Knowledge and games: theory and implementation

Witzel, S.A.

Citation for published version (APA):
Chapter 6

Time constraints in mixed auctions

6.1 Introduction

6.1.1 Motivation

Combinatorial auctions are auction mechanisms where bidders can bid for sets of goods rather than just single items. Despite the fact that solving a combinatorial auction, i.e., finding an allocation of goods to bidders that will maximize the sum of the prices offered by the bidders, is an intractable optimization problem, combinatorial auctions have turned out to be an extremely useful tool with many applications. See the textbook by Cramton et al. [43] for an overview.

In recent work, Cerquides et al. [37] have proposed an extension of the standard combinatorial auction model, called mixed multi-unit combinatorial auctions (or simply mixed auctions for short). In a mixed auction, bidders can offer transformations, consisting of a set of input goods and a set of output goods, rather than just plain goods. Bidding for such a transformation means declaring that one is willing to deliver the specified output goods after having received the input goods, for the price specified by the bid. Solving a mixed auction means choosing a sequence of transformations that satisfies the constraints encoded by the bids, that produces the goods required by the auctioneer from those he holds initially, and that maximizes the amount of money collected from the bidders (or minimizes the amount paid out by the auctioneer). Mixed auctions extend a number of other types of combinatorial auctions: direct auctions, reverse auctions, and combinatorial exchanges. A very promising application of mixed auctions is supply chain formation [148].

In this chapter, we extend the framework of mixed auctions by allowing bidders to specify constraints regarding the times at which they perform the transformations offered in their bids. The motivation for this extension is that, in a complex economy, the bidders (service providers) themselves may need services from others and have their own supply chains, so the bidders may have preferences
over the timing of transformations and over their relative ordering. A notion of
time is already implicit in the original mixed auction framework as far as the
auctioneer is concerned, because he has to build a sequence of transformations, but
this is not the case for the bidders. In this work we seek to redress this imbalance.

6.1.2 Approach

Our contribution covers four types of time constraints:

- **Relative time points**: This is the simplest model. It associates each transfor-
mation with a time point and allows bidders to express constraints regarding
their relative ordering. For example, a bidder may want to offer trans-
formations X and Y, but only under the condition that X be executed before Y.

- **Absolute time points**: This is an extension to the first model and in addition
allows for references to absolute time. For example, a bidder could specify
that transformation X can only be executed at time unit 15, or that it needs
to be executed at most 3 time units after transformation Y.

- **Intervals**: Alternatively, transformations may be associated with time in-
trvals, and bidders can express constraints on whether one transforma-
tion should be executed before, during, or after another transformation,
or whether two transformations need to be executed in overlapping time
intervals.

- **Intervals with absolute durations**: Combining the notions of interval and
absolute time, we can also express constraints on the duration of an interval.
For example, a bidder may want to specify that transformation X requires
at least 5 time units.

These constraint types can be freely mixed to, for instance, express an interval
taking place after a time point.

We also show how to model soft constraints, allowing bidders to offer discounts
in return for satisfying certain time constraints, and how to model the fact that
an auctioneer may sometimes be able to quantify the monetary benefit resulting
from a shorter supply chain. As in the original paper by Cerquides et al. [37], we
are working with the multi-unit variant of mixed auctions, where there may be
several copies of the same good available, but the restriction to single units is
certainly also of interest.

We define a bidding language for expressing combinatorial bids over trans-
formations with associated time constraints, we give a precise definition of the
winner determination problem, and we present an algorithm for solving it by
showing how the problem can be encoded as an integer program. As we will
see, our approach blends nicely into the existing framework of mixed auctions,
6.2. Bidding language

requiring surprisingly few modifications. This facilitates the integration of time constraints with other extensions and optimizations discussed in the literature.

6.1.3 Plan of the chapter

The remainder of this chapter is structured as follows.

Section 6.2 defines a bidding language for mixed auctions with time constraints. We give a formal semantics for this language by showing how any given bid expression is mapped to a valuation function defined over sets of transformations arranged on a time line, and we discuss its expressive power.

In Section 6.3 we formulate the winner determination problem. Building on the original algorithm by Cerquides et al. [37], we show how to model it as an integer program. For relative time constraints, the extension is particularly neat and simple; to accommodate absolute time we show how to overcome a number of conceptual and technical difficulties.

While Sections 6.2 and 6.3 cover the cases of constraints over relative and absolute time points, Section 6.4 presents the extension to time intervals.

Finally, Section 6.5 concludes and briefly discusses related work.

6.2 Bidding language

In this section, we define and discuss a bidding language for mixed auctions with time constraints.

We first define transformations with time points and then valuations, which are functions used to model the preferences of bidders by mapping sets of such transformations with time points to numerical values. We then define the syntax and semantics of our bidding language. The purpose of a bidding language is to encode a bidder’s valuation for transmission to the auctioneer.\(^1\) The basic idea is to use a simple XOR-language, of the kind familiar from standard combinatorial auctions [105, 133] and mixed auctions [37], to express possible combinations of transformations and associated prices, and to use time constraints to impose conditions on the ordering of their execution.

We also add some syntactic sugar to the time constraint language which prima facie seems to have very limited expressive power, and show that it is actually surprisingly expressive. This is achieved by “borrowing” expressive power from the underlying bidding language into the time constraint language. We conclude this section by arguing that our bidding language is fully expressive over the class of all “reasonable” valuations.

---

\(^1\)Bidders may or may not wish to transmit their true valuation. This is an important issue, but one that is entirely irrelevant for the representation of bids, as long as all necessary valuations can be expressed.
6.2.1 Transformations and time points

Let $G$ be the finite set of all types of goods under consideration. A transformation is a pair

$$(I, O) \in \mathbb{N}^G \times \mathbb{N}^G.$$

An agent offering such a transformation declares that, when provided with the multiset of goods $I$, he can deliver the multiset of goods $O$.

