to time-point variables in $\mathcal{T}'$ is an extendible partial solution for $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ over $\mathcal{T}$ if and only if $\mathcal{X}|_{\mathcal{T}'}$ is locally consistent with respect to the constraints in $\mathcal{C}_d$.*

Notice that an extendible partial solution for $\mathcal{S}$ relative to the empty set is simply a solution for $\mathcal{S}$. Notice also that an extendible partial solution for $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ over $\mathcal{T}'$ is necessarily a locally consistent assignment with respect to $\mathcal{C}$ (cf. Definition 4.26), but that the converse need not hold.

**Definition 5.61 (Relative Temporal Decoupling)** $\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1), \ldots, \mathcal{S}_n = (\mathcal{T}_n, \mathcal{C}_n)$ *are said to* temporally decouple $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ relative to $\mathcal{T}_W$ *if:*

- $(\mathcal{T}_1, \ldots, \mathcal{T}_n; \mathcal{T}_W)$ *is a relative z-partition of* $\mathcal{T}$*;*
- $\mathcal{S}_1, \ldots, \mathcal{S}_n$ *are consistent STNs; and*
- *(Relative Mergeable Solutions Property) Any solutions $\mathcal{X}_1, \ldots, \mathcal{X}_n$ to $\mathcal{S}_1, \ldots, \mathcal{S}_n$, may be merged to form an extendible partial solution for $\mathcal{S}$ relative to $\mathcal{T}_W$.*

Notice that a temporal decoupling relative to the empty set is simply a temporal decoupling (cf. Definition 5.44).

**Lemma 5.62** *If $\mathcal{S}_1, \ldots, \mathcal{S}_n$ temporally decouple $\mathcal{S}$ relative to $\mathcal{T}_W$, then $\mathcal{S}$ is consistent.*

**Lemma 5.63** *If $\mathcal{S}_1, \ldots, \mathcal{S}_n$ temporally decouple $(\mathcal{T}, \mathcal{C} \cup \mathcal{C}')$ relative to $\mathcal{T}_W$, then $\mathcal{S}_1, \ldots, \mathcal{S}_n$ also temporally decouple $(\mathcal{T}, \mathcal{C})$ relative to $\mathcal{T}_W$.*

**Lemma 5.64** *If $\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1), \ldots, \mathcal{S}_n = (\mathcal{T}_n, \mathcal{C}_n)$ temporally decouple $(\mathcal{T}, \mathcal{C})$ relative to $\mathcal{T}_W$, then $\mathcal{S}_1, \ldots, \mathcal{S}_n$ also temporally decouple $(\mathcal{T}, \mathcal{C} \cup \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_n)$ relative to $\mathcal{T}_W$.*

**Definition 5.65 (Relative Mergeable Constraints Property)** *Given consistent STNs, $\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1)$, ..., $\mathcal{S}_n = (\mathcal{T}_n, \mathcal{C}_n)$, where $(\mathcal{T}_1, \ldots, \mathcal{T}_n; \mathcal{T}_W)$ is a relative z-partition of $\mathcal{T}$, then $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are said to have the Mergeable Constraints Property in the STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ relative to $\mathcal{T}_W$ if whenever sets of constraints $\Delta\mathcal{C}_1, \ldots, \Delta\mathcal{C}_n$ over the time-points in $\mathcal{T}_1, \ldots, \mathcal{T}_n$, respectively, are consistent with $\mathcal{S}_1, \ldots, \mathcal{S}_n$, respectively, their union $\Delta\mathcal{C}_1 \cup \ldots \cup \Delta\mathcal{C}_n$ is consistent with $\mathcal{S}$.*

**Theorem 5.66 (Relative MSP $\equiv$ Relative MCP)** *Let $\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1)$, ..., $\mathcal{S}_n = (\mathcal{T}_n, \mathcal{C}_n)$ be consistent STNs such that $(\mathcal{T}_1, \ldots, \mathcal{T}_n; \mathcal{T}_W)$ is a relative z-partition of $\mathcal{T}$. Then $\mathcal{S}_1, \ldots, \mathcal{S}_n$ have the Mergeable Solutions Property in $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ relative to $\mathcal{T}_W$ if and only if they have the Mergeable Constraints Property in $\mathcal{S}$ relative to $\mathcal{T}_W$.*

**Definition 5.67 (The Relative Temporal Decoupling Problem)** *Given an STN $\mathcal{S}$ whose time-points $\mathcal{T}$ are z-partitioned by $\mathcal{T}_1, \ldots, \mathcal{T}_n$ relative to $\mathcal{T}_W$, find sets of constraints $\mathcal{C}_1, \ldots, \mathcal{C}_n$ such that the STNs $\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1)$, ..., $\mathcal{S}_n = (\mathcal{T}_n, \mathcal{C}_n)$ temporally decouple $\mathcal{S}$ relative to $\mathcal{T}_W$.*

Notice that the Relative TDP in the case where $\mathcal{T}_W = \emptyset$ is simply the General TDP (cf. Definition 5.50).

**Lemma 5.68** *Any instance of the Relative TDP in which $\mathcal{S}$ is consistent has a solution.*

Although it is easy to construct solutions to the Relative TDP by employing constraints that *fix* all of the time-points in $\mathcal{T}_1, \ldots, \mathcal{T}_n$, it would be preferable to find more flexible solutions.

### 5.5.2 Necessary and Sufficient Characterizations of Solutions to the Relative TDP

**Theorem 5.69 (Necessary Conditions)** *If $\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1), \ldots, \mathcal{S}_n = (\mathcal{T}_n, \mathcal{C}_n)$ temporally decouple $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ relative to the set $\mathcal{T}_W$, then the following properties hold:*

*(Property 1) For each $r$: $\mathcal{D}_r(t_i, t_j) \leq \mathcal{D}(t_i, t_j)$, for each $t_i, t_j \in \mathcal{T}_r$; and*

*(Property 2) For each $r, s$ such that $r \neq s$: $\mathcal{D}_r(t_i, z) + \mathcal{D}_s(z, t_j) \leq \mathcal{D}(t_i, t_j)$,*
*for each $t_i \in \mathcal{T}_r$ and each $t_j \in \mathcal{T}_s$,*

*where (for any $r$ and $s$), $\mathcal{D}_r, \mathcal{D}_s$ and $\mathcal{D}$ are the distance matrices for the respective STNs $\mathcal{S}_r, \mathcal{S}_s$ and $\mathcal{S}$.*

**Proof**

(Property 1) Given any $r$ and any $t_i, t_j \in \mathcal{T}_r$, let $\mathcal{X}_r$ be a solution to $\mathcal{S}_r$ in which $t_j - t_i = \mathcal{D}_r(t_i, t_j)$ (cf. Theorem 4.33). For each $s \neq r$, let $\mathcal{X}_s$ be an arbitrary solution for $\mathcal{S}_s$. The Relative Mergeable Solutions Property ensures that the union of the solutions $\mathcal{X}_1, \ldots, \mathcal{X}_n$ is extendible to a solution for $\mathcal{S}$. Let $\mathcal{X}$ be such a solution. In that solution, $t_j - t_i = \mathcal{D}_r(t_i, t_j)$. In addition, being a solution for $\mathcal{S}$ implies that $t_j - t_i \leq \mathcal{D}(t_i, t_j)$. Thus, $\mathcal{D}_r(t_i, t_j) = t_j - t_i \leq \mathcal{D}(t_i, t_j)$.

(Property 2) Given any $r$ and $s$ with $r \neq s$, and any $t_i \in \mathcal{T}_r$ and $t_j \in \mathcal{T}_s$, let $\mathcal{X}_r$ and $\mathcal{X}_s$ be solutions to $\mathcal{S}_r$ and $\mathcal{S}_s$, respectively, in which $z - t_i = \mathcal{D}_r(t_i, z)$ and $t_j - z = \mathcal{D}_s(z, t_j)$. Let $\mathcal{X}$ be a solution for $\mathcal{S}$ extended from the merger of $\mathcal{X}_r$, $\mathcal{X}_s$, and arbitrary solutions for all $\mathcal{S}_q$, $q \notin \{r, s\}$. In the solution, $\mathcal{X}$:

$$t_j - t_i \;=\; (z - t_i) + (t_j - z) \;=\; \mathcal{D}_r(t_i, z) + \mathcal{D}_s(z, t_j).$$

In addition, being a solution for $\mathcal{S}$ implies that $t_j - t_i \leq \mathcal{D}(t_i, t_j)$. Thus,

$$\mathcal{D}_r(t_i, z) + \mathcal{D}_s(z, t_j) \leq \mathcal{D}(t_i, t_j). \;\blacksquare$$

**Theorem 5.70 (Sufficient Conditions)** *Let $\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1), \ldots, \mathcal{S}_n = (\mathcal{T}_n, \mathcal{C}_n)$ and $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be consistent STNs, and $\mathcal{T}_W$ a set of time-points such that $\mathcal{T}_1, \ldots, \mathcal{T}_n$ z-partition $\mathcal{T}$ relative to $\mathcal{T}_W$. If Properties 1 and 2 of Theorem 5.69 hold in all instances, then $\mathcal{S}_1, \ldots, \mathcal{S}_n$ temporally decouple $\mathcal{S}$ relative to $\mathcal{T}_W$.*

**Proof** Given the conditions of the theorem, it is sufficient to verify that the Relative Mergeable Solutions Property holds (i.e., that the union of arbitrary solutions for $\mathcal{S}_1, \ldots, \mathcal{S}_n$ is extendible to a solution for $\mathcal{S}$). Let $\mathcal{X}_1, \ldots, \mathcal{X}_n$ be arbitrary solutions for $\mathcal{S}_1, \ldots, \mathcal{S}_n$, respectively. Let $\mathcal{X} = \mathcal{X}_1 \cup \ldots \cup \mathcal{X}_n$. For any $r$ and $s$, and any $t_i \in \mathcal{T}_r$ and $t_j \in \mathcal{T}_s$:

- If $r = s$, then $t_j - t_i \leq \mathcal{D}_r(t_i, t_j)$ (since $\mathcal{X}_r$ is a solution for $\mathcal{S}_r$). Property 1 then gives that $t_j - t_i \leq \mathcal{D}(t_i, t_j)$.
- If $r \neq s$, then $t_j - t_i = (z - t_i) + (t_j - z) \leq \mathcal{D}_r(t_i, z) + \mathcal{D}_s(z, t_j)$ (since $\mathcal{X}_r$ and $\mathcal{X}_s$ are solutions for $\mathcal{S}_r$ and $\mathcal{S}_s$, respectively). Property 2 then gives that $t_j - t_i \leq \mathcal{D}(t_i, t_j)$.

Thus, for any $t_i, t_j$ drawn from the set $\mathcal{T}' = \mathcal{T}_1 \cup \ldots \cup \mathcal{T}_n$, we have that $t_j - t_i \leq \mathcal{D}(t_i, t_j)$. In other words, $\mathcal{X}$ is a locally consistent assignment of the variables in $\mathcal{T}'$ with respect to the constraints in $\mathcal{C}_d$. Thus, by Theorem 4.27, the set $\mathcal{X}$ is extendible to a solution for $\mathcal{S}$. Since $\mathcal{X}_1, \ldots, \mathcal{X}_n$ were arbitrary solutions for $\mathcal{S}_1, \ldots, \mathcal{S}_n$, the Relative Mergeable Solutions Property necessarily holds. $\blacksquare$

### 5.5.3 Toward a Relative TDP Algorithm

The presence of the relativizing set $\mathcal{T}_W$ makes a noticeable difference in the generalization of the notion of a mixed edge with respect to a z-partition (cf. Definition 5.55) to the notion of a mixed *path* with respect to a *relative* z-partition. Particularly important is that a relative mixed path joining some $t_i \in \mathcal{T}_r$ to some $t_j \in \mathcal{T}_s$ may contain time-points in the relativizing set $\mathcal{T}_W$. In fact, the only points allowed to be outside the set $\mathcal{T}_W$ are the endpoints of a mixed path, as illustrated in Figure 5.33.

**Definition 5.71 (Relative Mixed Path / Relative Mixed Pair)** *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be an STN. Let $\mathcal{T}_1, \ldots, \mathcal{T}_n$ and $\mathcal{T}_W$ be sets of time-points such that $(\mathcal{T}_1, \ldots, \mathcal{T}_n; \mathcal{T}_W)$ is a relative z-partition of $\mathcal{T}$. Let $r$ and $s$ be arbitrary such that $r \neq s$. Let $t_i \in \mathcal{T}_r$ and $t_j \in \mathcal{T}_s$ be arbitrary. A path $P$ of the form $(t_i, w_1, w_2, \ldots, w_k, t_j)$ is called a* mixed path *in $\mathcal{S}$ relative to $\mathcal{T}_W$ if: $k \geq 0$ and $\{w_1, w_2, \ldots, w_k\} \subseteq \mathcal{T}_W$.*

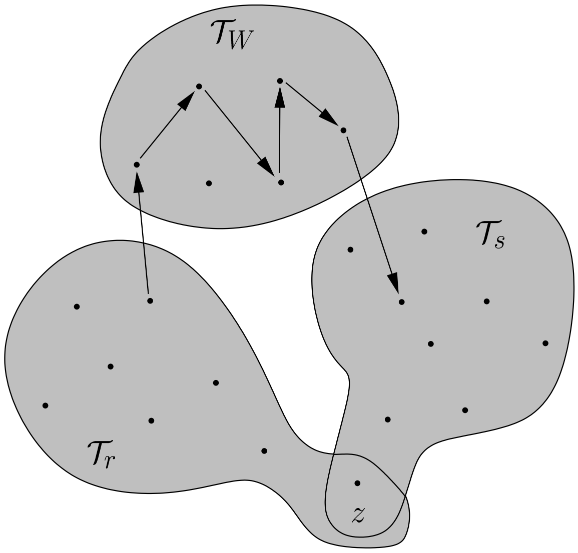



Figure 5.33: A sample *mixed path*

*If neither $t_i$ nor $t_j$ is the $z$ time-point, then $P$ is called a* proper, *mixed path in $\mathcal{S}$ relative to $\mathcal{T}_W$. (Since $z \notin \mathcal{T}_W$, $z$ cannot appear anywhere along a proper, mixed path relative to $\mathcal{T}_W$.) If, in addition, $P$ is a shortest path from $t_i$ to $t_j$ in $\mathcal{S}$, then $P$ is called a* tight, proper, mixed path *in $\mathcal{S}$ relative to $\mathcal{T}_W$. The endpoints, $(t_i, t_j)$, of a (proper) mixed path $P$ relative to $\mathcal{T}_W$ are called a (resp., proper) mixed pair relative to $\mathcal{T}_W$.*

Notice that if $k = 0$, then a mixed path corresponds to a mixed edge (cf. Definition 5.55).

**Note.** Although we speak of mixed paths relative to some set $\mathcal{T}_W$, it must be kept in mind that the definition of a mixed path depends on a specified relative z-partition (because the endpoints of each mixed path must belong to different sets $\mathcal{T}_r$ and $\mathcal{T}_s$ from the relative z-partition). Since this relative z-partition is usually clear from context, we typically omit the reference to it when speaking of mixed paths.

**Definition 5.72 (Relative Zero-Path Shortfall)** *The* zero-path shortfall *associated with a proper, mixed path $P$ from $t_i$ to $t_j$ is given by:*

$$\mathrm{ZPS}(P) \;=\; [\mathcal{D}(t_i, z) + \mathcal{D}(z, t_j)] - |P|.$$

**Definition 5.73 (Dominance by a Path through Zero)** *If the ZPS value for a tight, proper, mixed path $P$ relative to the set $\mathcal{T}_W$ is zero, then we say that $P$ is dominated by a path through zero.*

Lemma 5.74 shows that it suffices to consider only tight, proper, mixed paths when seeking a solution to an instance of the Relative TDP.

**Lemma 5.74** *Let* $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ *be a consistent STN and let* $(\mathcal{T}_1, \ldots, \mathcal{T}_n; \mathcal{T}_W)$ *be some relative z-partition of* $\mathcal{T}$ *that defines the mixed paths and mixed pairs in* $\mathcal{S}$*. If the inequality* $\mathcal{D}(t_i, z) + \mathcal{D}(z, t_j) \leq |P|$ *holds for every tight, proper, mixed* path $P$*, where* $t_i$ *and* $t_j$ *are the first and last time-points in* $P$*, respectively, then* $\mathcal{D}(t_p, z) + \mathcal{D}(z, t_q) \leq \mathcal{D}(t_p, t_q)$ *holds for every mixed* pair $(t_p, t_q)$ *in* $\mathcal{S}$.

**Proof** Let $(t_p, t_q)$ be an arbitrary mixed pair in $\mathcal{S}$ relative to $\mathcal{T}_W$ where $t_p \in \mathcal{T}_r$ and $t_q \in \mathcal{T}_s$. It suffices to show that $\mathcal{D}(t_p, z) + \mathcal{D}(z, t_q) \leq \mathcal{D}(t_p, t_q)$ holds.

