Generating Bids for Group-related Actions in the Context of Prior Commitments

Luke Hunsberger

Division of Engineering and Applied Sciences, Harvard University
33 Oxford St., Cambridge, MA 02138, USA
luke@eecs.harvard.edu

Abstract. This paper addresses the problem of bid generation for a group decision-making mechanism based on a combinatorial auction in which agents bid on sets of tasks in a proposed group activity. It presents algorithms that an agent may use to generate temporal constraints for bids in such auctions. These constraints allow an agent making a bid to protect its private schedule of pre-existing commitments in case the bid is ultimately awarded. A task-integration-scheduling algorithm determines whether a new set of tasks can be merged into an agent's private schedule of pre-existing commitments. A temporal-constraint-generation algorithm generates the weakest set of temporal constraints (for inclusion in a bid) sufficient to protect the agent's modified schedule from any possible auction outcome. A proof of correctness for the temporal-constraint-generation algorithm is given. Both algorithms employ Simple Temporal Networks. Although the algorithms are presented in the context of a particular group decision-making mechanism, they have broader applicability to scenarios in which one agent makes an offer to other agents.

Keywords: Collaborative Planning, Group Decision Making, Combinatorial Auctions, Simple Temporal Networks, Bid Generation, Temporal Constraints.

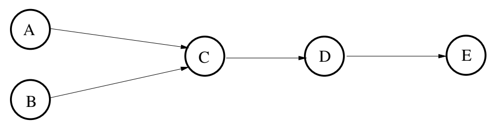
1 Introduction

Suppose a group of agents have been presented with an opportunity to do some group activity α for some reward R. To decide whether they should commit to doing α, they engage a group decision-making mechanism based on a combinatorial auction, as presented by Hunsberger and Grosz [6]. Agents bid on sets of tasks in the proposed group activity. To enable them to protect their private schedules of pre-existing commitments, agents are allowed to include temporal constraints with their bids. When contemplating making a bid, an agent must do the following:

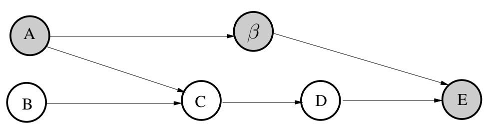
(1) Select a set of tasks to bid on;
(2) Determine ways to integrate those tasks into its private schedule of pre-existing commitments (e.g., by inserting ordering constraints among tasks);
(3) Based on (2), compute the temporal constraints to include with the bid; and
(4) Decide how much it wants to be paid for doing the tasks covered by the bid.

When all the bids are in, a winner-determination algorithm [6, 7, 3] is run to determine if some combination of non-conflicting bids covers all the tasks in the proposed activity for a total cost that is less than the offered reward. If so, the agents may profit by committing to the task.

Example. A group of agents is presented with an opportunity to build a tool shed this afternoon for a reward of $1000. The tasks include: (A) buying the wood, (B) fetching the tools, (C) sawing the wood, (D) building the shed, and (E) cleaning up, as depicted below with arrows representing ordering constraints.



Suppose Bill is already committed to doing a chair-reupholstering job (β) sometime this afternoon, but nonetheless is considering submitting a bid for the two tasks, A and E. First, Bill must integrate the two tasks A and E into his existing schedule. Suppose that Bill decides it would be best for him to do A first, then β, then E, as shown below, where the shaded tasks are to be done by Bill. (Even though Bill is not bidding on B, C or D, they must appear in his schedule since constraints involving those actions might affect when A and E are to be done.)



Next, Bill must determine what temporal constraints to place in his bid. He can't risk not putting any constraints in his bid because he must protect his pre-existing commitment to doing β. However, he can't include constraints such as "A must be finished before β begins" because the rules of the auction stipulate that constraints in a bid may refer only to the actions being bid on, not to arbitrary actions in the bidder's private schedule. Knowing that β will take at most two hours, Bill makes his bid contingent on E starting at least two hours after A finishes. Finally, Bill must submit an amount he wishes to be paid for doing A and E. Given that doing A and E would substantially impact his schedule, Bill adds $20 on top of the $100 in fixed costs he associates with A and E.

Algorithms Presented in this Paper. In earlier work, Hunsberger and Grosz [6] focused on the Winner Determination part of the auction-based group decision-making mechanism described above. In this paper, we focus on the bid-generation problem. In particular, we focus on how agents may generate appropriate temporal constraints for their bids. We present two algorithms, both of which manipulate Simple Temporal Networks (STNs):

- a task-integration scheduling algorithm that an agent may use to determine whether it can take on a new set of tasks; and
- a temporal-constraint-generation algorithm than an agent may use to generate temporal constraints to include in its bid for that set of tasks.

We assume that the problem of which tasks to bid on can be handled by random selection with backtracking. As for computing the desired compensation for a bid, Horty and Pollack [5] have presented an algorithm an agent might use to compute the cost of taking on new responsibilities in the context of pre-existing commitments. In addition, the measure of the *rigidity* of an agent's schedule, presented in this paper, may be used to generate additional cost information based on the impact on an agent's schedule from taking on new responsibilities.

The Task-Integration-Scheduling Algorithm. The task-integration-scheduling (TIS) algorithm is responsible for determining whether a set of new tasks may be merged into an agent's private schedule of pre-existing commitments. The TIS algorithm is allowed to solve this problem by inserting temporal constraints into the agent's schedule. The TIS algorithm presented in this paper maintains the actions the agent is already committed to in a fully-ordered list. (Alternative TIS algorithms might manipulate partially-ordered, rather than fully-ordered commitments.) New actions are integrated into the agent's schedule by inserting ordering constraints sufficient to ensure that the expanded set of actions will be fully ordered, and hence will not temporally overlap. (We assume an agent can only do one primitive action at a time.) The algorithm is a greedy algorithm based on a heuristic that measures the rigidity of the agent's schedule. The algorithm incrementally adds ordering constraints according to which constraint minimizes the resulting rigidity of the STN representing the agent's schedule.

The Temporal-Constraint-Generation Algorithm. The temporal-constraint-generation (TCG) algorithm presumes that the agent has already integrated the tasks being bid on into its private schedule of pre-existing commitments, but is independent of the particular TIS algorithm being used. The TCG algorithm generates constraints among the tasks being bid on sufficient to ensure that none of the task-integration constraints in the agent's private schedule are violated, no matter how the auction ultimately turns out.

An important aspect of the temporal-constraint-generation problem is that the auctioneer typically combines temporal constraints from multiple bids. From the perspective of an individual agent, it appears that the auctioneer is allowed to strengthen (but not violate) temporal constraints associated with the proposed activity. The TCG algorithm manipulates a pair of STNs, one representing the proposed group activity in the context of the agent's schedule, the other representing the proposed group activity in an empty context. The TCG algorithm compares these STNs to generate the weakest set of temporal constraints sufficient to effectively shield the agent's pre-existing schedule from any strengthening of constraints that the auctioneer might add.

A proof of correctness for the TCG algorithm is given.

2 The Setup

2.1 The Opportunity

Time Points. A *time point (TP)* is a point on the real number line. A *time-point variable* is a variable x that takes on real values.