When agents submit their bid, they should be able to put restrictions on the time points at which they offer to perform transformations. To this end, we introduce a finite (but big enough) set of time point identifiers $T$. The time points here are to be thought of merely as identifiers, not as variables having an actual value. Agents are negotiating over sets of transformations with time point identifiers

$$D \subset \mathbb{N}^G \times \mathbb{N}^G \times T,$$

which we can write as sets of the form

$$\{(I^1, O^1, \tau^1), \ldots, (I^\ell, O^\ell, \tau^\ell)\}.$$

For example, $\{(\{\}, \{q\}, \tau_1), (\{r\}, \{s\}, \tau_2)\}$ means that the agent in question is able to deliver $q$ without any input at some time point $\tau_1$, and to deliver $s$ if provided with $r$ at some time point $\tau_2$. Note that this is not the same as $\{(\{r\}, \{q, s\}, \tau)\}$. In the former case, if another agent can produce $r$ from $q$, we can get $s$ from nothing; in the latter case this does not work.

We assume that each time point identifier is used at most once, even across bidders, and thus indeed identifies one particular transformation of one particular bidder.

Note that this does not allow two transformations to be offered at the same time point identifier. If a bidder wants to do that, he can instead offer a single transformation with combined $I$ and $O$.

6.2.2 Valuations

Since bidders may have preferences over the time points at which they perform transformations, we first define a time line $\Sigma$ of transformations to be a finite sequence of transformations and “clock ticks” $c$. The latter stand for time passing without any transformations allocated to the bidder. That is,

$$\Sigma \in (\mathbb{N}^G \times \mathbb{N}^G \cup \{c\})^*,$$

where $^*$ denotes words as defined in Chapter 2, Section 2.2.

A valuation $v$ maps a time line $\Sigma$ to a real number $p$. Intuitively, $v(\Sigma) = p$ means that an agent with valuation $v$ is willing to make a payment of $p$ for getting the task of performing transformations according to the time line $\Sigma$ (intuitively, $p$
is usually negative, meaning that the agent is actually being paid for performing the transformations. We write \( v(\Sigma) = \bot \) if \( v \) is undefined for \( \Sigma \), i.e., the agent would be unable to accept the corresponding deal.

For example, the valuation \( v \) given by

\[
\begin{align*}
  v(((\{\text{oven, dough}\}, \{\text{oven, cake}\}))) &= -2 \\
  v(((\{\text{oven, dough}\}, \{\text{oven, cake}\})); (\{\}, \{\text{bread}\})) &= -3 \\
  v((\{\}, \{\text{bread}\}); (\{\text{oven, dough}\}, \{\text{oven, cake}\}))) &= \bot
\end{align*}
\]

expresses that for two dollars I could produce a cake if given an oven and dough, also returning the oven; for another dollar I could do the same and afterwards give you a bread without any input; but I could not do it the other way round (which should make you wonder whether you might be giving away too much dough in the first case).

We say that a valuation \( v \) uses relative time if for all \( \Sigma \) we have that \( v(\Sigma) = v(\Sigma - c) \), where \( \Sigma - c \) stands for \( \Sigma \) with all clock ticks \( c \) removed. Otherwise \( v \) is said to use absolute time.

That is, in the context of relative time, valuations depend only on the relative ordering of the transformations, while with absolute time valuations may depend on the absolute time point at which a transformation occurs within a given sequence. In the latter case, we interpret each step in the sequence as a time unit (one second, one day, one week, ...).

Note that the fact that we are dealing with a sequence agrees with the assumption above that no two transformations can be offered at the same time point identifier. Again, from the bidder’s side this can be overcome by simply combining the two transformations that he would like to offer concurrently. In Section 6.3, we get back to this issue from the auctioneer’s point of view, where it has slightly more serious conceptual consequences. Note also that all transformations are assumed to have the same duration of one time unit; in Section 6.4 we look at how to deal with transformations of variable durations.

### 6.2.3 Bids

We are now ready to define the bidding language. An atomic bid \( \text{bid}(D, p) \) specifies a finite set of finite transformations with time points and a price. Intuitively, submitting such a bid expresses the bidder’s willingness to perform the given transformations for price \( p \).

Constraints on the time points at which these transformations should be scheduled will be added on top of this bid. But before we turn to that, we define the rest of our bidding language. We restrict ourselves to the xor-language, which is known to be fully expressive for standard (single-unit) combinatorial auctions [105], and which, for multi-unit mixed auctions, has been shown to subsume many other intuitively useful language constructs and to fully express a
class of valuations comprising most (if not all) intuitively sensible ones [37]. In particular, it is not difficult to extend the auction framework presented in this chapter to also handle the so-called or-operator [105, 37].

So we consider complex bids in xor normal form,

\[ \text{Bid} = \text{bid}(D_1, p_1) \text{xor} \ldots \text{xor} \text{bid}(D_n, p_n), \]

with the intuitive interpretation that the bidder is willing to perform at most one of the \( D_j \) and pay the associated \( p_j \). Note again that the used time point identifiers are unique, also across the \( D_j \).

### 6.2.4 Time constraints

We now define two variants of a time constraint language. The atomic constraints for relative time are all of the form \( \tau < \tau' \). For absolute time, we allow the following (where \( \tau, \tau' \in \mathcal{T}, \xi, \xi' \in \mathbb{N} \)):

\[
\begin{align*}
\tau &= \xi \\
\tau < \xi \\
\tau > \xi \\
\tau + \xi < \tau' + \xi' \\
\tau + \xi &= \tau' + \xi'
\end{align*}
\]

The reason why we distinguish between the two variants, even though the latter may seem to subsume the former, is that the former is conceptually “safer”, at least in our formalism which is based on sequences: Concepts like “before” and “after” always make intuitive sense, while with absolute time we have to agree on the time unit of a sequence step and take timing issues into account. This is not always without difficulties, as noted in several places in this chapter; see, e.g., the discussion in Section 6.3.3.

As an example, the atomic bid with time constraint

\[
\text{BID} \{ \{\{\text{oven}, \text{dough}\}, \{\text{oven}, \text{cake}\}, \tau_1\}, \\
\{\{\}, \{\text{bread}\}, \tau_2\}, -3\}
\]

\[\tau_1 < \tau_2\]

expresses the above fact that I am willing to sell you the bread only after I have sold you the cake.