Let $P = (v_0, v_1, \ldots, v_c, v_{c+1})$ be an arbitrary *shortest* path from $t_p$ to $t_q$ in $\mathcal{S}$. If $z$ is on this path, then the desired inequality holds (because the subpaths from $v_0$ to $z$ and from $z$ to $v_{c+1}$ must be shortest paths by Proposition 4.20). Suppose $z$ is not on $P$. Let $a$ be the largest index such that $v_a \in \mathcal{T}_x$ for some $x \neq s$. ($a$ is well-defined since $v_0 \in \mathcal{T}_r$ and $r \neq s$.) Similarly, let $b$ be the smallest index such that $a < b$, $v_b \in \mathcal{T}_y$ and $y \neq x$. ($b$ is well-defined since $a < c + 1$, $v_{c+1} \in \mathcal{T}_s$ and $s \neq x$.) Let $P'$ be the subpath of $P$ from $v_a$ to $v_b$. By construction, any time-points between $v_a$ and $v_b$ in $P$ must be in $\mathcal{T}_W$; thus, $P'$ is a mixed path relative to $\mathcal{T}_W$. Since $z$ is not on $P$ (by assumption), $P'$ is a proper, mixed path. Since $P'$ is a subpath of a shortest path, $P'$ must itself be a shortest path (by Proposition 4.20). Thus, $P'$ is a tight, proper, mixed path from $v_a$ to $v_b$ relative to $\mathcal{T}_W$. Thus, by the premise of the Lemma, $\mathcal{D}(v_a, z) + \mathcal{D}(z, v_b) \leq |P'|$. However, since $P'$ is a shortest path, the opposite inequality also holds. Thus, $\mathcal{D}(v_a, z) + \mathcal{D}(z, v_b) = |P'|$. But this implies that $P'$ may be replaced in $P$ by a pair of shortest paths, one from $v_a$ to $z$, one from $z$ to $v_b$, without changing the length of $P$. Since $z$ is on this version of the shortest path $P$ from $v_0$ to $v_{c+1}$ (i.e., from $t_p$ to $t_q$), the desired inequality holds (as argued in the earlier case). ■

### 5.5.4 Algorithms for Solving the Relative TDP

Theorem 5.75 shows how to use the General TDP algorithm to solve instances of the Relative TDP.

**Theorem 5.75** *Let* $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ *be a consistent STN whose time-points* $\mathcal{T}$ *are z-partitioned by* $\mathcal{T}_1, \ldots, \mathcal{T}_n$ *relative to* $\mathcal{T}_W$*. If to each tight, proper, mixed path* $P$ *(with respect to the specified relative z-partition) in* $\mathcal{S}$*, there corresponds an explicit constraint of the form* $t_j - t_i \leq |P|$ *in* $\mathcal{C}$*, (where* $t_i$ *and* $t_j$ *are the first and last time-points in* $P$*, respectively), then running the General TDP algorithm on* $\mathcal{S}$ *(where mixed edges are determined according to the sets* $\mathcal{T}_1, \ldots, \mathcal{T}_n$*, which do* not *form a z-partition of* $\mathcal{T}$*) will produce a* relative *temporal decoupling of* $\mathcal{S}$ *with respect to the relative z-partition* $(\mathcal{T}_1, \ldots, \mathcal{T}_n; \mathcal{T}_W)$.

**Proof** Step 1 of the General TDP algorithm (Figure 5.29) initializes $\mathcal{E}$ to the set of tight, proper, mixed edges in $\mathcal{S}$ whose ZPS values are positive. Crucially, the determination of what constitutes a mixed edge is made, in this theorem, with respect to the sets, $\mathcal{T}_1, \ldots, \mathcal{T}_n$, which do *not* form a z-partition of $\mathcal{T}$.[6] The algorithm then processes the edges in $\mathcal{E}$ until

[6]Because the set $\mathcal{E}$ is "improperly initialized", this use of the General TDP algorithm is non-standard and will typically *not* result in a solution to any General TDP—which, for one thing, requires having a specified z-partition. To generate a temporal decoupling with respect to the z-partition of $\mathcal{T}$ formed by the sets $\mathcal{T}_1, \ldots, \mathcal{T}_n$ and $(\mathcal{T}_W \cup \{z\})$, would require processing not only the tight, proper, mixed edges described in the theorem, but also any such edges having endpoints in $\mathcal{T}_W$, which are ignored in the theorem.

each is dominated by a path through zero. Using the Greedy or Less-Greedy Strategy (cf. Definitions 5.24 and 5.27), the algorithm is guaranteed to terminate at Step 6.

Suppose $P$ was some tight, proper, mixed path (with respect to the specified relative z-partition) from $t_i$ to $t_j$ in $\mathcal{S}$ prior to running the General TDP algorithm as described in the theorem. By assumption, there is an edge $E$ of the form $t_j - t_i \leq |P|$ in $\mathcal{S}$. Since $P$ is tight, so is the edge $E$. Furthermore, since $P$ is a proper, mixed path with respect to the specified relative z-partition, $E$ is a proper, mixed edge with respect to the sets $\mathcal{T}_1, \ldots, \mathcal{T}_n$. By the time the algorithm terminates, the edge $E$ is guaranteed to be dominated by a path through zero. Thus, so is the path $P$. Since $P$ was an arbitrary tight, proper, mixed path, the premise of Lemma 5.74 holds. Thus, the conclusion of the lemma also holds:

$$\mathcal{D}(t_p, z) + \mathcal{D}(z, t_q) \ \leq \ \mathcal{D}(t_p, t_q),$$

for every mixed *pair* $(t_p, t_q)$ with respect to the specified relative z-partition. Since the constraint sets $\mathcal{C}_1, \ldots, \mathcal{C}_n$ returned by the General TDP algorithm (cf. Figure 5.29) are such that the subnetworks $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are *d-subnetworks* of $\mathcal{S}$, Properties 1 and 2 of Theorem 5.69 follow (using an argument similar to that used in the proof of Theorem 5.23). Thus, by Theorem 5.70, the subnetworks $\mathcal{S}_1, \ldots, \mathcal{S}_n$ returned by the algorithm are a temporal decoupling of $\mathcal{S}$ relative to $\mathcal{T}_W$. ■

Theorem 5.75 reduces the problem of finding an algorithm for the Relative TDP to the problem of finding a set of *redundant* constraints to add to the STN to trick the General TDP algorithm into doing the right thing. The pseudo-code for the resulting Relative TDP algorithm is given in Figure 5.34. The algorithm uses a helper STN $\mathcal{S}_{Wx} = (\mathcal{T}, \mathcal{C}_{Wx})$, where

$$\mathcal{C}_{Wx} \ = \ \{(t_j - t_i \leq \delta) \in \mathcal{C} : t_j \neq z \text{ and } t_i \in \mathcal{T}_W\}.$$

Once constructed, the helper STN is fixed. The paths in the distance graph of the helper STN have the form: $(w_1, w_2, \ldots, w_k, t_j)$ where $k \geq 1$, each $w_i \in \mathcal{T}_W$, and $z \neq t_j \in \mathcal{T}$. For any $w \in \mathcal{T}_W$, and any $t_i \in \mathcal{T}$, where $t_i \neq z$, the length of the shortest mixed path (cf. Definition 5.71) from $t_i$ to $t_j$ having the form $(t_i, w, \ldots, t_j)$, if such exists, is given by: $\mathcal{D}(t_i, w) + \mathcal{D}_{Wx}(w, t_j)$. Except for the possibility of a tight *edge* from $t_i$ to $t_j$, any tight, proper, mixed path from $t_i$ to $t_j$ *must* have the form $(t_i, w, \ldots, t_j)$, for some $w$. Thus, the algorithm checks for tight edges and tight paths of the specified form.[7] Anytime it finds such an edge or path, it adds a triple of the form $\langle t_i, t_j, \mathcal{D}(t_i, t_j)\rangle$ to the set $\mathcal{E}$. This information is sufficient to generate the desired constraints needed to trick the General TDP algorithm into processing the tight, proper, mixed paths required to generate a solution to the Relative TDP.

### 5.5.5 Lambda Bounds

Given a relative temporal decoupling, the agents $G_1, \ldots, G_n$ controlling the decoupled subnetworks $\mathcal{S}_1, \ldots, \mathcal{S}_n$ may operate independently. In other words, as long as each agent

[7] It is only necessary to check $w \in \mathcal{T}_W$ for which there is a tight edge from $t_i$ to $w$. It is trivial to associate with each time-point the set of tight edges (or "successor edges") emanating from that time-point, referred to as $\mathit{TightSuccs}(t_i)$ in the pseudo-code for the algorithm.

**Given:** A consistent STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ whose time-points $\mathcal{T}$ are z-partitioned by $\mathcal{T}_1, \ldots, \mathcal{T}_n$ relative to $\mathcal{T}_W$.

(0) Set $\mathcal{E} = \emptyset$.

(1) Let $\mathcal{C}_{Wx} = \{(t_j - t_i \leq \delta) \in \mathcal{C} : t_j \neq z \text{ and } t_i \in \mathcal{T}_W\}$.

(2) Let $\mathcal{D}_{Wx}$ be the distance matrix for $(\mathcal{T}, \mathcal{C}_{Wx})$.

(3) `FOR` each proper, mixed pair $(t_p, t_q)$ (w.r.t. the relative z-partition specified above):

If there is a tight edge $E$ from $t_p$ to $t_q$ in $\mathcal{C}$, such that ZPS$(E) > 0$, then add $\langle t_p, t_q, \mathcal{D}(t_p, t_q)\rangle$ to $\mathcal{E}$;

Otherwise, `FOR` each $w \in (\mathcal{T}_W \cap \mathit{TightSuccs}(t_p))$:

If: $\mathcal{D}(t_p, w) + \mathcal{D}_{Wx}(w, t_q) \ = \ \mathcal{D}(t_p, t_q)$

and: $[\mathcal{D}(t_p, z) + \mathcal{D}(z, t_q)] \ > \ \mathcal{D}(t_p, t_q)$

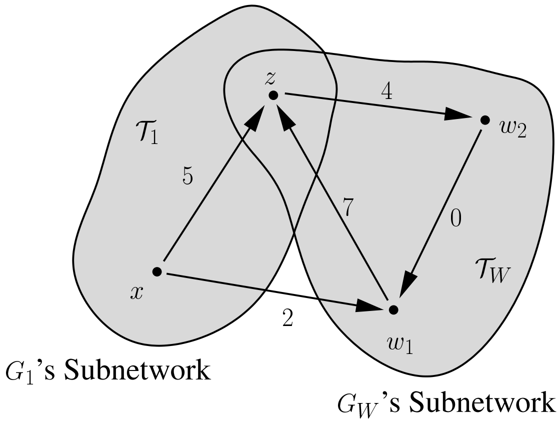
then: add $\langle t_p, t_q, \mathcal{D}(t_p, t_q)\rangle$ to $\mathcal{E}$

and break out of the inner `FOR` loop.

(4) Run Steps 2 through 6 of the General TDP Algorithm (cf. Figure 5.29).

**Return:** $\mathcal{C}_1 = \mathcal{C}_d|_{\mathcal{T}_1}, \ \ldots, \ \mathcal{C}_n = \mathcal{C}_d|_{\mathcal{T}_n}$ (generated by the General TDP Algorithm).

Figure 5.34: An algorithm for the Relative TDP based on the General TDP algorithm




Ordinary STN bound on edge from $w_1$ to $z$: $-4$

Lambda bound on edge from $w_1$ to $z$: $3$

Figure 5.35: An illustration of the use of lambda bounds

$G_i$ maintains the consistency of its own subnetwork $\mathcal{S}_i$, then each $G_i$ is free to further constrain $\mathcal{S}_i$ in any way whatsoever without having to worry that it might threaten the consistency of the global network $\mathcal{S}$.

Now consider the agent $G_W$ controlling the leftover time-points in $\mathcal{T}_W$. Since the time-points in $\mathcal{T}_W$ are *not* decoupled from $\mathcal{S}_1, \ldots, \mathcal{S}_n$, the agent $G_W$ may *not* operate independently of the subnetworks, $\mathcal{S}_1, \ldots, \mathcal{S}_n$. In particular, $G_W$ must refrain from adding any constraints that would impose any amount of additional constrainedness on $\mathcal{S}_1, \ldots, \mathcal{S}_n$. However, this does *not* imply that $G_W$ must refrain altogether from adding constraints to the network.

Suppose $G_W$ wants to add a constraint of the form, $t_j - t_i \leq \delta$, to the network. If $G_W$ were in control of the entire network, then according to Theorem 4.33, $G_W$ could add such a constraint if and only if $\delta \geq -\mathcal{D}(t_j, t_i)$. In other words, the negative-transpose entry in the distance matrix $\mathcal{D}$ for the global network would provide the lower bound for the length of the edge that $G_W$ wants to add to the network (where shorter edges correspond to stronger constraints).

In the context of a relative temporal decoupling in which $G_W$ controls the leftover time-points in $\mathcal{T}_W$, $G_W$ must still observe the lower bound specified by the corresponding negative-transpose distance-matrix entry. However, because $G_W$ must refrain from imposing any amount of additional constrainedness on the subnetworks $\mathcal{S}_1, \ldots, \mathcal{S}_n$, $G_W$ must also observe an extra set of lower bounds which, in this section, are called *lambda bounds*.

Figure 5.35 illustrates the use of lambda bounds by the agent $G_W$. $G_W$ is considering strengthening the edge from $w_1$ to $z$. If $G_W$ controlled the entire network, then $G_W$ could strengthen this edge by adding the constraint, $z - w_1 \leq -4$ (since $\mathcal{D}(z, w_1) = 4$). In other words, as long as $G_W$ refrains from introducing any loops with negative path-length, all is well. However, in the context of a relative temporal decoupling in which

$G_W$ must refrain from imposing any additional constrainedness on the subnetwork $\mathcal{S}_1$, $G_W$ cannot add the constraint $z - w_1 \leq -4$ since it would contribute to a shorter path from $x$ to $z$. In particular, it would introduce a path from $x$ to $w_1$ to $z$ of length $-2$, thus imposing additional constrainedness on the subnetwork $\mathcal{S}_1$. This does not mean that $G_W$ must refrain altogether from strengthening the edge from $w_1$ to $z$, only that $G_W$ must ensure that strengthening that edge does not impose on $\mathcal{S}_1$. In this case, $G_W$ may add the constraint $z - w_1 \leq 3$ without imposing on the subnetwork $\mathcal{S}_1$.

This section presents theorems specifying necessary and sufficient lower bounds, called *lambda bounds*, that the agent $G_W$ must observe to avoid imposing any amount of additional constrainedness on the subnetworks $\mathcal{S}_1, \ldots, \mathcal{S}_n$.

Unlike the agents $G_1, \ldots, G_n$, that are only allowed to add constraints among time-points within their own subnetworks, $G_W$ is allowed not only to add constraints among time-points in $\mathcal{T}_W$, but also to add constraints that connect a time-point in $\mathcal{T}_W$ to a time-point in one of the subnetworks. Doing so will not impose any amount of additional constrainedness on the subnetworks $\mathcal{S}_1, \ldots, \mathcal{S}_n$ as long as the lambda bounds are obeyed.

It is important to keep in mind that the lambda bounds do not represent constraints in the network; instead, they represent restrictions on which constraints $G_W$ may subsequently *add* to the network. In addition, whenever any agent $G_i$ adds a constraint to its subnetwork $\mathcal{S}_i$, the corresponding lambda bounds for $G_W$ are thereby *loosened*. In effect, when $G_i$ adds a constraint to its subnetwork, it is restricting its own independence, thus making it easier for $G_W$ to accommodate $G_i$'s independence.

As will be discussed in Sections 5.5.6 and 5.5.7, the lambda bounds have natural interpretations when the Relative TDP is applied to the Post-Auction-Coordination and Temporal-Constraint-Generation problems that were outlined in Chapter 3. For example, when an agent has an outstanding bid, the lambda bounds specify which constraints the agent can add to its private schedule without threatening its ability to honor its bid should it ultimately be awarded. In that case, the bidder is represented by $G_W$, the auctioneer by $G_1$.

### Finding the Lambda Bounds

If an agent controls all the time-points in an STN $\mathcal{S}$ and is concerned solely with maintaining the consistency of $\mathcal{S}$, then that agent may add any constraint of the form, $(t_j - t_i \leq \delta)$, where $\delta \geq -\mathcal{D}(t_j, t_i)$ (cf. Theorem 4.33). In other words, when contemplating strengthening an edge from $t_i$ to $t_j$, the negative-transpose distance-matrix entry, $-\mathcal{D}(t_j, t_i)$, provides a necessary, sufficient, and readily available lower bound for $\delta$.