Temporal Constraints. We consider both unary and binary temporal constraints. A *unary temporal constraint (UTC)* is of the form $L \leq x$ or $x \leq U$, where x is a time-point variable, and L and U are real constants (positive, negative or zero). A *binary temporal constraint (BTC)* is of the form $x_j - x_i \leq \delta$, where x_i and x_j are time-point variables, and the *offset* δ is a constant (positive, negative or zero).

Actions. Following Grosz and Kraus [4], an *act type* represents an abstract class of *act instances* (or *actions*). A *basic action* is a primitive action that a single agent may execute at will (under appropriate conditions), whereas a complex action (whether done by a single agent or a group of agents) is executed by executing the set of subacts in a recipe (described below). We assume that each *act type* τ (whether basic or complex) includes a begin time point b_τ and an end time point e_τ, as well as binary temporal constraints on those time points (e.g., to constrain the duration of the action). Unary temporal constraints are not necessary for general-purpose act types (or recipes).

Recipes. A recipe R_α for a complex act type α consists of a set of actions (called *subacts*) and constraints on those subacts such that the execution of the subacts under those constraints constitutes the doing of α [4]. In this paper, we restrict attention to recipes whose subacts are *basic* actions and whose constraints are purely *temporal*. The temporal constraints specified by the recipe may constrain time points defined by the act type α or the recipe R_α.

Opportunities. An *opportunity* OPP_α specifies a (complex, multi-agent) act type α, a recipe R_α for actions of type α, and sets of unary and binary temporal constraints (beyond those defined in α and R_α). Unlike the general-purpose act types and recipes that do not require unary constraints, an opportunity typically needs them. For example, an opportunity to do a `BUILD-SPACE-STATION` activity might have a unary constraint specifying that the activity be completed before January 1, 2016.

The above definitions are formalized in Figure 1.

2.2 The Combinatorial-Auction-based Group Decision-Making Mechanism

When a group of agents are confronted with an opportunity to engage in some group activity, they must decide whether they are able to carry out the proposed activity and, if so, whether they should commit to doing so. Hunsberger and Grosz [6] called this the Initial Commitment Decision Problem (ICDP) and presented a group decision-making mechanism based on a combinatorial auction that agents could use to solve the ICDP. Such an auction has a bidding phase and a winner-determination phase [6, 7, 3]. In the bidding phase, each agent submits zero or more bids, each of which includes a set of subacts being bid on and a set of temporal constraints, as follows.

<u>BID: B</u>

Subacts covered by the bid: Subacts(B)
Temporal Constraints: $\mathcal{TC}(B) = \mathcal{UTC}(B) \cup \mathcal{BTC}(B)$,

where $\mathcal{UTC}(B) =$ Unary constraints over $\mathcal{TP}(\alpha) \cup \mathcal{TP}(R_\alpha)$
and $\mathcal{BTC}(B) =$ Binary constraints over $\mathcal{TP}(\alpha) \cup \mathcal{TP}(R_\alpha)$

TEMPORAL CONSTRAINTS

Unary Constraints: $L \leq x$ or $x \leq U$
Binary Constraints: $x_j - x_i \leq \delta$

ACT TYPE (basic or complex): τ

Time Points: $\mathcal{TP}(\tau) = \{b_\tau, e_\tau\}$
Temporal Constraints: $\mathcal{TC}(\tau) =$ Binary constraints over $\mathcal{TP}(\tau)$

RECIPE: R_α (for act type α)

Subacts: $\{\beta_1, \ldots, \beta_n\}$
Time Points: $\mathcal{TP}(R_\alpha) = \{b_{\beta_1}, \ldots, b_{\beta_n}; e_{\beta_1}, \ldots, e_{\beta_n}\}$
Temporal Constraints: $\mathcal{TC}(R_\alpha) =$ Binary constraints over $\mathcal{TP}(\alpha) \cup \mathcal{TP}(R_\alpha)$

OPPORTUNITY: OPP_α

Act Type: α
Recipe: R_α
Temporal Constraints: $\mathcal{UTC}(\text{OPP}_\alpha) =$ Unary constraints over $\mathcal{TP}(\alpha) \cup \mathcal{TP}(R_\alpha)$
$\mathcal{BTC}(\text{OPP}_\alpha) =$ Binary constraints over $\mathcal{TP}(\alpha) \cup \mathcal{TP}(R_\alpha)$

TIME POINTS and CONSTRAINTS associated with the OPPORTUNITY OPP_α

$$\mathcal{TP}(\text{OPP}_\alpha) = \mathcal{TP}(\alpha) \cup \mathcal{TP}(R_\alpha)$$
$$\mathcal{TC}(\text{OPP}_\alpha) = \mathcal{TC}(\alpha) \cup \mathcal{TC}(R_\alpha) \cup \mathcal{UTC}(\text{OPP}_\alpha) \cup \mathcal{BTC}(\text{OPP}_\alpha)$$

Fig. 1. Constituents of the Definition of an Opportunity to do α using R_α

Bids may also include a *cost* or *reward*; however, in this paper, we focus on feasibility rather than cost-optimization. Thus, we henceforth ignore the costs of bids.

In the winner-determination phase, the auctioneer seeks an *awardable bid-set*—i.e., a set of bids $\mathcal{B}$ such that:

- each subact in R_α is covered by exactly one bid in $\mathcal{B}$; and
- the temporal constraints associated with the opportunity together with the temporal constraints from the bids in $\mathcal{B}$ are simultaneously satisfiable.

If a bid B is awarded to an agent, then that agent must commit to doing the subacts covered by B. Of crucial importance is that that agent must also respect the constraints from *all* the awarded bids. For example, if my bid to wash the dishes sometime between 1:00 and 3:00 is awarded together with your bid to dry the dishes sometime between 12:00 and 2:00, then I must be committed to washing the dishes *before* 2:00 to enable you to finish drying them by 2:00. If my washing the dishes before 2:00 might conflict with some pre-existing commitment of mine (e.g., an ATAL presentation that might take until 2:00), then the constraints in my bid are too weak; they do not protect my pre-existing commitment. One way to protect the pre-existing commitment would be to strengthen the temporal constraint in my bid. (Other options include allowing some form of disjunction or conditionality in the temporal constraints [11, 12, 8]; however, such added expressiveness carries substantial computational costs.) For example, by changing my dish-washing bid so that it more tightly constrains the dish-washing to occur between 2:00 and 3:00, I ensure that no matter what other bid is chosen to cover the dish-drying activity, my pre-existing commitment to do the ATAL presentation will be protected.

More formally, the temporal constraints $\mathcal{TC}(B)$ for a bid B must ensure that if the bid B is awarded, then no matter what additional constraints $\mathcal{T}$ the auctioneer might require from the other awarded bids:

$$\begin{array}{rll} \text{If:} & \mathcal{TC}(\mathrm{OPP}_\alpha) \cup \mathcal{TC}(B) \cup \mathcal{T} & \text{is satisfiable,} \\ \text{Then:} & \mathcal{TC}(\mathrm{OPP}_\alpha) \cup \mathcal{TC}(B) \cup \mathcal{T} \cup \mathcal{TC}(\mathrm{PS}) & \text{must also be satisfiable,} \end{array}$$

where $\mathcal{TC}(\mathrm{PS})$ refers to temporal constraints from the agent's private schedule (PS) of commitments. (Note that the only difference between the two sets of temporal constraints is that the latter contains constraints from the agent's private schedule.)