We allow time constraint formulas of the form

\[ \varphi = \gamma_1 \land \ldots \land \gamma_\nu \]

with atomic constraints \( \gamma_i \). A bidder submits a bid \( \text{Bid} \) together with a time constraint formula \( \varphi \) (the atomic constraints of which refer to time point identifiers occurring in \( \text{Bid} \)). Intuitively, this expresses that he is willing to perform according to \( \text{Bid} \), but only under the condition that \( \varphi \) is satisfied.

This condition is hard: the bidder will only accept if it is met. In Section 6.2.6 we give a method to express soft time constraints (associated with costs), and we also show that disjunctions over time constraints can be expressed.
6.2. Bidding language

6.2.5 Semantics

Syntactically, we are thus dealing with complex bids with time points together with constraint formulas over these time points:

$$\text{BID}(D_1, p_1) \text{ XOR } \ldots \text{ XOR } \text{BID}(D_n, p_n) \quad \gamma_1 \land \ldots \land \gamma_\nu.$$  

In order to make the intuitive meanings explained above explicit, we now specify a formal semantics. In the following, let $\Sigma$ be a time line (clock ticks allowed), let $\tau, \tau' \in \mathcal{T}$, $\xi, \xi' \in \mathbb{N}$, and let $\varphi$ and $\varphi'$ be time constraint formulas. Let $\tau \in \Sigma$ denote the fact that $\tau$ is associated with some transformation in $\Sigma$, and let $\Sigma(\tau)$ denote the sequence number (starting from 1) of the transformation associated with $\tau$, if $\tau \in \Sigma$. For clarity, we may include the time point identifiers in the sequence. For example, if $\Sigma = ((I^1, O^1, \tau^1); \ldots)$, then $\tau^1 \in \Sigma$ and $\Sigma(\tau^1) = 1$.

We inductively define a satisfaction relation $\models$ as follows:

$$\Sigma \models \tau \circ \xi \quad \text{iff} \quad \tau \not\in \Sigma \text{ or } \Sigma(\tau) \circ \xi,$$

$$\Sigma \models \tau + \xi < \tau' + \xi' \quad \text{iff} \quad \tau' \not\in \Sigma \text{ or } \tau \in \Sigma \text{ and } \Sigma(\tau) + \xi < \Sigma(\tau') + \xi'$$

$$\Sigma \models \varphi \land \varphi' \quad \text{iff} \quad \Sigma \models \varphi \text{ and } \Sigma \models \varphi'.$$

Relative time constraints are covered by omitting the $+ \xi$ and $+ \xi'$, and $\tau + \xi = \tau' + \xi'$ is defined to be an abbreviation for

$$\tau + \xi < \tau' + (\xi' + 1) \land \tau' + \xi' < \tau + (\xi + 1).$$

According to this semantics, time constraints over time point identifiers that are fully included in $\Sigma$ are interpreted as expected. Constraints over time point identifiers not in $\Sigma$ are simply ignored (they are always satisfied). Note that the choice of semantics for constraints such as $\tau < \tau'$ is somewhat arbitrary in case only one of the time points being compared occurs in $\Sigma$. As an intuitive justification for this detail of the semantics, $\tau$ may be thought of as a precondition for $\tau'$, for instance, because some outcome of the first transformation is needed for the second. In the case of $\tau + \xi = \tau' + \xi'$, this has the effect that either none of the two mentioned transformations is included, or both are and must have the specified distance. However, the exact details do not matter all that much, since the bidding language allows specifying in all detail which transformations can occur together and which cannot.

Using a more technical justification, we prefer this interpretation of constraints because it turns out that it has a straightforward translation to integer constraints, which we need for the implementation described in Section 6.3.3.

We say that a set of transformations $\mathcal{D}$ permits $\Sigma$ if $\Sigma$ consists of exactly the transformations in $\mathcal{D}$ (and optionally clock ticks). In contrast to this definition,
in [37], different assumptions concerning free disposal are distinguished. Informally, free disposal means that participants are always happy to accept more goods than they strictly require; if they really have absolutely no use for them (or are even bothered by them), they can dispose of them for free. Free disposal makes intuitive sense for most every-day goods; however it is not as appropriate for certain “goods” like nuclear waste. We do not delve further into this issue here and continue without any free-disposal assumptions; however, we emphasize that this is purely for the sake of clarity, and these assumptions could be built in with only minuscule changes. In particular, the issue of free disposal as far as bidders are concerned has no impact on the winner determination problem discussed in Section 6.3; it only affects the definition of the semantics of the bidding language.

We now define the valuation expressed by an atomic bid $Bid = (D, p)$ together with a time constraint formula $\varphi$ to be:

$$v_{Bid, \varphi}(\Sigma) = \begin{cases} p & \text{if } D \text{ permits } \Sigma \text{ and } \Sigma \models \varphi \\ \bot & \text{otherwise.} \end{cases}$$

Accordingly, the valuation expressed by a complex bid $Bid = \text{xor}_{j=1}^n Bid_j$ together with a time constraint formula $\varphi$ is (interpreting $\bot$ as $-\infty$):

$$v_{Bid, \varphi}(\Sigma) = \max\{v_{Bid_j, \varphi}(\Sigma) \mid j \in \{1, \ldots, n\}\}.$$ 

That is, out of all the applicable atomic bids $Bid_j$ (i.e., where $v_{Bid_j, \varphi}(\Sigma) \neq \bot$), the auctioneer is allowed to choose the one giving him maximum profit.

### 6.2.6 Syntactic sugar

The time constraint language seems rather limited at first glance, allowing only conjunctions of atomic constraints. However, it turns out that additional expressive power can be “borrowed” from the bidding language. We discuss two extensions to the time constraint language, which can be rewritten into the core language by using additional bid expressions.