In the context of a relative temporal decoupling, the agent $G_W$ cannot rely solely on the negative-transpose distance-matrix entries to determine which constraints it may safely add to the network. It is not sufficient that the STN remain consistent; $G_W$ must not add any constraints that would, directly or indirectly, add any additional amount of constrainedness to the subnetworks $\mathcal{S}_1, \ldots, \mathcal{S}_n$.

Theorem 5.76 shows that a characteristic set of inequalities hold in a consistent STN relative to the set $\mathcal{T}_W$, whether or not any temporal decoupling exists. These inequalities then motivate the definition of the lambda bounds (cf. Definition 5.77), which are computable in polynomial time. Theorems 5.79 and 5.80 verify that the lambda bounds are necessary and sufficient to ensure that $G_W$ does not impose any additional constrainedness on $\mathcal{S}_1, \ldots, \mathcal{S}_n$. Theorem 5.80 shows that if $G_W$ observes its lambda bounds, then the relative mergeable constraints property characterizing the relative temporal decoupling will necessarily be maintained.

**Theorem 5.76** *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be a consistent STN. Let $(\mathcal{T}'; \mathcal{T}_W)$ be a relative z-partition of $\mathcal{T}$ (i.e., $\mathcal{T} = \mathcal{T}' \cup \mathcal{T}_W$, $z \in \mathcal{T}'$, and $\mathcal{T}' \cap \mathcal{T}_W = \emptyset$). Let $\mathcal{C}_{xW}$ and $\mathcal{C}_{Wx}$ be given by:*

$$\begin{aligned} \mathcal{C}_{xW} &= \{(t_j - t_i \leq \delta) \in \mathcal{C} : t_j \in \mathcal{T}_W\} \\ \mathcal{C}_{Wx} &= \{(t_j - t_i \leq \delta) \in \mathcal{C} : t_i \in \mathcal{T}_W\}. \end{aligned}$$

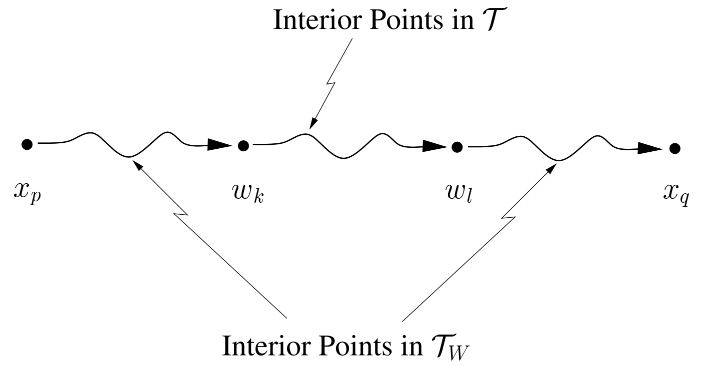
*Let $\mathcal{D}_{xW}$ and $\mathcal{D}_{Wx}$ be the distance matrices for the respective STNs, $\mathcal{S}_{xW} = (\mathcal{T}, \mathcal{C}_{xW})$ and $\mathcal{S}_{Wx} = (\mathcal{T}, \mathcal{C}_{Wx})$. Then the following inequalities hold for all $w_k, w_l \in \mathcal{T}_W$ and $x_p, x_q \in \mathcal{T}'$:*

$$\begin{array}{lrcl} (1) & \mathcal{D}(w_k, w_l) & \geq & [\mathcal{D}(x_p, x_q) - \mathcal{D}_{xW}(x_p, w_k) - \mathcal{D}_{Wx}(w_l, x_q)] \\ (2) & \mathcal{D}(x_p, w_l) & \geq & [\mathcal{D}(x_p, x_q) - \mathcal{D}_{Wx}(w_l, x_q)] \\ (3) & \mathcal{D}(w_k, x_q) & \geq & [\mathcal{D}(x_p, x_q) - \mathcal{D}_{xW}(x_p, w_k)] \end{array}$$

**Proof** First, notice that the paths in $\mathcal{S}_{xW}$ are precisely those paths in $\mathcal{S}$ having the form $(x, w_1, w_2, \ldots, w_k)$ or $(w_1, w_2, \ldots, w_k)$, where $x \in \mathcal{T}$ and $w_1, w_2, \ldots, w_k \in \mathcal{T}_W$. Similarly, the paths in $\mathcal{S}_{Wx}$ are precisely those paths in $\mathcal{S}$ having the form $(w_1, w_2, \ldots, w_k, x)$ or $(w_1, w_2, \ldots, w_k)$.

Let $w_k, w_l \in \mathcal{T}_W$ and $x_p, x_q \in \mathcal{T}'$ be arbitrary.

(1) Let $\mathcal{P}_1$ be the set of paths in $\mathcal{S}$ having the form:




where the interior points of the subpaths $x_p \rightsquigarrow w_k$ and $w_l \rightsquigarrow x_q$ are drawn exclusively from $\mathcal{T}_W$. If there are no such paths, then at least one of $\mathcal{D}_{xW}(x_p, w_k), \mathcal{D}(w_k, w_l)$ or $\mathcal{D}_{Wx}(w_l, x_q)$ is equal to positive infinity, in which case Inequality (1) necessarily holds. Otherwise, let $P_1$ be a path in $\mathcal{P}_1$ with minimal length, in which case:

$$|P_1| = \mathcal{D}_{xW}(x_p, w_k) + \mathcal{D}(w_k, w_l) + \mathcal{D}_{Wx}(w_l, x_q).$$

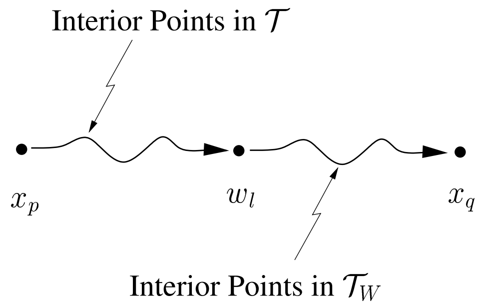
Since $P_1$ is a path from $x_p$ to $x_q$, it must be that $\mathcal{D}(x_p, x_q) \leq |P_1|$. Thus:

$$\mathcal{D}(x_p, x_q) \;\leq\; \mathcal{D}_{xW}(x_p, w_k) + \mathcal{D}(w_k, w_l) + \mathcal{D}_{Wx}(w_l, x_q).$$

Hence:

$$\mathcal{D}(w_k, w_l) \;\geq\; \mathcal{D}(x_p, x_q) - \mathcal{D}_{xW}(x_p, w_k) - \mathcal{D}_{Wx}(w_l, x_q).$$

(2) Let $\mathcal{P}_2$ be the set of paths in $\mathcal{S}$ having the form:




If there are no such paths, then $\mathcal{D}(x_p, w_l)$ or $\mathcal{D}_{Wx}(w_l, x_q)$ is equal to positive infinity, in which case Inequality (2) necessarily holds. Otherwise, let $P_2$ be a path in $\mathcal{P}_2$ with minimal length, in which case:

$$|P_2| \;=\; \mathcal{D}(x_p, w_l) + \mathcal{D}_{Wx}(w_l, x_q).$$

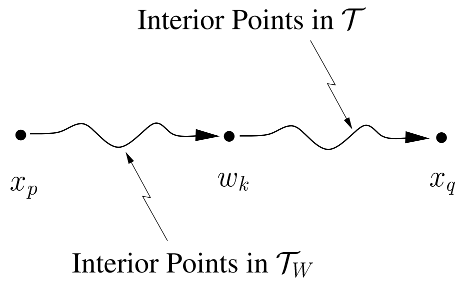
Since $P_2$ is a path from $x_p$ to $x_q$, it must be that $\mathcal{D}(x_p, x_q) \leq |P_2|$. Thus:

$$\mathcal{D}(x_p, x_q) \;\leq\; \mathcal{D}(x_p, w_l) + \mathcal{D}_{Wx}(w_l, x_q).$$

Hence:

$$\mathcal{D}(x_p, w_l) \;\geq\; \mathcal{D}(x_p, x_q) - \mathcal{D}_{Wx}(w_l, x_q).$$

(3) Let $\mathcal{P}_3$ be the set of paths in $\mathcal{S}$ having the form:




If there are no such paths, then $\mathcal{D}_{xW}(x_p, w_k)$ or $\mathcal{D}(w_k, x_q)$ is equal to positive infinity, in which case Inequality (3) necessarily holds. Otherwise, let $P_3$ be a path in $\mathcal{P}_3$ with minimal length, in which case:

$$|P_3| \;=\; \mathcal{D}_{xW}(x_p, w_k) + \mathcal{D}(w_k, x_q).$$

Since $P_3$ is a path from $x_p$ to $x_q$, it must be that $\mathcal{D}(x_p, x_q) \le |P_3|$. Thus:

$$\mathcal{D}(x_p, x_q) \;\le\; \mathcal{D}_{xW}(x_p, w_k) + \mathcal{D}(w_k, x_q).$$

Hence:

$$\mathcal{D}(w_k, x_q) \;\ge\; \mathcal{D}(x_p, x_q) - \mathcal{D}_{xW}(x_p, w_k). \blacksquare$$

**Definition 5.77 (Lambda Bounds)** *Given the same setup as in Theorem 5.76, let $\lambda$ be defined as follows, where $w_k, w_l \in \mathcal{T}_W$ and $x_p, x_q \in \mathcal{T}'$ are arbitrary:*

$$\begin{array}{lllll}
(1) & \lambda(w_k, w_l) & = & \max\limits_{x_p, x_q \in \mathcal{T}'} & [\mathcal{D}(x_p, x_q) - \mathcal{D}_{xW}(x_p, w_k) - \mathcal{D}_{Wx}(w_l, x_q)] \\
(2) & \lambda(x_p, w_l) & = & \max\limits_{x_q \in \mathcal{T}'} & [\mathcal{D}(x_p, x_q) - \mathcal{D}_{Wx}(w_l, x_q)] \\
(3) & \lambda(w_k, x_q) & = & \max\limits_{x_p \in \mathcal{T}'} & [\mathcal{D}(x_p, x_q) - \mathcal{D}_{xW}(x_p, w_k)].
\end{array}$$

*A constraint $E$ of the form, $t_j - t_i \le \delta$, where $t_i \in \mathcal{T}_W$ or $t_j \in \mathcal{T}_W$, is said to satisfy the appropriate lambda bound for $\mathcal{S}$ relative to $\mathcal{T}_W$ if $\delta \ge \lambda(t_i, t_j)$.*

**Proposition 5.78** *Given the distance matrices $\mathcal{D}, \mathcal{D}_{xW}$ and $\mathcal{D}_{Wx}$, and assuming constant matrix lookup time, each $\lambda(w_k, w_l)$ may be computed in $O(|\mathcal{T}'|^2)$ time, while each $\lambda(x_p, w_l)$ and each $\lambda(w_k, x_q)$ may be computed in $O(|\mathcal{T}'|)$ time. Since there are $O(|\mathcal{T}_W|^2)$ different values to compute for the first type and $O(|\mathcal{T}'||\mathcal{T}_W|)$ different values to compute for each of the other types, the entire set of lambda bounds may be computed in $O(|\mathcal{T}_W|^2|\mathcal{T}'|^2)$ time.*

**Theorem 5.79** *Given the same setup as in Theorem 5.76, let $E$ be a constraint of the form, $t_j - t_i \le \delta$, where $t_i \in \mathcal{T}_W$ or $t_j \in \mathcal{T}_W$. Let $\mathcal{C}^+ = \mathcal{C} \cup \{E\}$ and $\mathcal{S}^+ = (\mathcal{T}, \mathcal{C}^+)$. (In other words, $\mathcal{S}^+$ is the STN resulting from adding $E$ to $\mathcal{S}$.) Under the assumption that $E$ is consistent with $\mathcal{S}$ (i.e., that $\mathcal{S}^+$ is consistent), the following are equivalent:*

- *$E$ satisfies the appropriate lambda bound for $\mathcal{S}$ relative to $\mathcal{T}_W$.*
- *$\mathcal{D}^+(x_p, x_q) = \mathcal{D}(x_p, x_q)$, for all $x_p, x_q \in \mathcal{T}'$.*

**Proof** Suppose $E$ has the form $w_l - w_k \le \delta$. (The other forms of $E$ are handled similarly.) Thus, for $E$, the appropriate lambda bound is:

$$\lambda(w_k, w_l) \;=\; \max_{x_p, x_q \in \mathcal{T}'} [\mathcal{D}(x_p, x_q) - \mathcal{D}_{xW}(x_p, w_k) - \mathcal{D}_{Wx}(w_l, x_q)].$$

($\Leftarrow$) Suppose $E$ fails to satisfy the above lambda bound. Then, for some $x_p, x_q \in \mathcal{T}'$,

$$\delta \;<\; \mathcal{D}(x_p, x_q) - \mathcal{D}_{xW}(x_p, w_k) - \mathcal{D}_{Wx}(w_l, x_q),$$

which implies that

$$\mathcal{D}_{xW}(x_p, w_k) + \delta + \mathcal{D}_{Wx}(w_l, x_q) \;<\; \mathcal{D}(x_p, x_q).$$

Since the left-hand side of this inequality measures the length of some path in $\mathcal{S}^+$ from $x_p$ to $x_q$ (via $w_k$ and $w_l$), it must be that:

$$\mathcal{D}^+(x_p, x_q) \;\leq\; \mathcal{D}_{xW}(x_p, w_k) + \delta + \mathcal{D}_{Wx}(w_l, x_q),$$

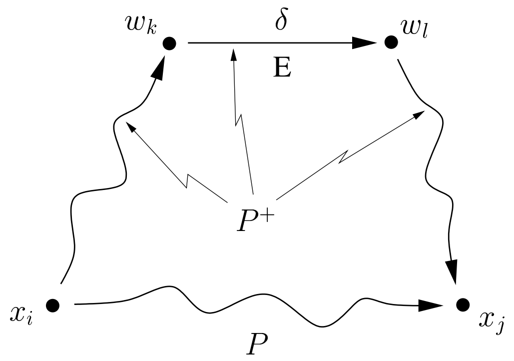
whence $\mathcal{D}^+(x_p, x_q) < \mathcal{D}(x_p, x_q)$. Thus, $\mathcal{D}^+(x_p, x_q) \neq \mathcal{D}(x_p, x_q)$.

($\Rightarrow$) Suppose $\mathcal{D}^+(x_i, x_j) \neq \mathcal{D}(x_i, x_j)$ for some $x_i, x_j \in \mathcal{T}'$. Since each constraint in $\mathcal{S}$ is present in $\mathcal{S}^+$, $\mathcal{D}^+(x_i, x_j) \leq \mathcal{D}(x_i, x_j)$. Thus, it must be that $\mathcal{D}^+(x_i, x_j) < \mathcal{D}(x_i, x_j)$.

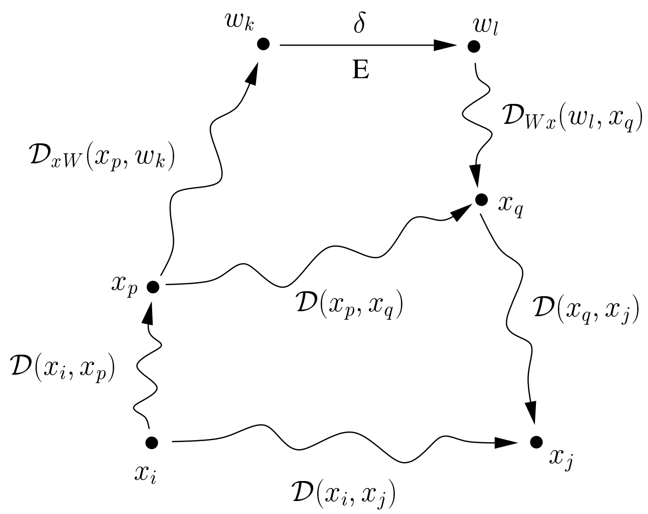
Let $P^+$ be a shortest path from $x_i$ to $x_j$ in $\mathcal{S}^+$, and let $P$ be a shortest path from $x_i$ to $x_j$ in $\mathcal{S}$. Thus,

$$|P^+| \;=\; \mathcal{D}^+(x_i, x_j) \;<\; \mathcal{D}(x_i, x_j) \;<\; |P|.$$

Since $\mathcal{S}$ and $\mathcal{S}^+$ differ only in that $\mathcal{S}^+$ contains $E$, it must be that $E$ is on the path $P^+$. Since removing loops from a shortest path does not change its length, we can assume that $E$ occurs only once in $P^+$. Thus, the situation is as shown below.




Now, let $x_p$ be the latest point in $P^+$ such that $x_p$ is before $w_k$, but $x_p \notin \mathcal{T}_W$. Similarly, let $x_q$ be the earliest point in $P^+$ such that $x_q$ is after $w_l$, but $x_q \notin \mathcal{T}_W$. Note that it might be that $x_i$ and $x_p$ are the same time-point variable; ditto for $x_j$ and $x_q$. The picture below illustrates the situation,



where the interior points of the subpaths from $x_p$ to $w_k$, and from $w_l$ to $x_q$, lie within $\mathcal{T}_W$ (by choice of $x_p$ and $x_q$); and where the shortest path in $\mathcal{S}$ from $x_p$ to $x_q$ is shown for later reference.