3 Simple Temporal Networks

The algorithms in this paper manipulate Simple Temporal Networks [2]. In this Section, we highlight the elements of STNs we'll need in subsequent Sections.

An STN $\mathcal{S}$ is a set of $n+1$ time-point variables $\mathcal{TP}(\mathcal{S}) = \{x_0, \ldots, x_n\}$ together with a set $\mathcal{TC}(\mathcal{S})$ of binary temporal constraints on those variables. The temporal constraints have the form: $x_j - x_i \leq \delta$. Note that lower values of δ correspond to *stronger* constraints. The "variable" x_0 represents an arbitrary, fixed reference point on the timeline. In this paper, we use $x_0 = 0$. Unary constraints on $x_1, \ldots, x_n$ are represented as binary constraints involving the special variable x_0, as follows.

$$\begin{array}{lclcl} L_i \leq x_i & \Longrightarrow & 0 - x_i \leq -L_i & \Longrightarrow & x_0 - x_i \leq -L_i \\ x_i \leq U_i & \Longrightarrow & x_i - 0 \leq U_i & \Longrightarrow & x_i - x_0 \leq U_i \end{array}$$

A *solution* to an STN is a set of variable assignments: $\{x_0 = 0, x_1 = v_1, \ldots, x_n = v_n\}$ satisfying all the constraints in $\mathcal{TC}(\mathcal{S})$. Two STNs $\mathcal{S}_1$ and $\mathcal{S}_2$ (over the same set of time point variables) are said to be *equivalent* if they admit identical solution sets.

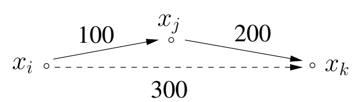
An STN $\mathcal{S}$ may be represented by a weighted, directed graph $G_\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S})$ where the vertices correspond to time points and the edges correspond to binary constraints. More precisely, $V_\mathcal{S} = \mathcal{TP}(\mathcal{S})$ and $E_\mathcal{S} = \{(x_i, \delta, x_j) : (x_j - x_i \leq \delta) \in \mathcal{TC}(\mathcal{S})\}$. In other words, each binary constraint $x_j - x_i \leq \delta$ is represented by a directed edge from x_i to x_j with weight δ. For example, the pair of constraints $x_j - x_i \leq 50$ and $x_i - x_j \leq -12$ are represented by the edges shown below.

$$\begin{tikzcd} x_i \arrow[r, bend left, "50"] & x_j \arrow[l, bend left, "-12"] \end{tikzcd}$$

While the maximum number of explicit constraints in $\mathcal{TC}(\mathcal{S})$ is $n(n+1)$, in practice $\mathcal{TC}(\mathcal{S})$ often contains many fewer constraints. The reason is that even a limited set of *explicit* constraints may give rise to numerous *implicit* constraints. For example, the explicit constraints $x_j - x_i \leq 100$ and $x_k - x_j \leq 200$ combine (using simple arithmetic) to entail the implicit constraint $x_k - x_i \leq 300$, as follows.

$$x_k - x_i \;=\; (x_k - x_j) + (x_j - x_i) \;\leq\; 200 + 100 \;=\; 300$$

In the graphical representation this may be stated as: the "shortest path" from x_i to x_k is no longer than the path from x_i to x_k via x_j:



An STN $\mathcal{S}$ is consistent (i.e., has a solution) if and only if its corresponding graph $G_{\mathcal{S}}$ has no negative cycles [2] (i.e., the path length around any loop is never negative).

The *distance* from x_i to x_j in an STN $\mathcal{S}$ is the *smallest* value δ such that the constraint $x_j - x_i \leq \delta$ is entailed by the constraints in $\mathcal{S}$. (It is appropriate to use the term "distance" for this quantity because it satisfies many of the properties normally associated with a distance function. However, it is important to keep in mind that the distance from x_i to x_j in $\mathcal{S}$ may be negative. In addition, the distance from x_i to x_j is typically not the same as the distance from x_j to x_i.) The distance from x_i to x_j is given by:

$$\text{Distance}(x_i, x_j) \;=\; lub\{(x_j - x_i) \text{ in any solution to } \mathcal{S}\}$$

This quantity is also referred to as the *strongest implicit constraint* from x_i to x_j in $\mathcal{S}$.

The *distance matrix* for an STN $\mathcal{S}$ is an $(n+1)$-by-$(n+1)$ matrix $\mathcal{D}_{\mathcal{S}}$ whose entries record the distance between each pair of time-point variables in $\mathcal{S}$:

$$\mathcal{D}_{\mathcal{S}}(i,j) \;=\; \text{Distance}(x_i, x_j).$$

Since the distance from x_i to x_j in $\mathcal{S}$ corresponds to the length of the shortest path from x_i to x_j in the graph $G_{\mathcal{S}}$, the distance matrix may be computed in polynomial time, for example, using Floyd-Warshall's $O(n^3)$ "all-pairs shortest-path" algorithm [1].

Equivalent STNs have identical distance matrices. In a consistent STN, it is always the case that: $\mathcal{D}_{\mathcal{S}}(i,j) + \mathcal{D}_{\mathcal{S}}(j,i) \;\geq\; 0$. If $\mathcal{D}_{\mathcal{S}}(i,j) + \mathcal{D}_{\mathcal{S}}(j,i) \,=\, 0$, the values x_i and x_j are said to be *rigidly connected* (because their difference is constrained to be a constant). In a consistent STN, it is always possible to add a constraint of the form $x_j - x_i \leq -\mathcal{D}_{\mathcal{S}}(j,i)$ without introducing any inconsistency [2].

4 Solving the Temporal-Constraint-Generation (TCG) Problem

In this section, we define the Temporal-Constraint-Generation Problem and provide an algorithm that solves it. We also provide a proof of correctness for our TCG algorithm.

Inputs:

- An opportunity OPP_α to do an action of type α using a recipe R_α, as in Figure 1;
- Sets of time points $\mathcal{TP}(\text{PS})$ and temporal constraints $\mathcal{TC}(\text{PS})$ representing the agent's private schedule of commitments *prior* to any consideration of OPP_α;
- A set $\Gamma = \{\gamma_1, \ldots, \gamma_k\}$ of subacts being bid on (i.e., a subset of the subacts from the recipe R_α); and
- A set $\mathcal{TC}(\Gamma)$ of additional temporal constraints generated by the agent's private task-integration-scheduling (TIS) algorithm (to ensure the schedulability of the subacts in Γ within the agent's pre-existing schedule of commitments).[1]

Goal: Find the weakest set of constraints $\mathcal{TC}(B)$ to include with the bid B (for the subacts in Γ) such that no matter what additional constraints are added as a result of the auction (according to the rules stipulated earlier), the agent's schedule of existing commitments will be protected.