**Disjunctive time constraints.** If a bidder wants to offer $(I^1, O^1)$, $(I^2, O^2)$ and $(I^3, O^3)$ with price $p$, where the third should take place after the second or after the first, it seems natural to write

$$\text{bid}((I^1, O^1, \tau^1), (I^2, O^2, \tau^2), (I^3, O^3, \tau^3), p)$$

$$\tau^1 < \tau^3 \lor \tau^2 < \tau^3,$$

with the obvious meaning of the disjunction $\lor$. This is not directly possible in our time constraint language. However, it can be translated into

$$\text{bid}((I^1, O^1, \vartheta^1), (I^2, O^2, \vartheta^2), (I^3, O^3, \vartheta^3), p)$$

$$\text{xor bid}((I^1, O^1, \zeta^1), (I^2, O^2, \zeta^2), (I^3, O^3, \zeta^3), p)$$

$$\vartheta^1 < \vartheta^3 \land \zeta^2 < \zeta^3.$$
The choice which of the disjuncts to satisfy has been moved into the bid expression and is determined by picking one of the atomic bids. Since their variables are disjoint, this pick makes one conjunct of the transformed time constraint formula vacuously true, while the other conjunct still needs to be satisfied. Since it may perfectly well happen that both of the original disjuncts are satisfied in the end, disjunction is the right notion here, even though it is translated into an \textit{XOR} of bids.

For a general formulation, we allow a bid expression in \textit{XOR} normal form together with a time constraint formula in disjunctive normal form:

\[
\text{XOR}_{j=1}^{n} \text{Bid}_j \vee_{i=1}^{\nu} \varphi_i,
\]

where the \(\varphi_i\) are standard (conjunctive) time constraint formulas. The bidder can thus conveniently express, e.g., several alternative partial orders over his transformations. Let now \(\sigma_i\) for \(i \in \{1, \ldots, \nu\}\) be substitutions (with disjoint ranges), each mapping all variables occurring in the bid to fresh (used nowhere else) ones. The resulting translation is

\[
\text{XOR}_{i=1}^{\nu} \text{XOR}_{j=1}^{n} \text{Bid}_j \sigma_i \wedge_{i=1}^{\nu} \varphi_i \sigma_i.
\]

This may seem surprising, because in the original formulation the auctioneer has two choices (which of the time constraint disjuncts to satisfy and which bid to pick), and in the translation he loses the choice among the time constraints. However, in return he gets the freedom to choose over the outer \textit{XOR}. As illustrated in the example above, this boils down to choosing one of the fresh variable spaces, which corresponds to choosing one of the original disjuncts. All the rest of the transformed time conjunction does not have any effect, because it talks about variables which do not occur in the chosen sub-bid. The auctioneer then proceeds to pick a bid from the inner \textit{XOR}, just as before.

\textbf{Soft time constraints.} Soft constraints are constraints with associated costs. Intuitively, such a constraint does not have to be satisfied, but if it is, then the price of the bid is modified by the given cost (usually a discount to the auctioneer).

For example, if a bidder wants to bid on \((I^1, O^1)\) and \((I^2, O^2)\) for price \(p\) and offer a discount, i.e., raise his bid by \(\delta\), if he gets to do the first before the second, then he could write

\[
\text{BID}((I^1, O^1, \tau^1), (I^2, O^2, \tau^2), p) \quad (\tau^1 < \tau^2, \delta).
\]
Again, this expression can be translated:

\[
\text{bid}(\{(I^1, O^1, \vartheta^1), (I^2, O^2, \vartheta^2)\}, p) \\
\text{xor bid}(\{(I^1, O^1, \zeta^1), (I^2, O^2, \zeta^2)\}, p + \delta) \\
\zeta^1 < \zeta^2.
\]

In general, we can allow discounts depending on time constraint formulas rather than only on single constraints, and also the possibility to specify several alternative such discount options. Only one of these options should be applicable (possible combinations can be expressed as separate options), which is why we use XOR to denote the alternatives, analogously to the XOR of the bidding language. We thus consider a bid expression in XOR normal form together with a soft time constraint formula,

\[
\text{xor}_{j=1}^{n}(D_j, p_j) \\
\text{xor}_{i=1}^{\nu}(\varphi_i, \delta_i),
\]

where the \(\varphi_i\) are time constraint formulas and the \(\delta_i \in \mathbb{R}\) (possibly 0). Again, let \(\sigma_i\) for \(i \in \{1, \ldots, \nu\}\) be substitutions, each mapping all variables occurring in the bid to fresh ones. The resulting translation then is

\[
\text{xor}_{i=1}^{\nu} \text{xor}_{j=1}^{n}(D_j \sigma_i, p_j + \delta_i) \\
\bigwedge_{i=1}^{\nu} \varphi_i \sigma_i.
\]

Note that the two translations we discussed are completely analogous. The difference is that the “exclusive” behavior of the bidding language XOR carries over to soft constraints in the sense that at most one discount will have an effect, while it does not carry over to disjunctive constraints, since multiple disjuncts may well end up being satisfied. Note also that the transformations can be combined. For example, a soft time constraint could have a disjunctive condition.

The blowup resulting from the transformations is straightforwardly seen to be linear in the number of disjuncts or of alternative discounts, respectively.

### 6.2.7 Expressive power

We say a valuation is finite if it has a finite domain (i.e., yields non-\(\bot\) for finitely many time lines only) consisting of finite sequences of finite transformations (i.e., with finite input and output). Having seen how versatile time constraints really are, it may not be surprising that XOR bids with time constraints are fully expressive for finite valuations.

Concretely, XOR bids with relative time constraints can express all finite valuations that use relative time; XOR bids with absolute time constraints can express all finite valuations. To see this, simply take an XOR bid with one atomic bid \(\text{bid}(D, p)\) for each \(\Sigma\) in the domain of \(v\), with \(D\) set to permit \(\Sigma\) and \(p\) set
to $v(\Sigma)$, and impose the order corresponding to $\Sigma$ using time constraints (note that there may be several atomic bids with the same transformations in $D$, but different time points).

For a discussion of the expressive power of mixed-auction bidding languages (without time constraints but) with free disposal we refer to [37].

6.3 Winner determination

We now study the winner determination problem (WDP). This is the problem, faced by the auctioneer, of determining which transformations to award to which bidder, so as to maximize (minimize) the sum of payments collected (made), given the bids of the bidders expressed in our bidding language. This may be interpreted as computing a solution that maximizes revenue for the auctioneer, or social welfare for the collective of bidders (if we interpret prices offered as reflecting bidder utility). Note that we are interested in the algorithmic aspects of the WDP. Game-theoretical considerations, such as how to devise a more sophisticated pricing rule that would induce bidders to bid truthfully, are orthogonal to the algorithmic problem addressed here. (We briefly comment on mechanism design issues in Section 6.5, but this is not the topic of this chapter.)