Now, since $|P^+| < \mathcal{D}(x_i, x_j)$, we have that:

$$\mathcal{D}(x_i, x_p) + \mathcal{D}_{xW}(x_p, w_k) + \delta + \mathcal{D}_{Wx}(w_l, x_q) + \mathcal{D}(x_q, x_j) \;=\; |P^+| \;<\; \mathcal{D}(x_i, x_j).$$

But we also have, by two applications of the Triangle Inequality to $\mathcal{D}(x_i, x_j)$, that:

$$\mathcal{D}(x_i, x_j) \;\leq\; \mathcal{D}(x_i, x_p) + \mathcal{D}(x_p, x_q) + \mathcal{D}(x_q, x_j).$$

Putting the above inequalities together and canceling like terms yields the following:

$$\mathcal{D}_{xW}(x_p, w_k) + \delta + \mathcal{D}_{Wx}(w_l, x_q) \;<\; \mathcal{D}(x_p, x_q),$$

whence:

$$\delta \;<\; \mathcal{D}(x_p, x_q) - \mathcal{D}_{xW}(x_p, w_k) - \mathcal{D}_{Wx}(w_l, x_q),$$

which implies that $E$ does not satisfy the appropriate lambda bound. ■

Theorem 5.80 shows that if a constraint $E$ is consistent with $\mathcal{S}$ and satisfies the appropriate lambda bound, then adding $E$ to $\mathcal{S}$ will not threaten the mergeable constraints property that characterizes a relative temporal decoupling.

**Theorem 5.80** *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$. Let $\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1), \ldots, \mathcal{S}_n = (\mathcal{T}_n, \mathcal{C}_n)$ be a temporal decoupling of $\mathcal{S}$ relative to $\mathcal{T}_W$. Let $E$ be a constraint of the form, $t_j - t_i \leq \delta$, where $t_i \in \mathcal{T}_W$ or $t_j \in \mathcal{T}_W$. Let $\mathcal{C}^+ = \mathcal{C} \cup \{E\}$ and $\mathcal{S}^+ = (\mathcal{T}, \mathcal{C}^+)$. (In other words, $\mathcal{S}^+$ is the STN resulting from adding $E$ to $\mathcal{S}$.) If $E$ is consistent with $\mathcal{S}$ and satisfies the appropriate $\lambda$ bound, then the STNs $\mathcal{S}_1, \ldots, \mathcal{S}_n$ temporally decouple $\mathcal{S}^+$ relative to $\mathcal{T}_W$. Furthermore, if Property 1 of Theorem 5.69 holds with equality in all instances for the temporal decoupling of $\mathcal{S}$, then it also does so for the temporal decoupling of $\mathcal{S}^+$.*

**Proof** To show that $\mathcal{S}_1, \ldots, \mathcal{S}_n$ temporally decouple $\mathcal{S}^+$ relative to $\mathcal{T}_W$, it suffices, by Theorem 5.70, to show that $\mathcal{S}^+$ is consistent and that Properties 1 and 2 from Theorem 5.69 hold. That $\mathcal{S}^+$ is consistent follows from $E$ being consistent with $\mathcal{S}$.

To show that Properties 1 and 2 hold, let $\mathcal{T}' = \mathcal{T}_1 \cup \ldots \cup \mathcal{T}_n$. By Theorem 5.79, since $E$ satisfies the appropriate lambda bound, $\mathcal{D}^+(x_p, x_q) = \mathcal{D}(x_p, x_q)$, for all $x_p, x_q \in \mathcal{T}'$. This in turn implies that Properties 1 and 2 hold with respect to $\mathcal{S}^+$ if and only if they hold with respect to $\mathcal{S}$. Furthermore, equality in Properties 1 and 2 holds in all instances for $\mathcal{S}^+$ if and only if it holds in all instances for $\mathcal{S}$. ■

### 5.5.6 The Temporal-Constraint-Generation Problem

This section addresses the Temporal-Constraint-Generation problem that an agent must solve when generating bids in the type of task-allocation auction described in Chapter 3. This section formally defines the TCG problem, presents necessary and sufficient characterizations of solutions to the TCG problem, and provides an algorithm that an agent can

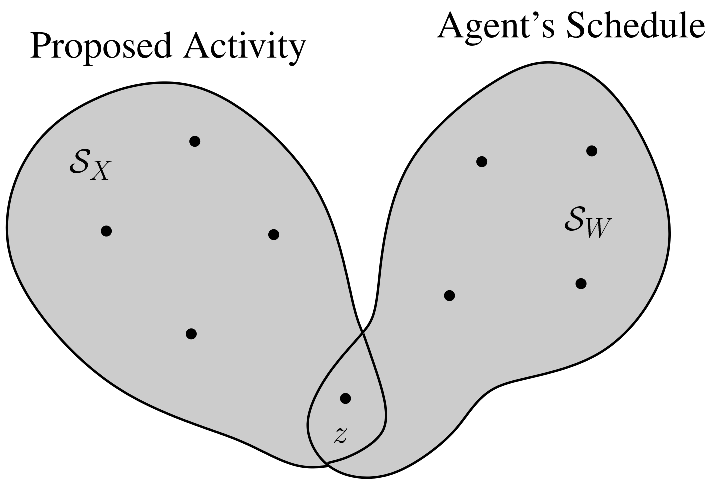



Figure 5.36: The pre-bidding situation for the TCG problem

use to solve instances of the TCG problem. The main result is proven by showing that the TCG problem is a special instance of the Relative TDP in which $n = 1$.

Let $\mathcal{S}_W = (\mathcal{T}_W \cup \{z\}, \mathcal{C}_W)$ be an STN representing the agent's private schedule of pre-existing commitments.[8] Let $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ be an STN representing the time-points and temporal constraints corresponding to a set of tasks in some proposed group activity. Since the activity is only in the proposal stage, it cannot share any time-points with the agent's pre-existing schedule of commitments. Thus, $\mathcal{T}_X \cap \mathcal{T}_W = \emptyset$. Notice, however, that $\mathcal{S}_X$ and $\mathcal{S}_W$ both include the zero time-point variable $z$. Figure 5.36 illustrates the pre-bidding situation.

Now suppose that the agent is considering submitting a bid on a set of tasks associated with the proposed group activity. Let $\mathcal{T}'_X \subseteq \mathcal{T}_X$ be the set of time-points associated with the tasks that the agent wants to bid on, as illustrated in Figure 5.37. To ensure that the tasks being bid on do not conflict with its schedule of pre-existing commitments, the agent must add some set $\mathcal{C}_{XW}$ of temporal constraints joining points in $\mathcal{T}_X$ and $\mathcal{T}_W$, as illustrated in Figure 5.38. Although in practice the edges in $\mathcal{C}_{XW}$ join points in $\mathcal{T}'_X$ and $\mathcal{T}_W$, the analysis below allows the constraints in $\mathcal{C}_{XW}$ to join points in $\mathcal{T}_X$ and $\mathcal{T}_W$. Let $\mathcal{S} = (\mathcal{T}, \mathcal{C}) = (\mathcal{T}_X \cup \mathcal{T}_W, \mathcal{C}_X \cup \mathcal{C}_W \cup \mathcal{C}_{XW})$ (i.e., the entire network represented in Figure 5.38).

**Note.** The process of generating the constraint set, $\mathcal{C}_{XW}$, is called the Task-Integration-Scheduling (TIS) problem (Hunsberger, 2002b). There are many possible algorithms for solving the TIS problem, but they are beyond the scope of this thesis. The TCG problem discussed in this section does not depend on the manner in which the constraint set $\mathcal{C}_{XW}$ is

[8]The set of time-points in $\mathcal{S}_W$ is written as $\mathcal{T}_W \cup \{z\}$ to facilitate the analogy with the corresponding instance of the Relative Temporal Decoupling Problem. The set $\mathcal{T}_W$ will be the set of leftover points for the relative temporal decoupling. The constraints in $\mathcal{C}_W$ may, of course, involve $\{z\}$.

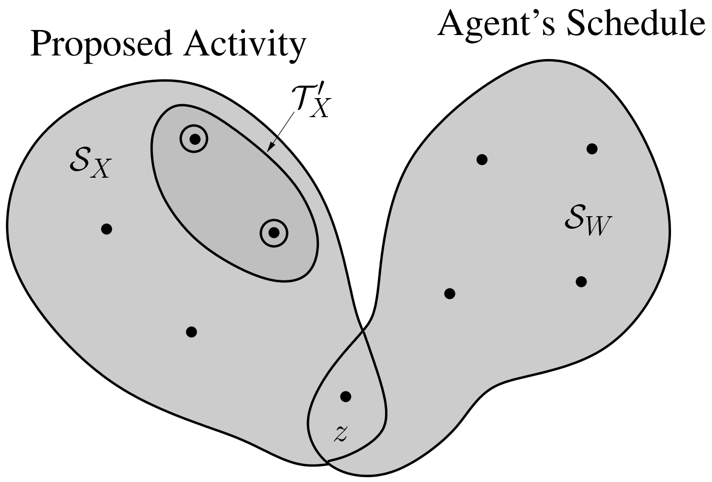



Figure 5.37: Highlighting the time-points associated with tasks the agent wants to bid on

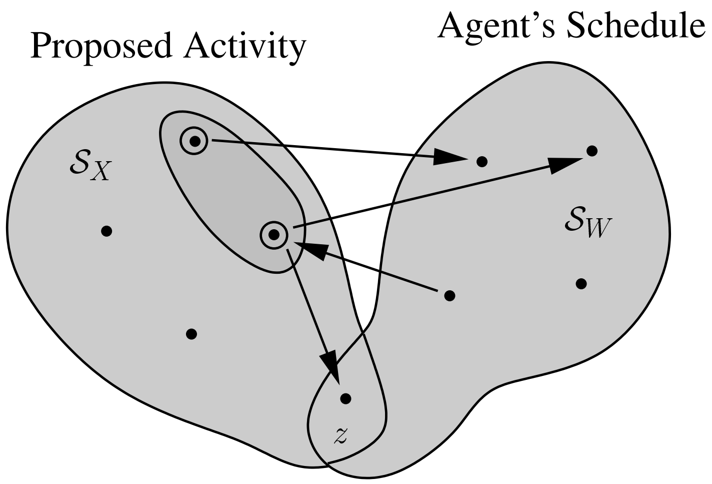



Figure 5.38: The set $\mathcal{C}_{XW}$ of temporal constraints for the TCG problem

generated.

As described in Chapter 3, the agent is allowed to include temporal constraints in its bid. However, the temporal constraints included with the bid can only refer to time-points associated with the proposed activity (i.e., time-points in the set $\mathcal{T}_X$). Thus, the agent is not allowed to simply make its bid conditioned on the entire contents of the STN $\mathcal{S}$.[9] The Temporal-Constraint-Generation problem (cf. Definition 5.81, below) is the problem of finding a set $\mathcal{C}_B$ of constraints to include with a bid such that no matter what additional constraints $\mathcal{C}''$ the auctioneer might add (in accordance with the rules of the auction), the agent's STN $\mathcal{S}$ will necessarily remain consistent.

The auctioneer is allowed to award any combination of bids that are mutually consistent and consistent with the constraints in the proposed activity. Thus, from the perspective of the bidding agent, the auctioneer is, in effect, allowed to add any set of constraints consistent with the constraints in $(\mathcal{C}_X \cup \mathcal{C}_B)$ (i.e., the constraints associated with the proposed activity and the constraints in the agent's bid).

**Definition 5.81 (The Temporal-Constraint-Generation Problem)** *Let $\mathcal{T}_X$ and $\mathcal{T}_W$ be sets of time-points such that $z \in \mathcal{T}_X$ and $\mathcal{T}_X \cap \mathcal{T}_W = \emptyset$. Let $\mathcal{C}_X$, $\mathcal{C}_W$ and $\mathcal{C}_{XW}$ be sets of constraints over the time-points in $\mathcal{T}_X$, $\{z\} \cup \mathcal{T}_W$, and $\mathcal{T}_X \cup \mathcal{T}_W$, respectively, such that $\mathcal{S} = (\mathcal{T}, \mathcal{C}) = (\mathcal{T}_X \cup \mathcal{T}_W, \mathcal{C}_X \cup \mathcal{C}_W \cup \mathcal{C}_{XW})$ is a consistent STN. Find a set of constraints $\mathcal{C}_B$ over $\mathcal{T}_X$ such that:*

*(1) $(\mathcal{T}_X, \mathcal{C}_X \cup \mathcal{C}_B)$ is consistent; and*

*(2) for any set of constraints $\mathcal{C}''$ over $\mathcal{T}_X$, if the constraints in $\mathcal{C}''$ are consistent with $(\mathcal{T}_X, \mathcal{C}_X \cup \mathcal{C}_B)$, then they are also consistent with $\mathcal{S}$.*

Notice that in Definition 5.81, $\mathcal{S}$ being consistent implies that both $\mathcal{S}_W = (\{z\} \cup \mathcal{T}_W, \mathcal{C}_W)$ and $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ are consistent. In addition, condition (1) is equivalent to saying that the constraints in $\mathcal{C}_B$ are consistent with $\mathcal{S}_X$.

**Theorem 5.82 (Necessary and Sufficient Conditions for TCG Solution)** *Given the same setup as in Definition 5.81, let $\mathcal{C}_B$ be a set of constraints over the time-points in $\mathcal{T}_X$ such that $\mathcal{S}_1 = (\mathcal{T}_X, \mathcal{C}_X \cup \mathcal{C}_B)$ is consistent. Then $\mathcal{C}_B$ is a solution to the TCG problem if and only if the constraints in $\mathcal{C}_X \cup \mathcal{C}_B$ entail the constraints in $\mathcal{C}_d|_{\mathcal{T}_X}$, where $\mathcal{C} = \mathcal{C}_X \cup \mathcal{C}_W \cup \mathcal{C}_{XW}$.*

**Proof** Let $\mathcal{D}_1$ be the distance matrix for $\mathcal{S}_1$.

($\Rightarrow$) Suppose $\mathcal{C}_B$ is a solution to the TCG problem. By Definition 5.61, the following imply that $\mathcal{S}_1$ is a temporal decoupling of $\mathcal{S} = (\mathcal{T}, \mathcal{C}) = (\mathcal{T}_X \cup \mathcal{T}_W, \mathcal{C}_X \cup \mathcal{C}_W \cup \mathcal{C}_{XW})$ relative to $\mathcal{T}_W$ in the case where $n = 1$.

- $\mathcal{S}_1$ and $\mathcal{S}$ are both consistent.

[9] This might be undesirable for other reasons as well. For example, the agent might not be *allowed* to reveal the contents of its private schedule. In addition, making the bid conditioned on its entire schedule could greatly restrict the agent's ability to take on additional commitments while the bid is outstanding.

- $(\mathcal{T}_X; \mathcal{T}_W)$ is a z-partition of $\mathcal{T} = (\mathcal{T}_X \cup \mathcal{T}_W)$, since $z \in \mathcal{T}_X$ and $\mathcal{T}_X \cap \mathcal{T}_W = \emptyset$.
- Since $\mathcal{C}_B$ is a solution to the TCG problem, Condition (2) of Definition 5.81 holds. Since Condition (2) is precisely the Relative Mergeable Constraints Property (cf. Definition 5.65 in the case, $n = 1$) which, by Theorem 5.66, is equivalent to the Relative Mergeable Solutions Property, the Relative MSP necessarily holds.

Since $\mathcal{S}_1$ temporally decouples $\mathcal{S}$ relative to $\mathcal{T}_W$, Property (1) of Theorem 5.69 holds:

$$\mathcal{D}_1(t_i, t_j) \leq \mathcal{D}(t_i, t_j), \text{ for all } t_i, t_j \in \mathcal{T}_X.$$

By Proposition 4.29, $\mathcal{D} = \mathcal{D}_d$. Furthermore, by Lemma 4.42, $(\mathcal{D}_d)|_{\mathcal{T}_X}$ is identical to the distance matrix, $\mathcal{D}_d|_{\mathcal{T}_X}$, for the d-subnetwork $\mathcal{S}_d|_{\mathcal{T}_X} = (\mathcal{T}_X, \mathcal{C}_d|_{\mathcal{T}_X})$. Thus,

$$\mathcal{D}_1(t_i, t_j) \leq \mathcal{D}_d|_{\mathcal{T}_X}(t_i, t_j), \text{ for all } t_i, t_j \in \mathcal{T}_X,$$

which, by Corollary 4.34, implies that the constraints in $\mathcal{C}_X \cup \mathcal{C}_B$ entail the constraints in $\mathcal{C}_d|_{\mathcal{T}_X}$.