Fig. 2. The Temporal-Constraint-Generation Problem

4.1 The TCG Problem

The Temporal-Constraint-Generation Problem is defined in Figure 2. Note that the constraints in $\mathcal{TC}(B)$ are required to be both necessary and sufficient for protecting the agent's schedule of pre-existing commitments from any potential auction outcome.

4.2 An Algorithm for Solving the TCG Problem

From the information in Figure 2, two STNs are constructed. The first, called $\mathcal{S}_\alpha$, represents the time points and temporal constraints bundled with the opportunity OPP_α. Thus, $\mathcal{S}_\alpha$ is defined by: $\mathcal{TP}(\mathcal{S}_\alpha) = \mathcal{TP}(\text{OPP}_\alpha)$ and $\mathcal{TC}(\mathcal{S}_\alpha) = \mathcal{TC}(\text{OPP}_\alpha)$.

The second, called $\mathcal{S}_{\text{Agt}}$, represents: (1) the time points and temporal constraints associated with the agent's private schedule of pre-existing commitments, (2) the time points and temporal constraints bundled with the opportunity, and (3) the additional temporal constraints inserted to ensure schedulability of the subacts being bid on. Thus, $\mathcal{S}_{\text{Agt}}$ is defined by:

$$\begin{aligned}\mathcal{TP}(\mathcal{S}_{\text{Agt}}) &= \mathcal{TP}(\text{OPP}_\alpha) \cup \mathcal{TP}(\text{PS})\\ \mathcal{TC}(\mathcal{S}_{\text{Agt}}) &= \mathcal{TC}(\text{OPP}_\alpha) \cup \mathcal{TC}(\text{PS}) \cup \mathcal{TC}(\Gamma).\end{aligned}$$

For convenience, we rename the time-point variables of $\mathcal{S}_\alpha$ and $\mathcal{S}_{\text{Agt}}$, indexing them so that the first $(n+1)$ time points of $\mathcal{S}_{\text{Agt}}$ correspond directly to the time points of $\mathcal{S}_\alpha$:

$$\mathcal{TP}(\mathcal{S}_\alpha) = \{x_0, x_1, \ldots, x_n\} \quad \text{and} \quad \mathcal{TP}(\mathcal{S}_{\text{Agt}}) = \{x_0, x_1, \ldots, x_n; y_1, \ldots, y_k\}$$

[1] One such TIS algorithm is presented in the next Section. A key point is that the TCG algorithm presented in this Section does *not* rely on the particular TIS algorithm used to produce $\mathcal{TC}(\Gamma)$.

We refer to the x_i's as the *opportunity time points*. The y_i's, together with the fixed time point x_0, are referred to as the *non-opportunity time points.* (Note that x_0 is thus both an opportunity and non-opportunity time point.) The y_i's by themselves are referred to as the *strictly* non-opportunity time points.

The temporal constraints in $\mathcal{TC}(\mathrm{OPP}_\alpha)$ may refer only to the opportunity time points. Likewise, the constraints in $\mathcal{TC}(\mathrm{PS})$ may refer only to the non-opportunity time points. However, the constraints in $\mathcal{TC}(\Gamma)$, added to ensure the schedulability of the subacts being bid on, may *link* the opportunity and strictly non-opportunity time points. As a result, the opportunity time points in $\mathcal{S}_{\mathrm{Agt}}$ may be more heavily constrained than their counterparts in $\mathcal{S}_\alpha$. Thus, letting $\mathcal{D}_\alpha$ and $\mathcal{D}_{\mathrm{Agt}}$ be the distance matrices for $\mathcal{S}_\alpha$ and $\mathcal{S}_{\mathrm{Agt}}$, respectively, we have that $\mathcal{D}_{\mathrm{Agt}}(i,j) \ \leq\ \mathcal{D}_\alpha(i,j)$ for each $i, j \leq n$.[2]

The $(n+1)$-by-$(n+1)$ sub-matrix of $\mathcal{D}_{\mathrm{Agt}}$ corresponding to the opportunity time points plays an important role in the algorithm. Thus, we define $\mathcal{D}_{\mathrm{Agt}|\alpha}$ to be this sub-matrix, which corresponds to the first $(n+1)$ rows and columns of $\mathcal{D}_{\mathrm{Agt}}$. It trivially follows that $\mathcal{D}_{\mathrm{Agt}|\alpha}(i,j) \ \leq\ \mathcal{D}_\alpha(i,j)$ for each $i, j \leq n$.

The constraints in $\mathcal{TC}(\Gamma)$ are part of $\mathcal{S}_{\mathrm{Agt}}$. These constraints may link opportunity and strictly non-opportunity time points. However, the constraints to be bundled with the bid may only refer to opportunity time points. We get around this problem by creating a new STN, called $\mathcal{S}_{\mathrm{Agt}|\alpha}$, that contains only the opportunity time points but whose *explicit* temporal constraints are derived one-to-one from the strongest *implicit* constraints represented in $\mathcal{D}_{\mathrm{Agt}|\alpha}$, as follows.[3]

$$
\begin{aligned}
\mathcal{TP}(\mathcal{S}_{\mathrm{Agt}|\alpha}) \ &= \ \mathcal{TP}(\mathrm{OPP}_\alpha) \\
\mathcal{TC}(\mathcal{S}_{\mathrm{Agt}|\alpha}) \ &= \ \{(x_j - x_i \leq \mathcal{D}_{\mathrm{Agt}|\alpha}(i,j)) : i, j \in \{0, 1, \ldots, n\}, i \neq j\}
\end{aligned}
$$

Note that unlike many STNs that contain comparatively few explicit constraints, $\mathcal{S}_{\mathrm{Agt}|\alpha}$ has an explicit constraint between *every* ordered pair of its time points. Note also that corresponding to each edge among opportunity points in $\mathcal{S}_{\mathrm{Agt}}$ there is an edge in $\mathcal{S}_{\mathrm{Agt}|\alpha}$ that is at least as strong. Finally, note that the previously-defined sub-matrix $\mathcal{D}_{\mathrm{Agt}|\alpha}$ is in fact equal to the distance matrix for the just-defined STN $\mathcal{S}_{\mathrm{Agt}|\alpha}$.

We shall see later that bundling the set $\mathcal{TC}(\mathcal{S}_{\mathrm{Agt}|\alpha})$ of temporal constraints with the bid B would be sufficient to protect the agent's schedule of pre-existing commitments from any potential auction outcome; however, we can pare down the set substantially by (carefully) removing redundancies. Before doing so, we explain why we ignore a minor complicating factor known as *rigid components* [9].

A rigid component exists whenever the difference between two (or more) time points is constrained to be some fixed value. Several researchers [9, 14, inter alia] have shown that rigid components are easily dealt with by collapsing each set of rigidly connected time points down to a single point. We employ the same technique but to save

[2] Note that $\mathcal{D}_\alpha$ is an $(n+1)$-by-$(n+1)$ matrix, whereas $\mathcal{D}_{\mathrm{Agt}}$ is an $(n+k+1)$-by-$(n+k+1)$ matrix.