For symmetry between bidders and auctioneer, we do not assume free disposal for the auctioneer (just like for the bidders), i.e., he does not want to end up with any goods except the required ones. Note, however, that the formulations are easily adapted to allow free disposal (and we point out the necessary changes along the way).

After formulating the WDP, we give an integer program [137] solving it and discuss some further topics. We aim at keeping the descriptions short and focus on the changes compared to the version from [37]. For more background and intuitive explanations, see there. The advantage of this approach, besides showing how few modifications are necessary and thereby how powerful the original framework already is, is that it is modular and can (hopefully) be combined without too much effort with other extensions or optimizations.

6.3.1 WDP with time constraints

The input to the WDP consists of

- a bid expression $Bid_i$ in xor normal form together with a conjunction of time constraints $\varphi_i$, for each bidder $i$;
- a multiset $U_m$ of goods the auctioneer holds in the beginning;
- and a multiset $U_{out}$ of goods the auctioneer wants to end up with.
Let \( \text{Bid}_{ij} \) denote the \( j \)th atomic bid \( \text{BID}(\mathcal{D}_{ij}, p_{ij}) \) occurring within \( \text{Bid}_i \), let \( t_{ijk} \) be a unique label for the \( k \)th transformation in \( \mathcal{D}_{ij} \) (for some arbitrary but fixed ordering of \( \mathcal{D}_{ij} \)), and let \( \tau_{ijk} \) be the time point identifier associated with transformation \( t_{ijk} \). Let \( (\mathcal{I}_{ijk}, \mathcal{O}_{ijk}) \) be the actual transformation labelled with \( t_{ijk} \). Finally, let \( T \) be the set of all \( t_{ijk} \).

An **allocation sequence** \( \Sigma \) resembles the time line we introduced before, but can only contain transformations actually offered by some bidder, and each one at most once. That is, \( \Sigma \) now is a permutation of a subset of \( T \), possibly interspersed with clock ticks \( c \).

We write \( t_{ijk} \in \Sigma \) to say that the \( k \)th transformation in the \( j \)th atomic bid of bidder \( i \) has been selected, and we write \( \Sigma(t_{ijk}) \) to denote the sequence number of \( t_{ijk} \) (starting from 1) if \( t_{ijk} \in \Sigma \).

By \( \Sigma_i \) we denote the **projection** of \( \Sigma \) to bidder \( i \), that is, \( \Sigma \) with each \( t_{ijk} \) replaced by \( (\mathcal{I}_{ijk}, \mathcal{O}_{ijk}, \tau_{ijk}) \) and all \( t_{i'jk} \) replaced by \( c \) for \( i' \neq i \).

By \( (\mathcal{T}^m, \mathcal{O}^m) \) we denote the \( m \)th transformation in \( \Sigma \). Thus, we have two ways of referring to a selected transformation: by its position in the received bids \( (t_{ijk}) \) and by its position \( m \) in the allocation sequence.

Given \( \Sigma \), we can inductively define the bundle of goods held by the auctioneer after each step (let \( g \in G \) be any good, and let \( \mathcal{M}^0 = \mathcal{U}_m \)):

\[
\mathcal{M}^m(g) = \mathcal{M}^{m-1}(g) + \mathcal{O}^m(g) - \mathcal{I}^m(g) \quad (6.1)
\]

under the condition that

\[
\mathcal{M}^{m-1}(g) \geq \mathcal{I}^m(g). \quad (6.2)
\]

Given a multiset \( \mathcal{U}_m \) of goods available to the auctioneer, a multiset \( \mathcal{U}_{out} \) of goods required by the auctioneer, and a set of bids \( \text{Bid}_i \) with time constraints \( \varphi_i \), an allocation sequence \( \Sigma \) is a **valid solution** if:

(i) For each bidder \( i \), some \( \mathcal{D}_{ij} \) permits \( \Sigma_i \), or \( \Sigma_i \in \{c\}^* \).

(ii) For each bidder \( i \), \( \Sigma_i \models \varphi_i \).

(iii) Equations (6.1) and (6.2) hold for each transformation \( (\mathcal{T}^m, \mathcal{O}^m) \in \Sigma \) and each good \( g \in G \).

(iv) For each good \( g \in G \), \( \mathcal{M}^{[\Sigma]}(g) = \mathcal{U}_{out}(g) \).

The **revenue** for the auctioneer associated with a valid solution \( \Sigma \) is the sum of the prices of the selected atomic bids:

\[
\sum \{ p_{ij} | \exists k : t_{ijk} \in \Sigma \}.
\]

---

2 Given a multiset \( S \in \mathbb{N}^G \) and an item \( g \in G \), we write \( S(g) \) to denote the number of copies of \( g \) in \( S \).

3 Replace \( = \) by \( \geq \) to model free disposal.
Given multisets $U_{in}$ and $U_{out}$ of initial and required goods and a set of bids with time constraints, the \textbf{winner determination problem} (WDP) consists in finding a valid solution that maximizes the auctioneer’s revenue.

We now show how to solve the WDP using integer programming (IP). To this end, we first review the original formulation from [37] and then discuss the modifications required for dealing with time constraints.

### 6.3.2 Original integer program

In this part, we closely follow [37]. The main issue is to decide, for each offered transformation, whether it should be selected for the solution sequence, and if so, at which position. Thus, we define a set of binary decision variables $x_{ijk} \in \{0, 1\}$, each of which takes on value 1 if and only if the transformation $t_{ijk}$ is selected at the $m$th position of the solution sequence.

The position number $m$ ranges from 1 to an upper bound $M$ on the solution sequence length. For the time being, we take $M = |T|$, the overall number of transformations, accommodating all sequences that can be formed using only transformations (and not clock ticks).