($\Leftarrow$) Suppose the constraints in $\mathcal{C}_X \cup \mathcal{C}_B$ entail the constraints in $\mathcal{C}_d|_{\mathcal{T}_X}$. Reversing the last few steps of the above argument implies that Property 1 from Theorem 5.69 holds. Since Property 2 is vacuous in the case, $n = 1$, Theorem 5.70 gives us that $\mathcal{S}_1$ is a temporal decoupling of $\mathcal{S}$ relative to $\mathcal{T}_W$. Thus, the Mergeable Solutions Property holds relative to $\mathcal{T}_W$ (cf. Definition 5.61). Hence, the equivalent Mergeable Constraints Property holds relative to $\mathcal{T}_W$ which, as already argued, is equivalent to Condition (2) of Definition 5.81. Since $\mathcal{S}_1$ being consistent is Condition (1) of Definition 5.81, $\mathcal{C}_B$ is a solution to the TCG problem. ■

**Corollary 5.83** *If $\mathcal{C}_B$ is a solution to the TCG problem and $\Delta\mathcal{C}_B$ is a set of constraints over the time-points in $\mathcal{T}_X$ such that $\Delta\mathcal{C}_B$ is consistent with $\mathcal{C}_X \cup \mathcal{C}_B$, then $\mathcal{C}_B \cup \Delta\mathcal{C}_B$ is also a solution to the TCG problem.*

Theorem 5.82 implies that if $\mathcal{C}_B$ is a set of constraints over $\mathcal{T}_X$ such that $(\mathcal{C}_X \cup \mathcal{C}_B)$ is equivalent to $\mathcal{C}_d|_{\mathcal{T}_X}$, then $\mathcal{C}_B$ is a solution to the TCG problem representing the weakest constraints that are sufficient to protect the bidder's private schedule of pre-existing commitments against the possibility of the bid subsequently being awarded. Stated differently, such a solution places the weakest possible constraints on the auctioneer while still protecting the bidder's private schedule of pre-existing commitments.

### An Algorithm for Solving the TCG Problem

An algorithm for solving the TCG problem is given in Figure 5.39. The inputs to the algorithm are equivalent to the setup of Definition 5.81. In Step 1, the algorithm sets $\tilde{\mathcal{S}}$ to the d-subnetwork of $\mathcal{S}$ relative to $\mathcal{T}_X$. By Theorem 5.82, the constraints in $\mathcal{C}_d|_{\mathcal{T}_X}$ would provide a solution to the TCG problem, but one that contains $|\mathcal{T}_X|^2$ explicit constraints. The rest of the algorithm is devoted to generating an equivalent constraint set, $\mathcal{C}_B$, with fewer

**Given:**

- sets $\mathcal{T}_X$ and $\mathcal{T}_W$ of time-points such that $z \in \mathcal{T}_X$ and $\mathcal{T}_X \cap \mathcal{T}_W = \emptyset$; and
- sets $\mathcal{C}_X, \mathcal{C}_W$ and $\mathcal{C}_{XW}$ of constraints over the time-points in $\mathcal{T}_X$, $\{z\} \cup \mathcal{T}_W$, and $\mathcal{T}_X \cup \mathcal{T}_W$, such that $\mathcal{S} = (\mathcal{T}, \mathcal{C}) = (\mathcal{T}_X \cup \mathcal{T}_W, \mathcal{C}_X \cup \mathcal{C}_W \cup \mathcal{C}_{XW})$ is a consistent STN.

(1) Let $\tilde{\mathcal{S}} = (\mathcal{T}_X, \mathcal{C}_d|_{\mathcal{T}_X})$.

(2) Let $\tilde{\mathcal{S}}^{nrc} = (\tilde{\mathcal{T}}^{nrc}, \tilde{\mathcal{C}}^{nrc})$ and $\tilde{\mathcal{S}}^{rc}_1 = (\tilde{\mathcal{T}}^{rc}_1, \tilde{\mathcal{C}}^{rc}_1)$, $\ldots$, $\tilde{\mathcal{S}}^{rc}_n = (\tilde{\mathcal{T}}^{rc}_n, \tilde{\mathcal{C}}^{rc}_n)$ be the subnetworks resulting from decoupling the rigid components from $\tilde{\mathcal{S}}$, as described in Section 4.1 (cf. Definition 4.50). (If $\tilde{\mathcal{S}}$ has no rigid components, then $\tilde{\mathcal{S}}^{nrc} = \tilde{\mathcal{S}}$ and $n = 0$.)

(3) Let $\mathcal{C}_B = \{ (t_j - t_i \leq \delta) \in ((\tilde{\mathcal{C}}^{nrc})^u_d \cup \tilde{\mathcal{C}}^{rc}_1 \cup \ldots \cup \tilde{\mathcal{C}}^{rc}_n) : \delta < \mathcal{D}_X(t_i, t_j) \}$.

**Return:** $\mathcal{C}_B$.

Figure 5.39: An algorithm for solving the TCG problem

explicit constraints. Step 2 decouples any rigid components from the rest of $\tilde{\mathcal{S}}$, as described in Section 4.1. By Theorem 4.48, Lemma 4.49 and Theorem 4.53, the constraint set

$$\mathcal{C}^* = (\tilde{\mathcal{C}}^{nrc})^u_d \cup \tilde{\mathcal{C}}^{rc}_1 \cup \ldots \cup \tilde{\mathcal{C}}^{rc}_n$$

is equivalent to the constraints in $\tilde{\mathcal{S}}$, and furthermore, has the fewest number of edges in any constraint set equivalent to $\tilde{\mathcal{S}}$. Step 3 of the algorithm simply removes any constraints from $\mathcal{C}^*$ that are not strictly stronger than the corresponding strongest implicit constraints in $\mathcal{C}_X$.

### Adding Constraints in the Context of Outstanding Bids

In the case of an agent ($G_W$) having an outstanding bid in the sort of auction described in Chapter 3, the lambda bounds described in Section 5.5.5 specify which constraints the agent may add to its private schedule without threatening its ability to carry out the tasks in that bid should it ultimately be awarded. For example, suppose Bob has an outstanding bid to do a dish-washing activity between 3:00 and 6:00 p.m. Because the auctioneer (played by $G_i$) is, in effect, free to make Bob do the dishes any time between 3:00 and 6:00, Bob must refrain from committing to a house-painting activity scheduled to run from 4:30 to 5:30. (The lambda bounds preclude Bob from imposing any additional constrainedness on the dish-washing activity.) However, should the auctioneer subsequently inform Bob that the dish-washing activity will occur between 3:30 and 4:00, then Bob would thereby become free to commit to the house-painting activity.

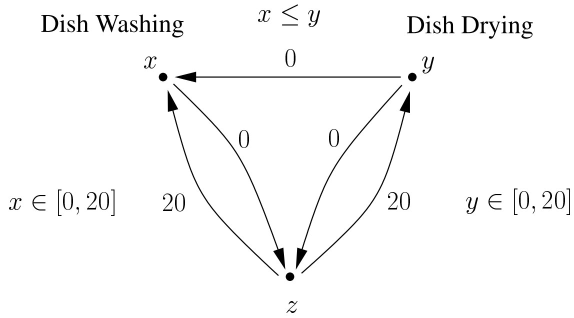



Figure 5.40: An STN representing dependent dish washing and drying activities

### 5.5.7 The Post-Auction Coordination Problem

One of the original motivations for analyzing the temporal decoupling problem was that agents working on a set of temporally dependent tasks must do something to ensure that the temporal constraints among tasks being done by different agents are satisfied. Three possible strategies they might follow are: (1) impose additional constraints on the network sufficient to decouple the tasks being done by different agents, thereby enabling agents to work independently, without the need for further coordination or communication; (2) decouple the tasks being done by *some* of the agents (so that they may work independently), while leaving the rest of the agents dependent on them; and (3) create a hierarchical decoupling in which each decoupled subnetwork is itself partitioned into decoupled subnetworks.

The first strategy can be realized by a direct application of the General TDP algorithm, as described in Section 5.4.5. By decoupling the tasks being done by all of the different agents, those agents are thereby free to operate independently, as long as each maintains the consistency of its local subnetwork. However, in some cases, fully decoupling the network may require the imposition of constraints beyond those that the agents are willing to accept.

For example, in the simple dish-washing/dish-drying scenario described in Section 5.4.5, and repeated in Figure 5.40, the tasks might be fully decoupled by adding such constraints as

$$x \leq 12 \qquad \text{and} \qquad y \geq 12;$$

however, it might instead be preferable to give the dish-washing agent $G_x$ complete freedom to select the time for its task, while making the dish-drying agent $G_y$ dependent on the choice made by $G_x$. Doing this would correspond to a *relative* temporal decoupling where

- $n = 1$;
- the subnetwork controlled by $G_x$ is: $\mathcal{S}_1 = (\{z, x\}, \{(x - z \leq 20), (z - x \leq 0)\})$;
- the set of leftover time-points controlled by $G_y$ is given by: $\mathcal{T}_W = \{y\}$.

While $G_y$ is waiting for $G_x$ to make up its mind about when it will wash the dishes, $G_y$ has less flexibility, and must use the lambda bounds described in Section 5.5.5 to decide which constraints it can safely add to the network. However, giving $G_x$ complete freedom

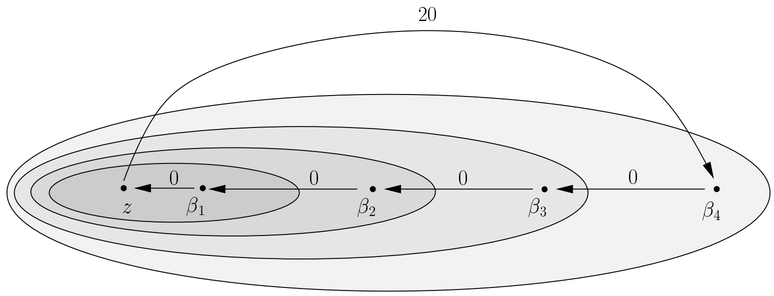



Figure 5.41: A hierarchical decoupling for a sequence of tasks

might end up giving $G_y$ more freedom, too. For example, should $G_x$ eventually decide to wash the dishes at time 4, then $G_y$ could decide to dry the dishes immediately thereafter instead of having to wait until time 12. When $G_x$ decides to set $x$ to 4, it causes the corresponding lambda bound for $G_y$ to *decrease*, as described in Section 5.5.5, resulting in greater flexibility for $G_y$. Furthermore, it may be that $G_x$ simply *requires* complete flexibility in making its dish-washing decision; for instance, it may not know when it will have access to a sink.

In general, the Relative TDP algorithm described in Section 5.5.4 enables a more flexible approach to solving the Post-Auction Coordination problem. In this approach, some agents are given greater freedom while others are required to be dependent on them. However, as seen in the above example, when the agents controlling the decoupled subnetworks further constrain their subnetworks (e.g., by deciding when to execute tasks under their control), it may translate into greater freedom of choice for the dependent agents as well, as evidenced by decreasing lambda bounds.

The third strategy calls for a structured, hierarchical approach to the allocation of temporal flexibility and dependence among the agents. To illustrate this strategy, consider the case of agents, $G_1, G_2, G_3$ and $G_4$, assigned to do tasks, $\beta_1, \beta_2, \beta_3$ and $\beta_4$, respectively, where the tasks are constrained to be executed consecutively within the interval $[0, 20]$. For simplicity, we suppose that the tasks have zero duration. In this case, it may be preferable to allow $G_1$ to have complete freedom to execute its task $\beta_1$ at any time within $[0, 20]$. Although $G_2$ would be dependent on $G_1$, once $G_1$ made up its mind to execute $\beta_1$ at some specified point, say, at time 3, then $G_2$ could act independently. Similarly, $G_3$ would be dependent on $G_2$ until $G_2$ made a decision to execute $\beta_2$, say, at time 7. Finally, $G_4$ would be dependent on $G_3$ until $G_3$ made a decision to execute $\beta_3$, say, at time 14—at which point, $G_4$ could act independently.

The simple hierarchy for the above example is illustrated in Figure 5.41. It can be generated by recursive applications of the Relative TDP algorithm, as follows. First, the subnetwork $\mathcal{S}'$ comprising the time-points $\{z, \beta_1, \beta_2, \beta_3\}$ is decoupled (in the case $n = 1$) relative to the set of leftover time-points $\{\beta_4\}$. Next, within the independent subnetwork $\mathcal{S}'$, the subnetwork $\mathcal{S}''$ comprising the time-points $\{z, \beta_1, \beta_2\}$ is decoupled (in the case $n = 1$)

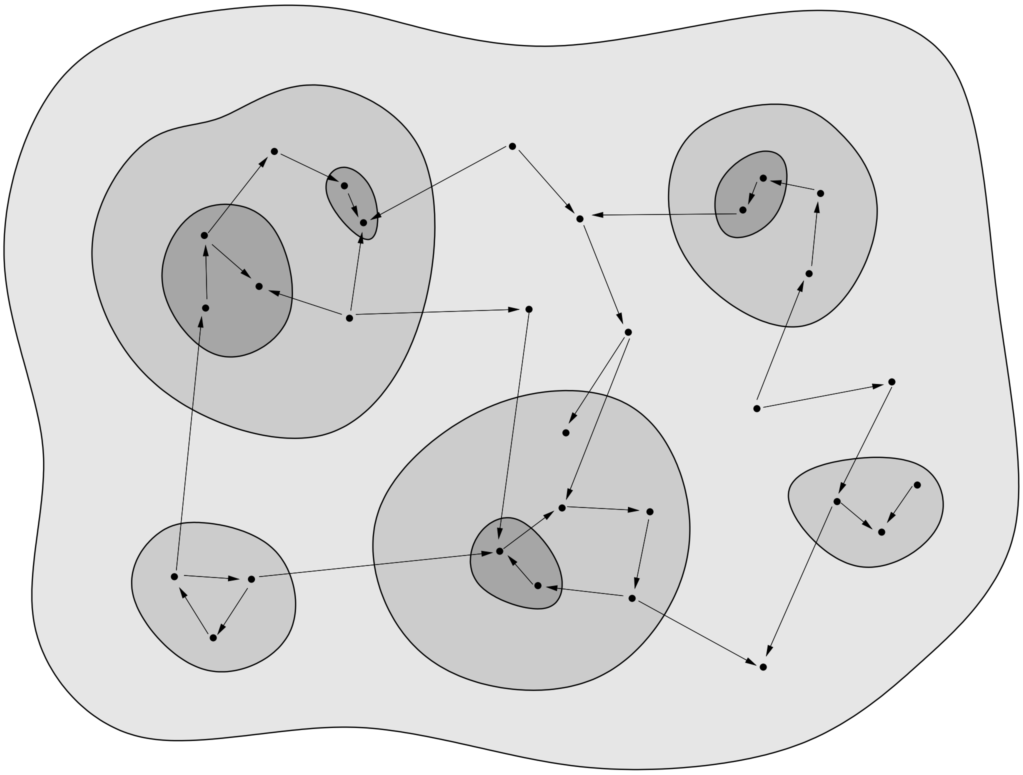


Figure 5.42: Sample Hierarchy of Partially Decoupled Subnetworks

relative to the set of leftover time-points $\{\beta_3\}$. Finally, within the independent subnetwork $\mathcal{S}''$, the subnetwork $\mathcal{S}'''$ comprising the time-points $\{z, \beta_1\}$ is decoupled (in the case $n = 1$) relative to the set of leftover time-points $\{\beta_2\}$.

Of course, relative temporal decouplings can be used to generate hierarchies with more complex structure, as illustrated in Figure 5.42. In general, given an STN $(\mathcal{T}, \mathcal{C})$, the procedure for generating a hierarchical decoupling is as follows.

- First, choose some relative z-partition $(\mathcal{T}_1, \ldots, \mathcal{T}_n; \mathcal{T}_W)$ of $\mathcal{T}$ and then use the Relative TDP algorithm to generate STNs $(\mathcal{T}_1, \mathcal{C}_1), \ldots, (\mathcal{T}_n, \mathcal{C}_n)$ that temporally decouple $(\mathcal{T}, \mathcal{C})$ relative to $\mathcal{T}_W$.

- Next, for each decoupled subnetwork $(\mathcal{T}_i, \mathcal{C}_i)$, choose some relative z-partition $(\mathcal{T}_{i1}, \ldots, \mathcal{T}_{in_i}; \mathcal{T}_{W_i})$ and then use the Relative TDP algorithm to generate STNs that temporally decouple $(\mathcal{T}_i, \mathcal{C}_i)$ relative to $\mathcal{T}_{W_i}$.

- Continue recursively down the hierarchy.

## 5.6 Related Work

### 5.6.1 Separation Vertices

The temporal decoupling problems presented in this chapter are related to *separation vertices*, as defined by Dechter et al. (1991).