[3] This construction is related to what Dechter et al. call the *distance graph* of an STN [2]. Here, we are creating a distance graph for a *portion* of an STN.

space omit further discussion of it. Henceforth, in this paper, it is assumed that there are no rigid components in the STNs under discussion.

Removing Redundant Edges From $\mathcal{S}_{\mathrm{Agt}|\alpha}$. We employ a notion of *dominance* similar to that used by Tsamardinos et al. [9, 10]. A fundamental difference is that their notion of dominance is restricted by their focus on the notion of the *dispatchability* of an STN whereas ours is not.

We say that an explicit constraint $x_k - x_i \leq \delta$ is *dominated* in an STN $\mathcal{S}$ if removing that edge would result in no change to the distance matrix for $\mathcal{S}$. If $\mathcal{D}_{\mathcal{S}}(i,k) < \delta$, then the explicit constraint $x_k - x_i \leq \delta$ is strictly dominated by a stronger implicit constraint. For the case where $\delta = \mathcal{D}_{\mathcal{S}}(i,k)$, we use the following Claim.

Claim. If $\delta = \mathcal{D}_{\mathcal{S}}(i,k)$, then, under the assumption that there are no rigid components in $\mathcal{S}$, the explicit constraint $x_k - x_i \leq \delta$ is dominated if and only if there exists some other point x_j such that: $\delta = \mathcal{D}_{\mathcal{S}}(i,j) + \mathcal{D}_{\mathcal{S}}(j,k)$ (i.e., if and only if there exists a dominating path from x_i to x_k that does not include that *edge* from x_i to x_k).

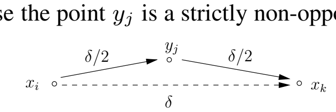
Proof of Claim. If there exists such an x_j, then we can construct a path of length δ by using a shortest path from x_i to x_j followed by a shortest path from x_j to x_k. The absence of rigid components precludes the edge from x_i to x_k being used in either of these shortest paths. Thus, the edge from x_i to x_k does not make a necessary contribution to the STN. On the other hand, if there does not exist such an x_j, then the edge from x_i to x_k represents the only way of constructing a path of length δ from x_i to x_k and hence its removal would alter the distance matrix. ■

The above formula not only provides a convenient way to test for dominance but also tells us that removing a dominated edge will not affect whether some other edge is dominated (because removing a dominated edge does not change the distance matrix and the distance matrix entries are the only things required to determine dominance). Thus, we may use a simple $O(n^3)$ algorithm to locate and remove all dominated edges. (Techniques analogous to those used by Tsamardinos [9] may provide somewhat faster algorithms for tagging dominated edges.)

Let $\mathcal{TC}^u(\mathcal{S}_{\mathrm{Agt}|\alpha})$ be the *undominated* constraints from $\mathcal{S}_{\mathrm{Agt}|\alpha}$.[4] For each (undominated) constraint $x_j - x_i \leq \delta$ in $\mathcal{TC}^u(\mathcal{S}_{\mathrm{Agt}|\alpha})$, if this constraint is *stronger* than the strongest implicit constraint from x_i to x_j in the *opportunity STN* $\mathcal{S}_\alpha$ (i.e., if $\delta < \mathcal{D}_\alpha(i,j)$), then include that constraint with the bid; otherwise, it is not needed. In other words, the set of temporal constraints to bundle with the bid B is given by:

$$\mathcal{TC}(B) = \{(x_j - x_i \leq \delta) \in \mathcal{TC}^u(\mathcal{S}_{\mathrm{Agt}|\alpha}) : \delta < \mathcal{D}_\alpha(i,j)\}.$$

[4] Note that this is *not* the same thing as the undominated constraints among the opportunity time points in $\mathcal{S}_{\mathrm{Agt}}$, as illustrated by the following example which shows an edge (x_i, δ, x_k) that is dominated in $\mathcal{S}_{\mathrm{Agt}}$ but not in $\mathcal{S}_{\mathrm{Agt}|\alpha}$—because the point y_j is a strictly non-opportunity time point and hence does not belong to $\mathcal{S}_{\mathrm{Agt}|\alpha}$:



4.3 Correctness of the TCG Algorithm

Proving the correctness of the TCG algorithm requires showing that the set of constraints $\mathcal{TC}(B)$ is both necessary and sufficient to protect the agent's private schedule of pre-existing commitments from any potential outcome. The sufficiency of the constraints in $\mathcal{TC}(B)$ is given by Theorem 1; the necessity of the constraints in $\mathcal{TC}(B)$ is given by Theorem 2.

Theorem 1. Given the set $\mathcal{TC}(B)$ of temporal constraints generated for a bid B using the TCG algorithm, suppose that the bid B is awarded to the bidder and that the auctioneer inserts an arbitrary set of additional constraints (consistent with the rules of the auction). Then the additional constraints inserted by the auctioneer will be consistent with the agent's schedule of pre-existing commitments (including all constraints concerning the subacts in the bid).

Proof of Theorem 1. Since the theorem assumes that the bid B is awarded to the bidder, we begin by defining $\mathcal{S}_{\alpha^+}$ to be the STN resulting from adding the constraints in $\mathcal{TC}(B)$ to the STN $\mathcal{S}_\alpha$. Let $\mathcal{D}_{\alpha^+}$ be the corresponding distance matrix. Lemmas 1 and 2 below complete the proof. ■

Lemma 1. The distance matrix $\mathcal{D}_{\alpha^+}$ is identical to the distance matrix $\mathcal{D}_{\mathrm{Agt}|\alpha}$. Hence, the corresponding STNs $\mathcal{S}_{\alpha^+}$ and $\mathcal{S}_{\mathrm{Agt}|\alpha}$ are equivalent.

Proof of Lemma 1. Let $x_j - x_i \leq \delta$ be an arbitrary constraint—call it **C**—in $\mathcal{TC}(\mathcal{S}_{\alpha^+}) = \mathcal{TC}(\mathrm{OPP}_\alpha) \cup \mathcal{TC}(B)$. If $\mathbf{C} \in \mathcal{TC}(\mathrm{OPP}_\alpha)$, then $\mathbf{C} \in \mathcal{TC}(\mathcal{S}_{\mathrm{Agt}})$, which implies that $\mathcal{D}_{\mathrm{Agt}}(i,j) \leq \delta$. But $\mathcal{D}_{\mathrm{Agt}}(i,j) = \mathcal{D}_{\mathrm{Agt}|\alpha}(i,j)$, and hence $\mathcal{D}_{\mathrm{Agt}|\alpha}(i,j) \leq \delta$. On the other hand, if $\mathbf{C} \in \mathcal{TC}(B)$, then $\mathbf{C} \in \mathcal{TC}^u(\mathcal{S}_{\mathrm{Agt}|\alpha}) \subseteq \mathcal{TC}(\mathcal{S}_{\mathrm{Agt}|\alpha})$. Thus, in either case, **C** is no stronger than the corresponding (strongest implicit) constraint in $\mathcal{S}_{\mathrm{Agt}|\alpha}$.