Further, $i$ ranges over all bidders; $j$ ranges for each bidder $i$ from 1 to the number of atomic bids submitted by $i$; and $k$ ranges for each atomic bid $j$ of bidder $i$ from 1 to the number of transformations in that bid.

We use the following auxiliary binary decision variables: $x^m$ takes on value 1 if and only if any transformation is selected at the $m$th position; $x_{ijk}$ takes on value 1 if and only if transformation $t_{ijk}$ is selected at all; and $x_{ij}$ takes on value 1 if and only if any of the transformations in the $j$th atomic bid of bidder $i$ are selected.

The following set of constraints define a valid solution \textit{without} taking time constraints into account (i.e., neglecting (ii) in the valid solution definition above):

1. Select either all or no transformations from an atomic bid (cf. (i) above):
   \[ x_{ij} = x_{ijk} \quad (\forall ijk) \]

2. Select at most one atomic bid from each XOR normal form bid (cf. (i) above):
   \[ \sum_j x_{ij} \leq 1 \quad (\forall i) \]

3. Select each transformation at most for one position:
   \[ x_{ijk} = \sum_m x^m_{ijk} \quad (\forall ijk) \]
(4) For each position, select at most one transformation:

\[ x^m = \sum_{ijk} x^m_{ijk} \quad (\forall m) \]

(5) There should be no gaps in the sequence:

\[ x^m \geq x^{m+1} \quad (\forall m) \]

Note that this is strictly speaking not required; indeed we drop this constraint later on in order to allow clock ticks between transformations.

(6) Treating each \( M^m(g) \) as an integer decision variable, ensure that necessary input goods are available (cf. (iii) above):

\[
M^m(g) = U_{in}(g) + \sum_{\ell=1}^{m} \sum_{ijk} x^\ell_{ijk} \cdot (O_{ijk}(g) - I_{ijk}(g))
\]

\[
M^m(g) \geq \sum_{ijk} x^m_{ijk} \cdot I_{ijk}(g) \quad (\forall g \in G, \forall m)
\]

(7) In the end, the auctioneer should have the bundle \( U_{out} \) (cf. (iv) above):

\[ M^M(g) = U_{out}(g) \quad (\forall g \in G) \]

Solving the WDP now amounts to solving the following integer program:

\[
\max \sum_{ij} x_{ij} \cdot p_{ij}, \quad \text{subject to constraints (1)--(7)}
\]

A valid solution is then obtained by making transformation \( t_{ijk} \) the \( m \)th element of the solution sequence \( \Sigma \) exactly when \( x^m_{ijk} = 1 \).

6.3.3 Modified integer program

To implement time constraint handling (thus obeying (ii) in the definition of valid solution given above), we first introduce an additional set of auxiliary binary decision variables \( y^m_{ijk} \in \{0, 1\} \), taking on value 1 if and only if transformation \( t_{ijk} \) is selected at the \( m \)th position or earlier in the solution sequence. This can be achieved by adding the following constraint:

(8) \( y^m_{ijk} \) should be 1 iff \( t_{ijk} \in \Sigma \) and \( \Sigma(t_{ijk}) \leq m \):

\[
y^m_{ijk} = y^{m-1}_{ijk} + x^m_{ijk} \quad (\forall ijk m),
\]

with \( y^0_{ijk} = 0 \).

We now give implementations for our two variants of time constraints.

\[4\text{With free disposal, } \geq \text{ would become } \geq.\]
6.3. Winner determination

**Relative time.** Each bidder \( i \)'s time constraint formula is a conjunction of atomic time constraints, and all bidders' time constraints need to be satisfied. The following set of integer constraints takes care of this.

\((9a)\) For each \( \tau_{ijk} < \tau_{ij'k'} \) occurring in \( \wedge_i \varphi_i \):

\[
y^{m}_{ijk} \geq y^{m+1}_{ij'k'} \quad (\forall m).
\]

In accordance with the time constraint semantics, if neither \( t_{ijk} \) nor \( t_{ij'k'} \) occurs in the solution sequence, this requirement is vacuously satisfied since both sides stay 0. If \( t_{ij'k'} \) does occur, then \( y^{m}_{ij'k'} \) will become 1 at some point \( m \). In this case, the requirement boils down to \( y^{m-1}_{ijk} \) being 1 as well, so \( t_{ijk} \) must have occurred already.

Solving the WDP with relative time constraints thus amounts to the same optimization as before, but subject to constraints (1)–(8) and (9a).

**Absolute time.** In order to have an absolute notion of time, we need some way of mapping points of a possible solution sequence to an absolute time line. The simplest way is to interpret each sequence point itself as a time unit (a minute, a day, a week, . . . ), and this is the approach we take.

Before giving the formalization, we need to discuss some conceptual details. If we interpret steps in the sequence as absolute time units, some issues arise which did not matter before.

Firstly, while it may be acceptable to break time down into discrete steps of equal duration, it is not so easy to defend that any transformation that can possibly be offered should have exactly that duration.

Secondly, there is no reason why the auctioneer should wait for one transformation to end before commissioning the next transformation, which may be offered by a different, idling bidder, unless the output of the former is needed as input to the latter.

To some extent, these issues can be addressed by a purely conceptual extension presented in Section 6.4. However, we leave it to future work to design frameworks which handle time in a more flexible way and truly optimize for effective parallelizations (which is a research field on its own). For our purposes, we simply assume that the auctioneer is busy when he is delivering or receiving goods of some particular transformation, and cannot deal with several bidders simultaneously.

To start the formalization, first of all we drop constraint (5). As mentioned, it is not strictly speaking necessary anyway, and since now the bidders can refer to arbitrary absolute time points, we actually might have to accept gaps in the sequence.

Now a technical issue arises: The length of possible solution sequences is no longer bounded by \( |T| \). While it may be possible to find a correct bound by looking at all numbers occurring in the bidders’ time constraints, we settle for a
different solution: The auctioneer manually specifies \( M \), the maximum length of the solution sequence.

At first glance this seems like a pure loss of generality; however the auctioneer may profit from having some control over the size of the WDP he has to solve, and he can always iterate over different values for \( M \) in his search for a good solution. Economically speaking, it also makes sense that the auctioneer wants some control over the length of his supply chain, rather than allowing an arbitrary length. Indeed, he might have graded preferences over the time his supply chain takes; in Section 6.3.4 we show how he can accomplish this, turning the ostensible loss of generality into a feature.