**Definition 5.84** *(Dechter, Meiri, and Pearl, 1991) A connected graph $G = (V, E)$ is said to have a* separation vertex *$v$ (sometimes also called an articulation point) if there exist vertices $a$ and $b$, $a \neq v$ and $b \neq v$, such that all paths connecting $a$ and $b$ pass through $v$. In this case we also say that $v$ separates $a$ from $b$. A graph which has a separation vertex is called* separable*, and one which has none is called* non-separable*. Let $V' \subseteq V$. The induced subgraph $G' = (V', E')$ is called a* non-separable component *if $G'$ is non-separable and if for every larger $V''$, $V \subset V' \subseteq V''$, the induced subgraph $G'' = (V'', E'')$ is separable.*

If the STNs $\mathcal{S}_1, \ldots, \mathcal{S}_n$ temporally decouple $\mathcal{S}$ relative to some set $\mathcal{T}_W$, then, *disregarding edges dominated by paths through zero* (cf. Definition 5.16), $z$ is a separation vertex for every proper, mixed pair in $\mathcal{S}$ (cf. Definition 5.71). The time-points in $\mathcal{T}_W$ need not participate in any such separation relationships.

Whereas Dechter et al. focus on finding the non-separable components of a fixed temporal network, the temporal decoupling problems presented in this chapter focus on finding a set of constraints to *add* to a given network, with the goal of temporally decoupling the network into independent subnetworks, any of which may be separable.

### 5.6.2 Simple Temporal Networks with Uncertainty

When a single agent *controls* an STN (i.e., has the authority to tighten constraints among any pair of time-points in the network), it is easy for that agent to maintain the consistency of the network. For example, when adding a new constraint, $t_j - t_i \leq \delta$, to the network, the agent need only ensure that $\delta$ satisfies $\delta \geq -\mathcal{D}(t_i, t_j)$ (cf. Theorem 4.33).

When different agents control different portions of the network, agents may need to observe a different set of requirements when adding new constraints to the network. For example, as seen in Section 5.5.5, when the agent controlling the leftover set of time-points $\mathcal{T}_W$ in a relative temporal decoupling adds a new constraint to the network, it must observe the *lambda bounds* defined in that section to avoid imposing any additional amount of constrainedness on the decoupled subnetworks controlled by other agents.

Several researchers (Vidal and Fargier, 1999; Morris and Muscettola, 1999; Morris and Muscettola, 2000; Morris, Muscettola, and Vidal, 2001) have investigated temporal networks in single-agent settings in which some of the temporal differences, $t_j - t_i$, are controlled not by the agent, but by nature. The temporal differences controlled by nature are called *contingent durations*, each of which is used to represent some causal process that an agent might initiate but, once initiated, does not control. For example, I can control when my web browser begins the process of fetching baseball scores; and I know that on a good day it will take my browser no more than twenty seconds to fetch the scores; but I cannot control how long it will actually take.

Each contingent duration is constrained to lie within some interval of the form $[\delta_1, \delta_2]$, where $0 < \delta_1 < \delta_2 < \infty$. Thus, associated with each contingent duration is a pair of edges in the distance graph. The *forward edge* is an edge, $E_f$, of the form, $t_j - t_i \leq \delta_1$; the *reverse edge* is an edge, $E_r$, of the form, $t_i - t_j \leq -\delta_2$. The pair of edges, $E_f$ and $E_r$, are also referred to as a *contingent link*. It is assumed that each contingent link corresponds to an *independent* causal process.

An STN extended to include contingent durations is called a *Simple Temporal Network with Uncertainty (STNU)*.[10] If an agent must execute a set of tasks that are dependent on causal processes controlled by nature, as represented by an STNU, then the challenge for the agent is to select execution times for its tasks such that no matter what numbers nature assigns to the contingent durations, which cannot be known in advance, the network will remain consistent over time until all of the tasks are completed. To ensure global consistency, the agent must not impose any additional constrainedness on nature. If it is possible for an agent to respond successfully to this challenge, the network is called *controllable*.

**Strong, Dynamic and Weak Controllability of STNUs.** Different varieties of STNU controllability arise from different assumptions about when execution times are selected and when information about contingent durations becomes available to an agent. Vidal and Fargier (1999) define the following:

- Strong Controllability: there exists a *single* set of execution times for the agent's tasks that guarantees that the network will remain consistent no matter how nature sets the contingent durations;
- Dynamic Controllability: there is a dynamic strategy for selecting execution times for the agent's tasks in real time, as information about contingent durations becomes available, such that the network will remain consistent; and
- Weak Controllability: for each possible set of values that nature might select for the contingent durations, there is *some* set of execution times for the agent's tasks such that all temporal constraints will be satisfied.

Vidal and Fargier show that the above properties stand in the following relation:

Strong Controllability $\Rightarrow$ Dynamic Controllability $\Rightarrow$ Weak Controllability.

Interestingly, although checking whether a given network has the weak controllability property is Co-NP-complete (Morris and Muscettola, 1999), checking for the strong controllability property can be done in deterministic polynomial time (Vidal and Fargier, 1999), as can checking for the dynamic controllability property (Morris, Muscettola, and Vidal, 2001). Furthermore, Morris, Muscettola and Vidal (2001) provide an algorithm for efficiently *executing* a network that is dynamically controllable (i.e., for selecting execution values for time-point variables in real-time, as information about contingent durations becomes available, such that all constraints are satisfied).

[10] Vidal and Fargier (1999) define a *Simple Temporal Problem under Uncertainty (STPU)* that is essentially equivalent to an STNU.

**Waypoint Controllability.** Morris and Muscettola (1999) define a *waypoint controllability* property that generalizes both weak and strong controllability. An STNU is called waypoint controllable with respect to a distinguished set $W$ of time-points (called *waypoints*) "if there is a fixed assignment of time values to the [time-points] in $W$ that can be extended to a solution in every projection of [the STNU]." They show that the definition of waypoint controllability reduces to that of weak controllability in the case where the set of waypoints is an arbitrary singleton. Similarly, they show that the definition of waypoint controllability reduces to that of strong controllability in the case where the set of waypoints is the entire set of time-points except for those time-points that end a contingent link without starting a subsequent contingent link (e.g., in a chain of contingent links). The value of having a waypoint controllable network is that it "may be effectively decomposed into (1) an induced STN involving only the waypoints, and (2) several STNUs corresponding to the subnetworks *between* the waypoints" (Morris and Muscettola, 2000). Morris and Muscettola also show that in certain special cases, being waypoint controllable implies being dynamically controllable. Finally, they provide an exponential propagation algorithm that an agent can use in real time when executing tasks in a waypoint controllable STNU (Morris and Muscettola, 1999).

**Safe and Potentially Safe Networks.** The goal of the above algorithms is to determine whether a given network has some controllability property and, if so, to provide a strategy for executing that network. In contrast, Morris and Muscettola (2000) define *safe* and *potentially safe* networks, and provide algorithms not only for determining whether a given network is potentially safe, but also for converting a potentially safe network into a safe network by adding new constraints to the network.

Morris and Muscettola define an STNU to be *safe*, if "for every valid execution and for each contingent link, the only effective propagations to the contingent link finishing point are those that propagate through the contingent link itself." They point out that safe networks are necessarily dynamically controllable. Morris and Muscettola provide a polynomial algorithm for verifying the safety of an STNU that is based on the notion of dominance presented by Tsamardinos et al. (2000)—which differs from the notion of dominance defined in Section 4.2 in that it is tied to the *dispatchability* of the network (i.e., the ability of the network to be executed in real-time). They show that checking whether a network is *potentially safe* is in $\mathcal{NP}$ and provide an algorithm for converting potentially safe networks into safe networks. They conjecture that the property of being potentially safe is closely related to the dynamic controllability property.

Figure 5.43 summarizes the known relationships among the various controllability properties.

**Weak Controllability and Relative Temporal Decouplings.** This section shows that an STNU that is weakly controllable is equivalent to an STN whose contingent subnetworks are temporally decoupled relative to the set of time-points directly controlled by the agent. The equivalence depends on modifying the definition of a relative temporal decoupling (cf. Section 5.5.1) to allow separation vertices other than $z$.

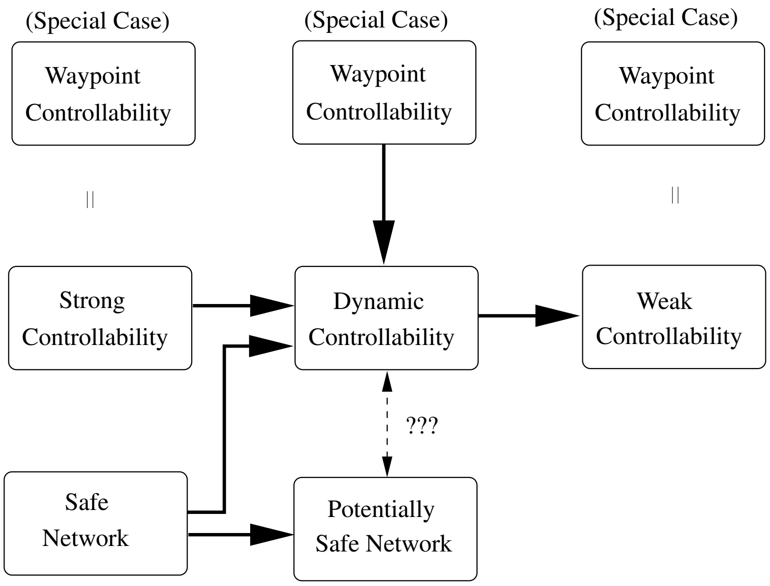



Figure 5.43: Relationships among network controllability properties

Let $\mathcal{S}_U = (\mathcal{T}, \mathcal{C}, \mathcal{C}^c)$ be an STNU that is equivalent to the STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$, except that the set $\mathcal{C}^c \subseteq \mathcal{C}$ contains constraints representing the contingent links in $\mathcal{S}_U$. As described above, the constraints in $\mathcal{C}^c$ come in pairs,

$$E_f\colon t_j - t_i \leq \delta_1 \qquad \text{and} \qquad E_r\colon t_i \leq t_j \leq -\delta_2,$$

such that $0 < \delta_1 < \delta_2 < \infty$. Notice that the length of the forward edge is positive, whereas the length of the reverse edge negative.

STNUs are not allowed to have multiple contingent links terminating at the same time-point (since doing so would make the STNU uncontrollable). However, multiple contingent links may start at the same point. In addition, contingent links may form chains. Thus, the set of contingent links in an STNU form a disjoint set of *trees*, $T_1, \ldots, T_n$, each tree having a unique root node.

For each tree $T_i$, let the *contingent subnetwork* corresponding to $T_i$ be the subnetwork $\mathcal{S}_i^c = (\mathcal{T}_i^c, \mathcal{C}_i^c)$, where $\mathcal{T}_i^c \subseteq \mathcal{T}$ is the set of time-points in the tree $T_i$, and $\mathcal{C}_i^c \subseteq \mathcal{C}_c$ is the set of constraints corresponding to the contingent links in $T_i$. Notice that, like rc-subnetworks (cf. Definition 4.50), contingent subnetworks need not contain the zero time-point variable, $z$. Let $\mathcal{T}_W$ be the set of time-points remaining when all of the non-root time-points from the contingent subnetworks are removed from $\mathcal{T}$. Notice that $\mathcal{T}_W$ contains the root time-point for each tree of contingent links.

If the definition of a relative temporal decoupling is modified to allow decouplings in which time-points other than $z$ are permitted to serve as separation vertices, then $\mathcal{S}_U$ is weakly controllable if and only if the contingent subnetworks $\mathcal{S}_1^c, \ldots, \mathcal{S}_n^c$ temporally decouple $\mathcal{S}$ relative to $\mathcal{T}_W$.[11] In such a decoupling, the root node of each tree serves as a

[11]Thus, a corresponding modification to the definition of the lambda bounds in Section 5.5.5 would allow

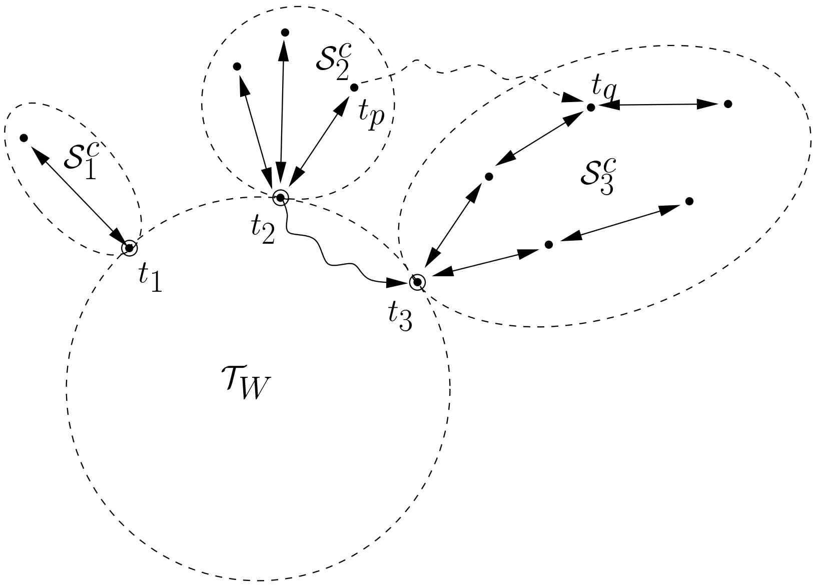



Figure 5.44: A weakly controllable STNU as a temporally decoupled STN.

separation vertex. If the network is temporally decoupled relative to $\mathcal{T}_W$, then any path joining non-root time-points, $t_p$ and $t_q$, where $t_p$ and $t_q$ belong to different contingent subnetworks, is dominated by a path through the root nodes of the trees containing $t_p$ and $t_q$, as illustrated in Figure 5.44, where the path from $t_p$ to $t_q$ (shown as a dashed path) is dominated by the path from $t_p$ to $t_2$ to $t_3$ to $t_q$. For convenience, contingent links are shown as double-headed arrows in the figure.

**Summary.** Although other connections may exist between STNUs and the temporal decoupling problems presented in this chapter, there are substantial differences. First, STNUs presume a single agent, whereas this chapter accommodates arbitrarily many agents. Second, the contingent subnetworks in an STNU have a restricted form, whereas the subnetworks that temporally decouple an STN are not restricted. Third, when adding constraints to an STNU, for example, to make it *safe* (Morris and Muscettola, 2000), an agent is not allowed to impose any additional constrainedness on contingent links. There is no negotiating with nature! However, when seeking to decouple a network involving tasks being done by multiple agents, constraints that impose on each participant may be added if the participants are willing.

Finally, this chapter presumes that the decoupling is done in advance of execution, based on the negotiation of multiple agents, whereas the work cited above considers real-time execution issues, but involves only a single agent. Combining these threads of research to enable groups of negotiating agents to handle real-time execution issues is also left to future work.

them to apply to an agent controlling the non-contingent durations in an STNU.

### 5.6.3 Other Types of Temporal Networks

One of the most attractive features of Simple Temporal Networks is that the operations required to maintain them can be done in polynomial time. However, STNs do not accommodate either disjunctive or conditional constraints, which limits their applicability. Schwalb and Dechter (1997) present polynomial approximation algorithms for temporal networks involving disjunction. Tsamardinos (2001) presents consistency checking algorithms for temporal networks involving disjunctive or conditional constraints. Extending the work in this chapter to cover temporal networks with disjunctive or conditional constraints is left to future work.

# Chapter 6

# Conclusion

This thesis contributes to both the theory and the implementation of collaborative, multi-agent systems. It contributes to the theory by providing a model of the dynamic function of group decision-making in collaborative activity. It contributes to the implementation by providing an auction-based group decision-making mechanism for solving the Initial-Commitment Decision Problem and a suite of sound, complete and polynomial-time temporal-reasoning algorithms to enable agents to participate effectively in that mechanism. In addition, the formal analysis behind the temporal-reasoning algorithms extends the theory of Simple Temporal Networks (Dechter, Meiri, and Pearl, 1991).

## 6.1 The Coordinated Cultivation of Group-Related Intentions

In the context of single-agent activity, the process of intention cultivation is under the control of a single agent. In the context of collaborative, multi-agent activity, the process of intention cultivation is more complicated. This thesis presents the CCGI model of the coordinated cultivation of group-related intentions which (1) explicitly represents the general prohibition against unilateral intention updating, (2) specifies how group decisions establish obligations that authorize and oblige member agents to update their intentions together; and (3) shows how the intentions of group members motivate them to participate in group decision-making mechanisms. By explicating the dynamic function of group decision making in the coordinated cultivation of group-related intentions, the CCGI model provides a unifying framework for the web of commitments to group planning processes specified by the SharedPlans formalization of collaborative activity.

To reason effectively about the obligations that might be established by group decision-making mechanisms, such mechanisms must be rigorously defined. This thesis provides the GDMM framework for rigorously specifying such mechanisms in Dynamic Deontic Linear Time Temporal Logic (Dignum and Kuiper, 1997). For each mechanism, the specification includes the classes of declarative speech-acts used in the mechanism and the conditions under which such speech-acts are authorized. To illustrate the framework, a sample, proposal-based mechanism is specified and its properties formally analyzed.