In the other direction, let $x_k - x_l \leq \delta'$ be an arbitrary constraint—call it **C**′—in $\mathcal{TC}(\mathcal{S}_{\mathrm{Agt}|\alpha})$. If $\mathbf{C}' \in \mathcal{TC}(B)$, then by the definition of $\mathcal{S}_{\alpha^+}$, $\mathbf{C}' \in \mathcal{TC}(\mathcal{S}_{\alpha^+})$. Otherwise, according to the definition of $\mathcal{TC}(B)$, either $\delta' \geq \mathcal{D}_\alpha(i,j)$ or $\mathbf{C}' \notin \mathcal{TC}^u(\mathcal{S}_{\mathrm{Agt}|\alpha})$. In the first case, some set of constraints in $\mathcal{TC}(\mathrm{OPP}_\alpha) \subseteq \mathcal{TC}(\mathcal{S}_{\alpha^+})$ entail **C**′. In the second case, **C**′ is a dominated constraint in $\mathcal{S}_{\mathrm{Agt}|\alpha}$ which, by definition, means it does not make a necessary contribution to $\mathcal{S}_{\mathrm{Agt}|\alpha}$ and thus may be ignored. In every case, **C**′ is no stronger than the corresponding (strongest implicit) constraint in $\mathcal{S}_{\alpha^+}$. ■

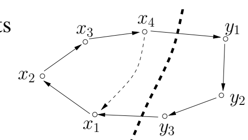
Lemma 2. Let $\mathcal{T}$ be an arbitrary set of temporal constraints over the opportunity time points such that the constraints in $\mathcal{T}$ are *consistent* with the STN $\mathcal{S}_{\alpha^+}$ (i.e., adding that set of constraints to $\mathcal{S}_{\alpha^+}$ would not cause it to become inconsistent). Then that set of constraints is also consistent with $\mathcal{S}_{\mathrm{Agt}}$.

Proof of Lemma 2. Let $\mathcal{S}'_{\alpha^+}$ be the STN resulting from adding the constraints in $\mathcal{T}$ to $\mathcal{S}_{\alpha^+}$. Similarly, let $\mathcal{S}'_{\mathrm{Agt}|\alpha}$ and $\mathcal{S}'_{\mathrm{Agt}}$ be the STNs resulting from adding those same constraints to $\mathcal{S}_{\mathrm{Agt}|\alpha}$ and $\mathcal{S}_{\mathrm{Agt}}$, respectively. By Lemma 1, $\mathcal{S}_{\alpha^+}$ and $\mathcal{S}_{\mathrm{Agt}|\alpha}$ are equivalent; therefore $\mathcal{S}'_{\alpha^+}$ and $\mathcal{S}'_{\mathrm{Agt}|\alpha}$ are equivalent.

Saying that the set of constraints in $\mathcal{T}$ is consistent with $\mathcal{S}_{\alpha+}$ is equivalent to saying that adding the edges corresponding to those constraints would *not* introduce any negative cycles into the graph associated with the STN. In other words, $\mathcal{S}'_{\alpha+}$ does not have any negative cycles (and thus neither does $\mathcal{S}'_{\mathrm{Agt}|\alpha}$, since it is equivalent to $\mathcal{S}'_{\alpha+}$).

Now suppose that, contrary to the statement of the Lemma, the constraints in $\mathcal{T}$ are inconsistent with $\mathcal{S}_{\mathrm{Agt}}$; in other words, suppose that $\mathcal{S}'_{\mathrm{Agt}}$ *does* have a negative cycle.

Case 1. Suppose the negative cycle in $\mathcal{S}'_{\mathrm{Agt}}$ involves at least one strictly non-opportunity time point. We illustrate the method of proof using the following example, leaving it to the reader to carry out the completely straightforward generalization. Consider the cycle shown below (which includes the strictly non-opportunity time points y_1, y_2 and y_3).



Since all of the explicit constraints in $\mathcal{T}$ involve only opportunity time points, all of the edges (i.e., explicit constraints) in $\mathcal{S}'_{\mathrm{Agt}}$ that involve strictly non-opportunity time points are identical to their counterparts in $\mathcal{S}_{\mathrm{Agt}}$. However, the edge from x_4 to x_1 in $\mathcal{S}_{\mathrm{Agt}|\alpha}$ was defined to have the same length as the shortest path from x_4 to x_1 in $\mathcal{S}_{\mathrm{Agt}}$. Thus, if the cycle shown above has negative length in $\mathcal{S}'_{\mathrm{Agt}}$, then the short-circuited cycle (i.e., $x_1 \to x_2 \to x_3 \to x_4 \to x_1$) must have negative length in $\mathcal{S}'_{\mathrm{Agt}}$—although the "edge" from x_4 to x_1 might only be implicit in $\mathcal{S}'_{\mathrm{Agt}}$. Each edge in the short-circuited cycle is either in $\mathcal{T}$ or in $\mathcal{TC}(\mathcal{S}_{\mathrm{Agt}})$; thus it is either in $\mathcal{T}$ or is dominated by an edge in $\mathcal{TC}(\mathcal{S}_{\mathrm{Agt}|\alpha})$. Thus, that short-circuited cycle must have negative length in $\mathcal{S}'_{\mathrm{Agt}|\alpha}$, which, since the "edge" from x_4 to x_1 is a bona fide edge in $\mathcal{S}'_{\mathrm{Agt}|\alpha}$, contradicts that $\mathcal{S}'_{\mathrm{Agt}|\alpha}$ does not have any negative cycles.

Case 2. Suppose the negative cycle in $\mathcal{S}'_{\mathrm{Agt}}$ involves only opportunity time points. The second half of Case 1 treats exactly this case, leading to a contradiction. ■

Theorem 2. If any of the constraints in $\mathcal{TC}(B)$ are weakened, then the resulting constraint set would be insufficient to protect the agent's private schedule of pre-existing commitments from all potential auction results. In other words, some potential auction result would be inconsistent with the agent's pre-existing commitments.

Prior to proving Theorem 2, we develop some notation and some preliminary results.

Let **C** be an arbitrary constraint $x_j - x_i \leq \delta$ in the set $\mathcal{TC}(B)$.

Result 1. Since $\mathbf{C} \in \mathcal{TC}(B)$, we have that $\delta < \mathcal{D}_\alpha(i,j)$ and hence every path or edge from x_i to x_j in $\mathcal{S}_\alpha$ has length strictly greater than δ. Also, since **C** is undominated in $\mathcal{S}_{\mathrm{Agt}|\alpha}$, we have that $\mathcal{D}_{\mathrm{Agt}|\alpha}(i,j) = \delta$. ■

Let $\mathbf{C}_w$ be the constraint $x_j - x_i \leq \delta_w$, where δ_w is an arbitary value such that $\delta < \delta_w$ (i.e., $\mathbf{C}_w$ is a *weaker* constraint than $\mathbf{C}$ over x_i and x_j). Let $\mathcal{TC}_w(B)$ be the same as the set of constraints $\mathcal{TC}(B)$ except with $\mathbf{C}$ replaced by $\mathbf{C}_w$. Similarly, let $\mathcal{S}_{(\mathrm{Agt}|\alpha)_w}$ be the STN that results from replacing $\mathbf{C}$ by $\mathbf{C}_w$ in $\mathcal{TC}(\mathcal{S}_{\mathrm{Agt}|\alpha})$. (Recall that $\mathbf{C} \in \mathcal{TC}(B) \subseteq \mathcal{TC}(\mathcal{S}_{\mathrm{Agt}|\alpha})$.)