We now give the integer constraints for handling absolute time constraints.

(9b) For each \( \tau_{ijk} + \xi < \tau_{ij'k'} + \xi' \) occurring in \( \bigwedge_i \varphi_i \):
\[
y_{m+\xi'} \geq y_{m+\xi+1} (\forall m);
\]
for each \( \tau_{ijk} + \xi = \tau_{ij'k'} + \xi' \) occurring in \( \bigwedge_i \varphi_i \):
\[
y_{m+\xi'} = y_{m+\xi} (\forall m)
\]
(10) For each \( \tau_{ijk} \circ \xi \), with \( \circ \in \{=,\,<,\,>\} \), occurring in \( \bigwedge_i \varphi_i \):
\[
x_{m} = 0 \ (\forall m \neq \xi).
\]

Constraint (9b) requires some explanation. First of all, note that (9a), the version for relative time, is covered as a special case. As indicated by the semantics, the absolute time variant is thus an extension of the relative time variant. Secondly, note that the second half of (9b) can be obtained from the first half if interpreted as an abbreviation, as in Section 6.2.5. Now consider the case where \( \xi' = 0 \). Intuitively speaking, the time constraint then says that \( t_{ijk} \) must take place at least \( \xi + 1 \) time steps before \( t_{ij'k'} \). That is, whenever \( t_{ij'k'} \) is selected, \( t_{ijk} \) must already have been selected for at least \( \xi + 1 \) time steps. In terms of the integer program, this means that, for all positions \( m \), \( y_{ij'k'}^{m+\xi-1} \) must be 0 unless \( y_{ijk}^{m-\xi} \) was already 1. Now it is only a small step to the formulation in (9b).

Solving the WDP with absolute time constraints amounts to the same optimization as before, but subject to constraints (1)–(4), (6)–(8), (9b) and (10).

A valid solution is then obtained by making transformation \( t_{ijk} \) the \( m \)th element of the solution sequence \( \Sigma \) if and only if \( x_{ijk}^{m} = 1 \), and using a clock tick \( c \) as \( m \)th element when there is no \( x_{ijk}^{m} \) which equals 1 (i.e., when \( x^{m} = 0 \)).

### 6.3.4 Valuation for the auctioneer

Given that we decided to require the auctioneer to specify the maximum length \( M \) of the solution sequence (for the absolute-time variant of the framework), we
6.3. Winner determination

may also want to give him the possibility to express more detailed preferences over durations. This turns out to be achievable in a neat way, also enabling the auctioneer to express graded preferences over the final bundles.

So let us assume that the auctioneer derives a certain value from a given supply chain, depending on its overall duration and on its outcome, the bundle of goods he owns in the end. Note that this discussion assumes absolute time; with relative time, preferences over *durations* do not make much sense, but the results can easily be adjusted to only model preferences over *outcomes*.

We thus assume the auctioneer’s valuation is a function

\[ u : \mathbb{N} \times \mathbb{N}^G \to \mathbb{R} \cup \{\bot\}, \]

mapping duration/outcome pairs to a value or \( \bot \), meaning the duration/outcome pair is not acceptable. This valuation can be incorporated into the WDP in the following way.

After receiving the bids, the auctioneer decides on a maximum duration \( M \) and creates an additional bid under an unused bidder identity:

\[
\text{\textsc{xor}} \left\{ (m, \mathcal{U}) | u(m, \mathcal{U}) \neq \bot \right\} \textsc{bid}\left(\{\mathcal{U}, \{\textcircled{\text{⊘}}\}, \tau_m, \mathcal{U}\}, u(m, \mathcal{U})\right),
\]

where \( \textcircled{\text{⊘}} \) is a special token that does not occur as a good in any other bid, together with time constraints:

\[
\bigwedge_{\{ (m, \mathcal{U}) | u(m, \mathcal{U}) \neq \bot \}} \tau_m, \mathcal{U} = m.
\]

The transformations in this bid are to be thought of as terminal transformations: they denote the possible time points and outcomes at which a solution sequence may end, and the associated values for the auctioneer. The idea is that using this method, the auctioneer’s valuation can be expressed with almost no change to the integer program.

Let us look at the requirements needed for the auctioneer’s valuation to work.

- The terminal transformations should only be used at the respective intended positions in the sequence; this is ensured by the given time constraints.
- At *most* one of them should be used; this is ensured by the \textsc{xor} (and strictly speaking also follows from the last point below).\(^5\)
- At *least* one\(^6\) of them should be used; this can be ensured by setting \( \mathcal{U}_{\text{out}} = \{\textcircled{\text{⊘}}\} \).

\(^5\)Even more strictly speaking, it also follows from the next requirement and the fact that we assume no free disposal; we include it nevertheless for conceptual clarity and in order to accommodate a possible free disposal assumption.

\(^6\)This could read “*exactly* one”, but again, we want to accommodate a possible free disposal assumption.
The unique terminal transformation which is used should indeed constitute the end of the solution sequence.\footnote{As a last remark, this requirement could be dropped if we \textit{did} assume free disposal and all bids’ prices were positive.}

For this last point, we need an additional integer constraint:

\begin{equation}
    x_{ijk}^{m+1} \leq 1 - y_{1j'k'}^m \quad (\forall ijkj'k'm)
\end{equation}

\((-1\) being the auctioneer’s “bidder identity”).

While the remaining requirements could also be encoded more directly and more efficiently into the integer program, for clarity we here restrict ourselves to this version using the high-level features of the bidding language.

Many further extensions and optimizations along these lines are conceivable. We do not try to exhaust them here, but sketch only one example. The auctioneer might want to extract some goods $U$ from the supply chain by some intermediate time point $\xi$, not necessarily at its end. To express this, he can add a transformation $(U, \{\diamond\}, \tau)$ with time constraint $\tau < \xi + 1$ to his bid, and add $\diamond$ to $U_{\text{out}}$. Dropping constraint (11) for this transformation, he makes it non-terminal. He can also make this a soft requirement by including another transformation that yields $\diamond$ from no input and attaching appropriate prices to the corresponding bids.