## 6.2 The Initial Commitment Decision Problem

This thesis defines the Initial-Commitment Decision Problem and specifies a group decision-making mechanism based on a combinatorial auction that agents can use to solve instances of the ICDP. In the auction-based mechanism, which is specified according to the GDMM framework, agents bid on sets of tasks in a proposed activity using temporal constraints in their bids to protect the feasibility of their private schedules of pre-existing commitments. The auction thus serves to mediate between computations that are best done locally, by individuals, and those that require global computations. To participate effectively in such a mechanism, agents need to solve several temporal reasoning problems. The definitions of, and solutions to, these problems are the focus of the second half of the thesis.

## 6.3 Temporal Reasoning for Collaborative Group Activities

This thesis defines and formally analyzes an important family of temporal decoupling problems. The basic Temporal Decoupling Problem involves searching for constraints to add to a temporal network to partition it into independent subnetworks. The Relative TDP involves partitioning a portion of a temporal network into independent subnetworks. The Allocative TDP involves not only adding temporal constraints, but also allocating time-point to various subnetworks.

For each type of Temporal Decoupling Problem, the problem is formally defined in terms of Simple Temporal Networks (Dechter, Meiri, and Pearl, 1991), theorems characterizing solutions are proven, and a family of sound, complete and polynomial-time algorithms is given. For the special case of two decoupled subnetworks, a *minimal* temporal decoupling is defined and an Iterative Weakening algorithm is provided that, given some arbitrary decoupling, generates a minimal decoupling by iteratively weakening intra-subnetwork constraints.

The algorithms for solving the different Temporal Decoupling Problems may be directly applied to the sorts of temporal reasoning problems that collaborating agents must solve. For example, to generate temporal constraints for bids in the auction-based mechanism, agents can use an algorithm based on the Relative TDP algorithm. Furthermore, when an agent has outstanding bids, it is subject to tighter restrictions on the temporal constraints that it can safely add to its private schedule. These tighter restrictions are specified by the *lambda bounds* that this thesis derives for relative temporal decouplings.

The computations associated with generating bids for an auction-based mechanism depend on the size of the temporal network upon which the bid-generation computations are based. To reduce the burden of bid-generation computations, an agent must be able to limit the portion of its private schedule of pre-existing commitments upon which its bid-generation computations are based. This thesis provides an algorithm for finding approximate solutions to the Allocative TDP that can be directly applied to this problem.

Evan after allocating tasks to various group members as determined by a successful instance of an ICDP auction, there may be temporal dependencies among tasks assigned to different agents. To avoid violating such temporal constraints, agents must be able to coordinate their post-auction activity. This thesis presents several strategies for doing so.

The first strategy involves adding new constraints to the network sufficient to temporally decouple the tasks being done by different agents. This method, which enables each agent to operate independently, is directly related to the General Temporal Decoupling Problem.

The second strategy involves temporally decoupling the tasks being done by *some* of the agents, while the tasks being done by the rest of the agents remain dependent. This method, which allows the agents controlling the decoupled tasks to operate independently, while restricting the freedom of the rest of the agents, is directly related to the Relative Temporal Decoupling Problem.

The restrictions on the freedom of the rest of the agents are specified by the corresponding lambda bounds for the relative temporal decoupling. The properties of the lambda bounds indicate that although the agents controlling the non-decoupled tasks may initially be restricted in the constraints they are allowed to add to the network, once the agents controlling the decoupled tasks decide when they will execute their tasks, the rest of the agents may end up having greater flexibility than they would have had under the first strategy.

The third strategy involves applying the second strategy recursively, resulting in a task-dependency hierarchy that can take advantage of the structure of the temporal dependencies among tasks in the group activity to provide flexibility to the agents that need it the most.

# Appendix A

# Proofs of Theorems for the Proposal-based Group Decision-Making Mechanism

Chapter 2 presented a framework for rigorously specifying group decision-making mechanisms. The formal analysis in that chapter included several theorems whose proofs are provided below.

**Theorem 2.8** If an agent $G$ believes that a possibly different agent $G_1$ made an authorized declaration of a context-free proposal at some point in the past, then $G_1$ did, in fact, do so:

$$\begin{aligned} \models \;& Bel(G,\ \Diamond^{\leftarrow} authDecl(G_1, GR, MadeCFP(G_1, GR, I, \alpha, \Upsilon_0))) \\ & \Rightarrow\ \Diamond^{\leftarrow} authDecl(G_1, GR, MadeCFP(G_1, GR, I, \alpha, \Upsilon_0)). \end{aligned}$$

**Proof** Let $\psi$ abbreviate $MadeCFP(G_1, GR, I, \alpha, \Upsilon_0)$.

$Bel(G, \Diamond^{\leftarrow} authDecl(G_1, GR, \psi))$
Given.
$\Rightarrow\ Bel(G,\ \Diamond^{\leftarrow}(\mathcal{P}auth(G_1, GR,\ \delta(G_1, GR, \psi))\ \wedge\ declared(G_1, GR, \psi)))$
Definition of $authDecl$.
$\Rightarrow\ Bel(G,\ \Diamond^{\leftarrow}(\mathcal{P}(G_1 \in GR)\ \wedge\ declared(G_1, GR, \psi)))$
Definition of $auth(G_1, GR, \delta(G_1, GR, \psi))$.
$\Rightarrow\ (G_1 \in GR)\ \wedge\ Bel(G, \Diamond^{\leftarrow} declared(G_1, GR, \psi))$
Axiom 2.6.
$\Rightarrow\ (G_1 \in GR)\ \wedge\ \Diamond^{\leftarrow} declared(G_1, GR, \psi)$.
Axiom 2.4.
$\Rightarrow\ \Diamond^{\leftarrow}(\mathcal{P}(G_1 \in GR)\ \wedge\ declared(G_1, GR, \psi))$.
Axiom 2.6.
$\Rightarrow\ \Diamond^{\leftarrow}(\mathcal{P}auth(G_1, GR, \delta(G_1, GR, \psi))\ \wedge\ declared(G_1, GR, \psi))$
Definition of $auth(G_1, GR, \delta(G_1, GR, \psi))$.
$\Rightarrow\ \Diamond^{\leftarrow} authDecl(G_1, GR, \psi)$
Definition of $authDecl$. ■

**Theorem 2.9** If an agent $G$ believes it is authorized to vote either to accept or reject a context-free proposal, then $G$ is, in fact, so authorized:

$$\models \ \ \mathit{Bel}(G, \mathit{auth}(G, G_1, \delta(G, G_1, \xi))) \ \Rightarrow \ \mathit{auth}(G, G_1, \delta(G, G_1, \xi)),$$

where $\xi$ is $\mathit{AcceptedCFP}(G, G_1, GR, I, \alpha, \Upsilon_0)$ or $\mathit{RejectedCFP}(G, G_1, GR, I, \alpha, \Upsilon_0)$. Similarly, if the originator $G_1$ of the proposal believes $G$ is authorized to vote on that proposal, then $G$ is, in fact, so authorized.

**Proof** $\mathit{Bel}(G, \mathit{auth}(G, G_1, \delta(G, G_1, \xi)))$ Given.
$\Rightarrow \mathit{Bel}(G, V_1 \wedge V_2 \wedge V_3 \wedge V_4 \wedge V_5)$ Definition of $\mathit{auth}(G, G_1, \delta(G, G_1, \xi))$
$\Rightarrow \mathit{Bel}(G, V_1) \wedge \ldots \wedge \mathit{Bel}(G, V_5)$ Distribute belief over conjunction.

Consider each of $\mathit{Bel}(G, V_1), \ldots, \mathit{Bel}(G, V_5)$ in turn.

- For $\mathit{Bel}(G, V_1)$:

  $\Rightarrow \ \mathit{Bel}(G, \Diamond^{\leftarrow} \mathit{MadeCFP}(G_1, GR, I, \alpha, \Upsilon_0))$
  Definition of $V_1$

  $\Rightarrow \ \mathit{Bel}(G, \Diamond^{\leftarrow} \mathit{authDecl}(G_1, GR, \mathit{MadeCFP}(G_1, GR, I, \alpha, \Upsilon_0)))$
  Axiom 2.3

  $\Rightarrow \ \Diamond^{\leftarrow} \mathit{authDecl}(G_1, GR, \mathit{MadeCFP}(G_1, GR, I, \alpha, \Upsilon_0))$
  Theorem 2.8.

  $\Rightarrow \ \Diamond^{\leftarrow} \mathit{MadeCFP}(G_1, GR, I, \alpha, \Upsilon_0)$
  Axiom 2.3

  $\Rightarrow \ V_1$
  Definition of $V_1$

- For $\mathit{Bel}(G, V_2)$:

  $\Rightarrow \ \mathit{Bel}(G, (G \neq G_1))$
  Given

  $\Rightarrow \ (G \neq G_1)$
  Axiom 2.6

  $\Rightarrow \ V_2$
  Definition of $V_2$

- For $\mathit{Bel}(G, V_3)$:

  $\Rightarrow \ \mathit{Bel}(G, \neg \Diamond^{\leftarrow} \mathit{declared}(G, G_1, \mathit{AcceptedCFP}(G, G_1, GR, I, \alpha, \Upsilon_0)))$
  Definition of $V_3$

  $\Rightarrow \ \neg \Diamond^{\leftarrow} \mathit{declared}(G, G_1, \mathit{AcceptedCFP}(G, G_1, GR, I, \alpha, \Upsilon_0))$
  Axiom 2.4

  $\Rightarrow \ V_3$
  Definition of $V_3$

- For $Bel(G, V_4)$ and $Bel(G, V_5)$: Similar to $Bel(G, V_3)$ case.

The above proof only depended on $G$ being a participant in the authorization (i.e., $G \in \{G, G_1\}$). Thus, it is also a proof for second case (where it is $G_1$ that believes $G$ is authorized to vote on the proposal). ■

**Theorem 2.10** If an agent $G_1$ believes that another agent $G$ accepted a context-free proposal originated by $G_1$, then $G$ did, in fact, accept that proposal:

$$\models \; Bel(G_1, AcceptedCFP(G, G_1, GR, I, \alpha, \Upsilon_0)) \\ \Rightarrow \; AcceptedCFP(G, G_1, GR, I, \alpha, \Upsilon_0).$$

**Proof** Let $\xi$ abbreviate $AcceptedCFP(G, G_1, GR, I, \alpha, \Upsilon_0)$.

$Bel(G_1, \xi)$
Given

$\Rightarrow \; Bel(G_1, authDecl(G, G_1, \xi))$
Axiom 2.3

$\Rightarrow \; Bel(G_1, \; \mathcal{P}auth(G, G_1, \delta(G, G_1, \xi)) \; \wedge \; declared(G, G_1, \xi))$
Definition of $authDecl$

$\Rightarrow \; Bel(G_1, \; \mathcal{P}(V_1 \wedge V_2 \wedge V_3 \wedge V_4 \wedge V_5) \; \wedge \; declared(G, G_1, \xi))$
Definition of $auth(G, G_1, \delta(G, G_1, \xi))$

$\Rightarrow \; Bel(G_1, \mathcal{P}V_1) \; \wedge \; \ldots \; \wedge \; Bel(G_1, \mathcal{P}V_5) \; \wedge \; Bel(G_1, declared(G, G_1, \xi))$
Distribute belief over conjunction

$\Rightarrow \; \mathcal{P}Bel(G_1, V_1) \; \wedge \; \ldots \; \wedge \; \mathcal{P}Bel(G_1, V_5) \; \wedge \; Bel(G_1, declared(G, G_1, \xi))$
Axiom 2.7

$\Rightarrow \; \mathcal{P}V_1 \; \wedge \; \ldots \; \wedge \; \mathcal{P}V_5 \; \wedge \; Bel(G_1, declared(G, G_1, \xi))$
Follows from same techniques used in proof of Theorem 2.9

$\Rightarrow \; \mathcal{P}V_1 \; \wedge \; \ldots \; \wedge \; \mathcal{P}V_5 \; \wedge \; declared(G, G_1, \xi)$
Axiom 2.4

$\Rightarrow \; authDecl(G, G_1, \xi)$
Definitions of $auth(G, G_1, \delta(G, G_1, \xi))$ and $authDecl$

$\Rightarrow \; \xi$
Axiom 2.3 ■

**Theorem 2.11** If an agent $G_1$ believes that it is authorized to declare, on behalf of the group, that the group has accepted a context-free proposal, then $G_1$ is, in fact, so authorized:

$$\models \; Bel(G_1, auth(G_1, GR, \; \delta(G_1, GR, CCR^+(GR, I, \alpha, \Upsilon_0)))) \\ \Rightarrow \; auth(G_1, GR, \; \delta(G_1, GR, CCR^+(GR, I, \alpha, \Upsilon_0))).$$

**Proof** $Bel(G_1, auth(G_1, GR, \delta(G_1, GR, CCR^+(GR, I, \alpha, \Upsilon_0))))$
Given
$\Rightarrow\ Bel(G_1, A_1 \wedge A_2 \wedge A_3 \wedge A_4)$
Definition of $auth(G_1, GR, \delta(G_1, GR, CCR^+(GR, I, \alpha, \Upsilon_0)))$
$\Rightarrow\ Bel(G_1, A_1) \wedge \ldots \wedge Bel(G_1, A_4)$

Consider each of $A_1, A_2, A_3$ and $A_4$ in turn.

- For $Bel(G_1, A_1)$:
  $\Rightarrow\ Bel(G_1, \Diamond^{\leftarrow} MadeCFP(G_1, GR, I, \alpha, \Upsilon_0))$
  Definition of $A_1$
  $\Rightarrow\ Bel(G_1, \Diamond^{\leftarrow} authDecl(G_1, GR, MadeCFP(G_1, GR, I, \alpha, \Upsilon_0)))$
  Axiom 2.3
  $\Rightarrow\ \Diamond^{\leftarrow} authDecl(G_1, GR, MadeCFP(G_1, GR, I, \alpha, \Upsilon_0))$
  Theorem 2.8
  $\Rightarrow\ \Diamond^{\leftarrow} MadeCFP(G_1, GR, I, \alpha, \Upsilon_0)$
  Axiom 2.3
  $\Rightarrow\ A_1$
  Definition of $A_1$

- For $Bel(G_1, A_2)$:
  $\Rightarrow\ Bel(G_1, \neg \Diamond^{\leftarrow}\ declared(G_1, GR, GroupRejectedCFP(G_1, GR, I, \alpha, \Upsilon_0)))$
  Definition of $A_2$
  $\Rightarrow\ \neg \Diamond^{\leftarrow}\ declared(G_1, GR, GroupRejectedCFP(G_1, GR, I, \alpha, \Upsilon_0))$
  Axiom 2.4
  $\Rightarrow\ A_2$
  Definition of $A_2$

- For $Bel(G_1, A_3)$: Similar to $A_2$ case.