Result 2. By definition, since $\mathbf{C}$ is undominated in $\mathcal{S}_{\mathrm{Agt}|\alpha}$, removing $\mathbf{C}$ from $\mathcal{S}_{\mathrm{Agt}|\alpha}$ causes a change to the distance matrix $\mathcal{D}_{\mathrm{Agt}|\alpha}$. However, $\mathcal{S}_{\mathrm{Agt}|\alpha}$ was constructed such that the strongest implicit constraint between each pair of time points was represented by an explicit edge. Therefore, the change to $\mathcal{D}_{\mathrm{Agt}|\alpha}$ caused by removing $\mathbf{C}$ is limited to the (i, j) entry. Thus, replacing $\mathbf{C}$ by $\mathbf{C}_w$ yields: $\delta < \mathcal{D}_{(\mathrm{Agt}|\alpha)_w}(i, j) \leq \delta_w$, where $\mathcal{D}_{(\mathrm{Agt}|\alpha)_w}$ is the distance matrix for $\mathcal{S}_{(\mathrm{Agt}|\alpha)_w}$. ■

Let $\mathcal{S}_{\alpha_w^+}$ be the STN that results from adding the constraints in $\mathcal{TC}_w(B)$ to $\mathcal{S}_\alpha$ (i.e., $\mathcal{S}_{\alpha_w^+}$ is the same as $\mathcal{S}_{\alpha^+}$ except that $\mathbf{C}$ has been replaced by $\mathbf{C}_w$). Let $\mathcal{D}_{\alpha_w^+}$ be the distance matrix for $\mathcal{S}_{\alpha_w^+}$.

Lemma 3. $\mathcal{D}_{\alpha_w^+}(i, j) > \delta$.

Proof of Lemma 3. To the contrary, suppose $\mathcal{D}_{\alpha_w^+}(i, j) \leq \delta$. Then there is some path P from x_i to x_j whose edges are in $\mathcal{TC}(\mathcal{S}_{\alpha_w^+})$ such that: $\mathrm{length}(P) \leq \delta$.

Case 1. Suppose the path P included an *edge* from x_i to x_j. Then P would look like this: $x_i \to \ldots \to x_i \to x_j \to \ldots \to x_j$. Since the length around a loop is always non-negative in a consistent STN (which $\mathcal{S}_{\alpha_w^+}$ is, by assumption), removing the loops from P could only make the length of the path go down or stay the same. Thus, the length of that edge—call it e—from x_i to x_j in $\mathcal{S}_{\alpha_w^+}$ must satisfy: $\mathrm{length}(e) \leq \delta$. However, if $e \in \mathcal{S}_\alpha$ (i.e., if $e \in \mathcal{TC}(\mathrm{OPP}_\alpha)$), Result 1 says that $\mathrm{length}(e) > \delta$. On the other hand, if $e \in \mathcal{TC}_w(B) \subseteq \mathcal{S}_{(\mathrm{Agt}|\alpha)_w}$, Result 2 says that $\mathrm{length}(e) > \delta$. Thus, $e \notin \mathcal{TC}(\mathrm{OPP}_\alpha) \cup \mathcal{TC}_w(B) = \mathcal{TC}(\mathcal{S}_{\alpha_w^+})$, which is a contradiction.

Case 2. Suppose the path P does *not* include an edge from x_i to x_j. Consider an arbitrary edge e on that path. Either $e \in \mathcal{TC}(\mathrm{OPP}_\alpha) \subseteq \mathcal{TC}(\mathcal{S}_{\mathrm{Agt}})$, in which case there is a corresponding edge in $\mathcal{TC}(\mathcal{S}_{Agt|\alpha})$ that is at least as strong as e, or else $e \in \mathcal{TC}_w(B) \subseteq \mathcal{TC}(\mathcal{S}_{\mathrm{Agt}|\alpha})$. In other words, e is either in $\mathcal{TC}(\mathcal{S}_{\mathrm{Agt}|\alpha})$ or is dominated by a corresponding edge in $\mathcal{TC}(\mathcal{S}_{\mathrm{Agt}|\alpha})$. Since e was an arbitrary edge in P, it must be that there is a path in $\mathcal{S}_{Agt|\alpha}$ from x_i to x_j with length less than or equal to δ that does not involve any edge from x_i to x_j, and hence does not involve the edge $\mathbf{C}$. But this contradicts that $\mathbf{C}$ is undominated in $\mathcal{S}_{Agt|\alpha}$. ■

Proof of Theorem 2. Given $\mathcal{S}_{\mathrm{Agt}}$ and $\mathcal{S}_{\alpha_w^+}$ as defined above, we show that there is a constraint $\mathbf{C}^*$ that is consistent with $\mathcal{S}_{\alpha_w^+}$ (and hence is a constraint the auctioneer could legitimately apply), but that is inconsistent with $\mathcal{S}_{\mathrm{Agt}}$ (and hence would be inconsistent with the agent's schedule of pre-existing commitments).

Let $\mathbf{C}^*$ be the constraint $x_i - x_j \leq -\mathcal{D}_{\alpha_w^+}(i, j)$. From Section 3, adding such a constraint is always possible (i.e., it will not affect the consistency of $\mathcal{S}_{\alpha_w^+}$). Thus $\mathbf{C}^*$, which is equivalent to $x_j - x_i \geq \mathcal{D}_{\alpha_w^+}(i, j)$, is a constraint that the auctioneer could legitimately apply. However, by Lemma 3, $\mathcal{D}_{\alpha_w^+}(i, j) > \delta$. Thus, $\mathbf{C}^*$ is inconsistent with the constraint $x_j - x_i \leq \delta$, which is an explicit constraint in $\mathcal{S}_{\text{Agt}|\alpha}$, and hence an implicit constraint in $\mathcal{S}_{\text{Agt}}$. Thus, $\mathbf{C}^*$ is inconsistent with $\mathcal{S}_{\text{Agt}}$. ■

5 The Task-Integration-Scheduling (TIS) Algorithm

In this Section we present a task-integration-scheduling (TIS) algorithm that an agent may use to determine whether a new set of tasks may be merged into its schedule of pre-existing commitments. The algorithm assumes that the actions the agent is already committed to are fully-ordered. The algorithm inserts ordering constraints among the pre-existing commitments and the new tasks being bid on to ensure that the entire collection of tasks forms a fully-ordered list. The algorithm greedily selects the next ordering constraint to add based on a heuristic that measures the resultant change in rigidity in the STN representing the agent's schedule. Recalling that the quantity $\mathcal{D}_{\mathcal{S}}(i,j) + \mathcal{D}_{\mathcal{S}}(j,i)$ is always non-negative, but is zero when x_i and x_j are rigidly connected, we define the (average) *rigidity* of an STN by:

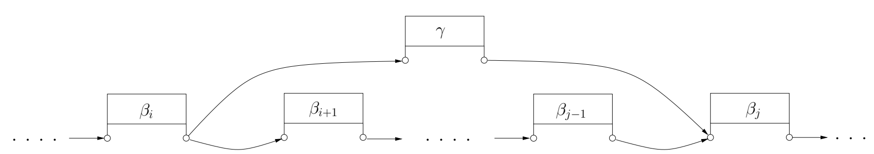
$$\text{Rigidity}(\mathcal{S}) \;=\; \frac{1}{n(n+1)} \sum_{i>j} \frac{1}{1 + \mathcal{D}_{\mathcal{S}}(i,j) + \mathcal{D}_{\mathcal{S}}(j,i)}$$

We first show how the algorithm deals with the case of a single new task. We then show the more general case.

TIS Algorithm: Adding a Single Action.

- Scenario: Agent is already committed to doing actions $\beta_1, \ldots, \beta_m$ which are subject to previously-inserted ordering constraints sufficient to ensure that they are fully-ordered (and hence will not overlap). Agent is considering committing to a new action γ which is *not* yet integrated into the fully-ordered list.

1. Agent adds all time points and temporal constraints associated with γ into its STN. If the distance matrix update shows the STN is still consistent, continue; else fail.
2. Agent uses distance matrix to find the maximal i such that β_i is constrained to occur *before* γ, and the minimal j such that β_j is constrained to occur *after* γ.



   If $j = i + 1$, the list is already fully ordered (with γ constrained to occur between β_i and β_{i+1}); exit with success. Otherwise, continue.
3. Agent computes the increase in the STN's rigidity resulting from adding the ordering constraint that γ occur after β_{i+1}, and also from adding the ordering constraint that γ occur before β_{j-1}. If each constraint would cause the STN to become inconsistent, then fail. Otherwise, add the constraint to the STN that results in the minimal increase in rigidity.
4. Go to Step 2.

If the algorithm succeeds, then the agent may commit to doing γ.

TIS Algorithm: Adding Multiple Actions.

- Scenario: Same as above except that agent is considering committing to a *set* of actions: $\Gamma = \{\gamma_1, \ldots, \gamma_k\}$.

1. Agent adds all time points and temporal constraints associated with $\gamma_1, \ldots, \gamma_k$ into its STN. If the distance matrix update shows that the STN is still consistent, continue; else fail.
2. Same as step 2 above, except that it is done for each γ_s in parallel. If some action gets inserted into the fully-ordered list, then remove that action from Γ and redo this step (because there are now more actions in the fully-ordered list from which the i's and j's are chosen). If all actions are fully-ordered (i.e., $\Gamma = \emptyset$), then exit with success; else continue.
3. Same as step 3 above, except that it is done for each γ_s (i.e., there are $2|\Gamma|$ ordering constraints under consideration). If each ordering constraint would cause the STN to become inconsistent, then fail; otherwise choose the ordering constraint that minimizes the increase in the STN's rigidity.
4. Go to Step 2.

We are currently engaged in testing this algorithm in dynamic domains.

6 Conclusions

This paper presents algorithms that an agent may use to generate bids in a combinatorial auction as part of a group decision-making mechanism. The algorithms manipulate Simple Temporal Networks to generate temporal constraints for bids in such auctions. Although the algorithms were designed for this use, they have broader applicability. They may be used to generate protective temporal constraints any time an agent makes an offer to another agent.

The algorithms might be extended to include disjunctive constraints. However, this would be expected to have substantial computational penalties. As an alternative, agents might express limited amounts of disjunction by using multiple bids.

The techniques for manipulating STNs presented in this paper may also be used to describe the flexibility remaining in an agent's schedule while one or more bids are outstanding. This is important if an agent wants to know whether it can adopt additional commitments while waiting for the auction to complete. In addition, the degree to which a bid impacts the flexibility of an agent's schedule may be used to compute part of the bid's contextually-dependent cost.

When the auction mechanism described in this paper succeeds, the agents may still have substantial amounts of temporal reasoning in front of them. For example, if the auction says that Bill must paint the ceiling between 1:00 and 2:00 p.m. and that Bob must wash the brushes after Bill, Bob can't arbitrarily take on outside commitments that force him to wash the brushes at 1:15 p.m. without checking first with Bill. This sort of problem is related to what Vidal [13] calls *contingent* durations (i.e., durations that are determined by external reality, rather than some agent). Generalizing Vidal's notion of *dynamic controllability* to the distributed, multi-agent scenario described in this paper is the subject of current research.

Finally, as the moment of execution approaches, techniques developed by Tsamardinos et al. [9, 10] may be integrated to efficiently *dispatch* actions to be executed in real time.

Acknowledgments

The research reported in this paper has been supported in part by NSF grants IRI-9618848, IIS-9978343, IRI 95-25915 and CDA-94-01024.

References

1. Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.
2. Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.
3. Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 548–553, 1999.
4. Barbara J. Grosz and Sarit Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86:269–357, 1996.
5. John F. Horty and Martha E. Pollack. Evaluating new options in the context of existing plans. *Artificial Intelligence*, 127(2):199–220, April 2001.
6. Luke Hunsberger and Barbara J. Grosz. A combinatorial auction for collaborative planning. In *Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS-2000)*, pages 151–158. IEEE Computer Society, 2000.
7. Tuomas Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, 1999.
8. Eddie Schwalb and Rina Dechter. Processing disjunctions in temporal constraint networks. *Artificial Intelligence*, 93:29–61, 1997.
9. Ioannis Tsamardinos. Reformulating temporal plans for efficient execution. Master's thesis, University of Pittsburgh, 2000.
10. Ioannis Tsamardinos, Nicola Muscettola, and Paul Morris. Fast transformation of temporal plans for efficient execution. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*. AAAI Press/MIT Press, 1998.
11. Ioannis Tsamardinos, Martha E. Pollack, and John F. Horty. Merging plans with quantitative temporal constraints, temporally extended actions, and conditional branches. In *Proceedings of the Artificial Intelligence Planning and Scheduling 2000 Conference (AIPS 2000)*, 2000.
12. Ioannis Tsamardinos, Martha E. Pollack, and Colleen McCarthy. Efficient solution techniques for the disjunctive temporal problem. Technical Report ISP001-01, Univeristy of Pittsburgh, 2001.
13. Thierry Vidal and Hélène Fargier. Contingent durations in temporal CSPs: from consistency to controllabilities. *Linkoping Electronic Articles in Computer and Information Science, ISSN 1401-9841*, 2(2), 1997. Available from: http://www.ep.liu.se/ea/cis/1997/002/.
14. Rattana Wetprasit and Abdul Sattar. Qualitative and quantitative temporal reasoning with points and durations. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*. AAAI Press/MIT Press, 1998.