### 6.3.5 Computational complexity

The (decision problem underlying the) WDP for mixed auctions with time constraints is NP-complete. NP-hardness follows from NP-hardness of the WDP for standard combinatorial auctions \cite{127}. NP-membership follows from the fact that the validity of a given allocation sequence can clearly be verified in polynomial time. That is, in terms of (abstract) computational complexity, the integration of time constraints does not have too much of an impact when compared to the original mixed-auction model \cite{37}.

This is also the reason why time constraints seem to fall into place so easily in our case, which may be somewhat unexpected given the amount of research dedicated to handling them (see, e.g., \cite{138}): Most of those research efforts are directed towards tractability, which in our context does not have such a high priority since even without any optimizations, time constraints do not increase the complexity already inherent in the underlying framework.

Consequently, not much has changed with respect to the complexity of the integer programming formulation. While there is room for optimizations, the number of variables we introduce is of the same order as in the original formulation: $O(n^2)$, where $n$ is the number of transformations occurring in the bids submitted.
6.4. Intervals

The most recent work on winner determination algorithms for mixed auctions has tried to reduce the number of decision variables needed so as to improve performance [67, 107]. Due to the modular nature of our approach, we are optimistic that it will be possible to take advantage of these optimizations and integrate them with the extensions for handling time constraints presented here. In this chapter, our focus has been on presenting the basic model of mixed auctions with time constraints, and on demonstrating the feasibility of the approach.

6.4 Intervals

As discussed before, it is desirable to allow transformations to overlap or take place during other transformations, and to allow transformations to have different durations (meaningful mostly in the variant with absolute time).

One step towards this goal, which can be expressed in our framework without any modifications, is to allow for a transformation to specify an interval during which it takes place, rather than only a single time point identifier.

Interval handling is pure syntactic sugar in our framework. To represent a transformation that takes place during an interval, we include two time point identifiers: start time and end time. Internally, intervals are reduced to two transformations with single time points and an appropriate time constraint:

\[(I, O, [\tau, \tau']) \leadsto (I, \emptyset, \tau), (\emptyset, O, \tau')\]

Since the replacement takes place within a single atomic bid, it is guaranteed that either both the start and end transformations will be selected, or neither. That is, the interval transformation remains intact.

The usual interval relations (see the interval calculus by Allen [2]; due to sequentiality we consider only the strict relations) can be defined as macros yielding standard time constraints:

\[\begin{align*}
[\tau_1, \tau'_1] \text{ BEFORE } [\tau_2, \tau'_2] & \leadsto \tau_1' < \tau_2 \\
[\tau_1, \tau'_1] \text{ OVERLAPS } [\tau_2, \tau'_2] & \leadsto \tau_1 < \tau_2 \land \tau_1' < \tau_2' \\
[\tau_1, \tau'_1] \text{ DURING } [\tau_2, \tau'_2] & \leadsto \tau_2 < \tau_1 \land \tau_1' < \tau_2
\end{align*}\]

With absolute time, absolute restrictions on the durations can also be implemented:

\[\begin{align*}
duration([\tau, \tau']) > \xi & \leadsto \tau + \xi < \tau' \\
duration([\tau, \tau']) < \xi & \leadsto \tau' < \tau + \xi \\
duration([\tau, \tau']) = \xi & \leadsto \tau' = \tau + \xi
\end{align*}\]

Note that expressions like \(duration(\cdot) > duration(\cdot)\) are not so straightforwardly expressible in our framework, but arguably also much less useful in the context of specifying bids.
6.5 Conclusions and related work

We have presented an extension to the existing framework of mixed multi-unit combinatorial auctions [37], enabling bidders to impose time constraints on the transformations they offer.

In the original framework, the auctioneer is free to schedule the offered transformations in any way suitable to achieve his desired outcome, while bidders are left with no control over this process. Our work redresses this asymmetry, thus representing an important step towards a more realistic model of supply chain formation, where bidders themselves may have supply chains or other factors restricting the possible schedules for performing certain transformations.

Starting from a very basic core language for expressing time constraints, we have given various extensions, many purely syntactic, showing the somewhat unexpected power inherent to the core language.

We have also extended the integer program given in [37] to handle time constraints. Our extensions are modular in a way that will facilitate combining them with other extensions and optimizations for mixed auctions, such as [67, 107].

6.5.1 Related work

Time constraints have been applied to different types of combinatorial auctions in the literature. For example, Hunsberger and Grosz [81] extend an existing algorithm for winner determination in combinatorial auctions to allow precedence constraints when bidding on roles in a prescribed action plan (“recipe”). Collins [42] permits relative time constraints in a combinatorial reverse auction over combinations of tasks, and the efficiency of various approaches to solving the winner determination problem is tested.

Auction frameworks involving time have also been fruitfully applied to problems of distributed scheduling. In the work of Wellman et al. [152], time constraints do not enter separately, but rather time slots are the actual objects being auctioned, and game-theoretic properties and mechanism design issues are discussed.

While it would be interesting to examine whether the insights about efficiency and alternative approaches to handling time could be applied to our framework, the roles, tasks, and time slots being auctioned in those contributions are “atomic”, and the formulations and results do not easily translate to transformations in the context of mixed auctions.

6.5.2 Possible extensions

Concerning mechanism design, the remarks by Cerquides et al. [37] still apply. The bottom line is that, with finite valuations, the incentive-compatibility of the Vickrey-Clarke-Groves (VCG) mechanism carries over from standard combinatorial auctions to mixed multi-unit combinatorial auctions with time constraints. It
is a question of independent interest, whether and how this can be extended to non-finite valuations when still allowing only finite bids.

Other topics for future work include the exact interplay between the various syntactic extensions we have given, defining a uniform general language, and determining whether some of the features can be implemented in more direct (and efficient) ways than through the translation to the core language used in this work. The same holds for the underlying bidding language, where operators such as OR may be executed more efficiently than through translation to XOR. Finally, an empirical analysis needs to be performed, including testing and optimizing our integer program.