- For $Bel(G_1, A_4)$:
  $\Rightarrow\ Bel(G_1, (\forall G \in GR, G \neq G_1)\ \Diamond^{\leftarrow} AcceptedCFP(G, G_1, GR, I, \alpha, \Upsilon_0))$
  Definition of $A_4$
  $\Rightarrow\ (\forall G \in GR, G \neq G_1)\ Bel(G_1, \Diamond^{\leftarrow} AcceptedCFP(G, G_1, GR, I, \alpha, \Upsilon_0))$
  Axiom 2.6
  $\Rightarrow\ (\forall G \in GR, G \neq G_1)\ \Diamond^{\leftarrow} Bel(G_1, AcceptedCFP(G, G_1, GR, I, \alpha, \Upsilon_0))$
  Theorem 2.7
  $\Rightarrow\ (\forall G \in GR, G \neq G_1)\ \Diamond^{\leftarrow} AcceptedCFP(G, G_1, GR, I, \alpha, \Upsilon_0)$
  Theorem 2.10
  $\Rightarrow\ A_4$
  Definition of $A_4$ ■

**Theorem 2.12** If an agent $G_1$ declares, on behalf of the group, that the group has accepted a context-free proposal, then that agent was, in fact, authorized to make that declaration:

$$\begin{aligned}\models\ & declared(G_1, GR,\ CCR^+(GR, I, \alpha, \Upsilon_0)) \\ & \Rightarrow\ auth(G_1, GR,\ \delta(G_1, GR, CCR^+(GR, I, \alpha, \Upsilon_0))).\end{aligned}$$

Hence,

$$\models \ declared(G_1, GR,\ CCR^+(GR, I, \alpha, \Upsilon_0)) \\ \Rightarrow\ authDecl(G_1, GR,\ CCR^+(GR, I, \alpha, \Upsilon_0)).$$

**Proof** Let $\omega$ abbreviate $CCR^+(GR, I, \alpha, \Upsilon_0)$.

| | |
|---|---|
| $declared(G_1, GR, \omega)$ | Given |
| $Bel(G_1, auth(G_1, GR, \delta(G_1, GR, \omega)))$ | Axiom 2.5 (the sincerity assumption) |
| $auth(G_1, GR, \delta(G_1, GR, \omega))$ | Theorem 2.11. ∎ |

**Theorem 2.13** If an agent $G$ believes that a possibly different agent $G_1$ declared that the group $GR$ accepted a context-free proposal, then $G_1$ was, in fact, authorized to make such a declaration and did so:

$$\models \ Bel(G, \Diamond^{\leftarrow} declared(G_1, GR,\ CCR^+(GR, I, \alpha, \Upsilon_0))) \\ \Rightarrow\ \Diamond^{\leftarrow} authDecl(G_1, GR,\ CCR^+(GR, I, \alpha, \Upsilon_0)).$$

**Proof** Let $\omega$ abbreviate $CCR^+(GR, I, \alpha, \Upsilon_0)$.

| | |
|---|---|
| $Bel(G, \Diamond^{\leftarrow} declared(G_1, GR, \omega))$ | Given |
| $\Rightarrow\ \Diamond^{\leftarrow} Bel(G, declared(G_1, GR, \omega))$ | Axiom 2.7 |
| $\Rightarrow\ \Diamond^{\leftarrow} declared(G_1, GR, \omega)$ | Theorem 2.4 |
| $\Rightarrow\ \Diamond^{\leftarrow} authDecl(G_1, GR, \omega)$ | Theorem 2.12 ∎ |

**Theorem 2.14** If an agent $G$ believes that a possibly different agent $G_1'$ made an authorized declaration of a context-bound proposal at some point in the past, then $G_1'$ did, in fact, do so:

$$\models \ Bel(G, \Diamond^{\leftarrow} authDecl(G_1', GR, MadeCBP(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0))) \\ \Rightarrow\ \Diamond^{\leftarrow} authDecl(G_1', GR, MadeCBP(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0)).$$

**Proof** Let $\theta$ abbreviate $MadeCBP(G_1', \Upsilon, P, G_1, GR, I, \alpha, \Upsilon_0)$.

$Bel(G, \Diamond^{\leftarrow} authDecl(G_1', GR, \theta))$
Given

$\Rightarrow\ \Diamond^{\leftarrow} Bel(G, authDecl(G_1', GR, \theta))$
Axiom 2.7

$\Rightarrow\ \Diamond^{\leftarrow} Bel(G, \mathcal{P} auth(G_1', GR, \delta(G_1', GR, \theta)) \wedge declared(G_1', GR, \theta))$
Definition of $authDecl$

$$\Rightarrow\ \Diamond^{\leftarrow} Bel(G, \begin{pmatrix} \mathcal{P}((G_1' \in GR) \wedge \Diamond^{\leftarrow} authDecl(G_1, GR, \omega_c)) \\ \wedge\ declared(G_1', GR, \theta) \end{pmatrix}),$$

where $\omega_c = CCR^+(GR, I, \alpha, \Upsilon_0)$
Definition of $auth(G_1', GR, \delta(G_1', GR, \theta))$

$$\Rightarrow\ \Diamond^{\leftarrow} \begin{pmatrix} Bel(G, \mathcal{P}((G_1' \in GR)\ \wedge\ \Diamond^{\leftarrow} authDecl(G_1, GR, \omega_c))) \\ \wedge\ Bel(G, declared(G_1', GR, \theta)) \end{pmatrix},$$

Distribute belief over conjunctions

$$\Rightarrow \ \Diamond^{\leftarrow} \begin{pmatrix} \mathcal{P}Bel(G, (G_1' \in GR)) \ \wedge \ \mathcal{P}Bel(G, \Diamond^{\leftarrow} authDecl(G_1, GR, \omega_c)) \\ \wedge \ Bel(G, declared(G_1', GR, \theta)) \end{pmatrix}$$

Axiom 2.7 and distribute belief over conjunctions

$$\Rightarrow \ \Diamond^{\leftarrow} \begin{pmatrix} \mathcal{P}(G_1' \in GR) \ \wedge \ \mathcal{P}Bel(G, \Diamond^{\leftarrow} authDecl(G_1, GR, \omega_c)) \\ \wedge \ Bel(G, declared(G_1', GR, \theta)) \end{pmatrix}$$

Axiom 2.6

$$\Rightarrow \ \Diamond^{\leftarrow} \begin{pmatrix} \mathcal{P}(G_1' \in GR) \ \wedge \ \mathcal{P}Bel(G, \Diamond^{\leftarrow} declared(G_1, GR, \omega_c)) \\ \wedge \ Bel(G, declared(G_1', GR, \theta)) \end{pmatrix}$$

Since $authDecl \Rightarrow declared$

$$\Rightarrow \ \Diamond^{\leftarrow} \begin{pmatrix} \mathcal{P}(G_1' \in GR) \ \wedge \ \mathcal{P} \ \Diamond^{\leftarrow} authDecl(G_1, GR, \omega_c) \\ \wedge \ Bel(G, declared(G_1', GR, \theta)) \end{pmatrix}$$

Theorem 2.13

$$\Rightarrow \ \Diamond^{\leftarrow} \begin{pmatrix} \mathcal{P}(G_1' \in GR) \ \wedge \ \mathcal{P} \ \Diamond^{\leftarrow} authDecl(G_1, GR, \omega_c) \\ \wedge \ declared(G_1', GR, \theta) \end{pmatrix}$$

Axiom 2.4

$$\Rightarrow \ \Diamond^{\leftarrow}(\mathcal{P}auth(G_1', GR, \delta(G_1', GR, \theta)) \ \wedge \ declared(G_1', GR, \theta))$$

Definition of $auth(G_1', GR, \delta(G_1', GR, \theta))$

$$\Rightarrow \ \Diamond^{\leftarrow} authDecl(G_1', GR, \theta)$$

Definition of $authDecl$ ■

# References


Andersson, Arne, Mattias Tenhunen, and Fredrik Ygge. 2000. Integer programming for combinatorial auction winner determination. In *Proceedings, Fourth International Conference on MultiAgent Systems (ICMAS-2000)*. IEEE Computer Society, Los Alamitos, CA, pages 39–46.

Austin, J.L. 1962. *How to do things with words*. Harvard University Press, Cambridge, MA.

Bratman, Michael E. 1987. *Intentions, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA.

Bratman, Michael E. 1999. *Faces of Intention: Selected Essays on Intention and Agency*. Cambridge University Press.

Brown, Mark A. 2000. Conditional obligation and positive permission for agents in time. *Nordic Journal of Philosophical Logic*, 5(2):83–112.

Castelfranchi, Cristiano. 1995. Commitments: From individual intentions to groups and organizations. In Victor Lesser, editor, *First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 41–48. The MIT Press.

Chellas, Brian. 1980. *Modal Logic: An Introduction*. Cambridge University Press, Cambridge, England.

Chleq, Nicolas. 1995. Efficient algorithms for networks of quantitative temporal constraints. In *Proceedings of CONSTRAINTS-95*, pages 40–45.

Cohen, Philip R. and Hector J. Levesque. 1990. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261.

Cohen, Philip R., Hector J. Levesque, and Ira Smith. 1997. On team formation. In G. Holmstrom-Hintikka and R. Tuomela, editors, *Contemporary Action Theory*, volume II. Kluwer Academic Publishers, Netherlands, pages 87–114.

Collins, John, Rashmi Sundareswara, Maksim Tsvetovat, Maria Gini, and Bamshad Mobasher. 2000. Bid selection strategies for multi-agent contracting in the presence of scheduling constraints. In Alexandros Moukas, Carles Sierra, and Fredrik Ygge, editors, *Agent Mediated Electronic Commerce (AmEC-99)*, volume 1788 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin.

Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest. 1990. *Introduction to Algorithms*. The MIT Press, Cambridge, MA.

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

Dignum, Frank and Ruurd Kuiper. 1997. Combining dynamic deontic logic and temporal logic for the specification of deadlines. In Jr. Ralph H. Sprague, editor, *Proceedings of Thirtieth Hawaii International Conference on System Sciences (HICSS)*.

Dignum, Frank and Hans Weigand. 1995a. Communication and deontic logic. In R. Wieringa and R. Feenstra, editors, *Information Systems, Correctness and Reusability*. World Scientific, pages 242–260.

Dignum, Frank and Hans Weigand. 1995b. Modelling communication between cooperative systems. In *Conference on Advanced Information Systems Engineering*, pages 140–153.

Estlin, Tara, Gregg Rabideau, Darren Mutz, and Steve Chien. 2000. Using continuous planning techniques to coordinate multiple rovers. *Electronic Transactions on Artificial Intelligence*, 4:45–57.

Fujishima, Yuzo, Kevin Leyton-Brown, and Yoav Shoham. 1999. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In Thomas Dean, editor, *Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 548–553. Morgan Kaufman Publishers, Inc.

Gerevini, Alfonso, Anna Perini, and Francesco Ricci. 1996. Incremental algorithms for managing temporal constraints. Technical Report IRST-9605-07, IRST.

Gilbert, Margaret. 2000. *Sociality and Responsibility*. Rowman & Littlefield Publishers, Inc., New York.

Grosz, Barbara J. 1996. AAAI-94 Presidential Address: Collaborative systems. *AI Magazine*, pages 67–85.

Grosz, Barbara J. and Sarit Kraus. 1996. Collaborative plans for complex group action. *Artificial Intelligence*, 86:269–357.

Grosz, Barbara J. and Sarit Kraus. 1999. The evolution of SharedPlans. In Michael Wooldridge and Anand Rao, editors, *Foundations of Rational Agency*, number 14 in Applied Logic Series. Kluwer Academic Publishers, The Netherlands, pages 227–262.

Grosz, Barbara J. and Candace L. Sidner. 1990. Plans for discourse. In Philip R. Cohen, J. Morgan, and Martha E. Pollack, editors, *Intentions in Communication*. The MIT Press, Cambridge, Massachusetts, chapter 20.

Horty, John F. and Martha E. Pollack. 2001. Evaluating new options in the context of existing plans. *Artificial Intelligence*, 127:199–220.

Hunsberger, Luke. 1999. Making SharedPlans more concise and easier to reason about. In *Agent Architectures, Theories and Languages V*, volume 1555 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, pages 81–98.

Hunsberger, Luke. 2002a. Algorithms for a temporal decoupling problem in multi-agent planning. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*. Forthcoming.

Hunsberger, Luke. 2002b. Generating bids for group-related actions in the context of prior commitments. In John-Jules Ch. Meyer and Milind Tambe, editors, *Intelligent Agents VIII (ATAL-01)*, volume 2333 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag.

Hunsberger, Luke and Barbara J. Grosz. 2000. A combinatorial auction for collaborative planning. In *Fourth International Conference on MultiAgent Systems (ICMAS-2000)*, pages 151–158. IEEE Computer Society.

Hunsberger, Luke and Massimo Zancanaro. 2000. A mechanism for group decision making in collaborative activity. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 30–35, Cambridge, MA. The MIT Press.

Jennings, N. R. 1995. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75:195–240.

Kinny, D., M. Ljungberg, A.S. Rao, E. Sonenberg, G. Tidhar, and E. Werner. 1994. Planned team activity. In C. Castelfranchi and E. Werner, editors, *Artificial Social Systems*, Lecture Notes in Artificial Intelligence. Springer Verlag, Amsterdam.

Kutanoglu, Erhan and S. David Wu. 1997. On combinatorial auction and Lagrangean relaxation for distributed resource scheduling. Technical Report 97T-012, Lehigh University.

Laird, John E. 2000. Human-level AI's killer application: Interactive computer games. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, Cambridge, MA. The MIT Press.

Leiserson, C.E. and J.B. Saxe. 1983. A mixed-integer linear programming problem which is efficiently solvable. In *21st Annual Allerton Conference on Communications, Control and Computing*, pages 204–213.

Levesque, Hector J., Philip R. Cohen, and Jose H. T. Nunes. 1990. On acting together. In *Seventh National Conference on Artificial Intelligence*, volume 1, pages 94–99. AAAI Press/MIT Press.

Liao, Y.Z. and C.K. Wong. 1983. An algorithm to compact a VLSI symbolic layout with mixed constraints. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 2(2):62–69.

Morris, Paul and Nicola Muscettola. 1999. Managing temporal uncertainty through waypoint controllability. In Thomas Dean, editor, *Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 1253–1258, San Francisco, CA. Morgan Kaufman Publishers, Inc.

Morris, Paul and Nicola Muscettola. 2000. Execution of temporal plans with uncertainty. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 491–496, Cambridge, MA. The MIT Press.

Morris, Paul, Nicola Muscettola, and Thierry Vidal. 2001. Dynamic control of plans with temporal uncertainty. In Bernhard Nebel, editor, *Proceedings, Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 494–499, San Francisco, CA. Morgan Kaufmann Publishers, Inc.

Ortiz, Jr., Charles L. 1999. Introspective and elaborative processes in rational agents. *Journal of the Annals of Mathematics and Artificial Intelligence*.

Parkes, David C. and Lyle H. Ungar. 2001. An auction-based method for decentralized train scheduling. In *Proceedings of Fifth Annual Conference on Autonomous Agents (Agents-01)*.

Rassenti, S., V. Smith, and R. Bulfin. 1982. A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics*, 13:402–417.

Royakkers, Lamber and Frank Dignum. 1999. Collective obligation and commitment. In *Proceedings International Conference on Artificial Intelligence and Law*. June.

Russell, Stuart J. and Peter Norvig. 1995. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ.

Sandholm, Tuomas. 1999. An algorithm for optimal winner determination in combinatorial auctions. In Thomas Dean, editor, *Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*. Morgan Kaufman Publishers, Inc.

Sandholm, Tuomas, Subhash Suri, Andrew Gilpin, and David Levine. 2001. CABOB: A fast optimal algorithm for combinatorial auctions. In *Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*.

Schwalb, Eddie and Rina Dechter. 1997. Processing disjunctions in temporal constraint networks. *Artificial Intelligence*, 93:29–61.

Searle, John R. 1990. Collective intentions and action. In Philip R. Cohen, J. Morgan, and Martha E. Pollack, editors, *Intentions in Communication*. The MIT Press, Cambridge, Massachusetts.

Searle, J.R. 1995. *The Construction of Social Reality*. Allen Lane, London.

Shostak, R. 1981. Deciding linear inequalities by computing loop residues. *ACM*, 28(4).

Simon, Herbert A. 1969. *The sciences of the artificial*. Karl Taylor Compton lectures, 1968. MIT Press, Cambridge, MA.

Singh, Munindar P. 1991. Social and psychological commitments in multiagent systems. In *AAAI Fall Symposium on Knowledge and Action at Social and Organizational Levels*.

Singh, Munindar P. 1992. A critical examination of the Cohen-Levesque theory of intentions. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 364–368.

Singh, Munindar P. 1996. A conceptual analysis of commitments in multiagent systems. Technical Report TR-96-09, North Carolina State University, May.

Singh, Munindar P. 2000. A social semantics for agent communication languages. In *Proceedings of the IJCAI Workshop on Agent Communication Languages*. Springer-Verlag.

Tambe, Milind. 1997. Agent architectures for flexible, practical teamwork. In *Fourteenth National Conference on Artificial Intelligence*, pages 22–28, Cambridge, MA. AAAI Press/MIT Press.

Tsamardinos, Ioannis. 2000. Reformulating temporal plans for efficient execution. Master's thesis, University of Pittsburgh.

Tsamardinos, Ioannis. 2001. *Constraint-Based Temporal Reasoning Algorithms with Applications to Planning*. Ph.D. thesis, University of Pittsburgh.

Tsamardinos, Ioannis, Nicola Muscettola, and Paul Morris. 1998. Fast transformation of temporal plans for efficient execution. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*. The MIT Press, Cambridge, MA, pages 254–261.

Tuomela, Raimo. 1995. *The Importance of Us: A Philosophical Study of Basic Social Notions*. Stanford University Press, Stanford, CA.

Venkatraman, Mahadevan and Munindar P. Singh. 1999. Verifying compliance with commitment protocols: Enabling open web-based multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 2(3):217–236.

Vidal, Thierry and Hélène Fargier. 1999. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental and Theoretical Computer Science*, 11:23–45.

Walsh, William, Michael P. Wellman, and Fredrik Ygge. 2000. Combinatorial auctions for supply chain formation. In *ACM Conference on Electronic Commerce*, pages 260–269.

Walsh, William E. and Michael P. Wellman. 1998. A market protocol for decentralized task allocation and scheduling with hierarchical dependencies. In *Third International Conference on Multi-Agent Systems (ICMAS-98)*. IEEE Computer Society.

Wetprasit, Rattana and Abdul Sattar. 1998. Qualitative and quantitative temporal reasoning with points and durations (an extended abstract). In *TIME*, pages 69–73.