Timed Negotiations
Sundararaman Akshay, Blaise Genest, Loïc Hélouët, Sharvik Mital

To cite this version:
Sundararaman Akshay, Blaise Genest, Loïc Hélouët, Sharvik Mital. Timed Negotiations. 2019. hal-02337887

HAL Id: hal-02337887
https://inria.hal.science/hal-02337887v1
Preprint submitted on 29 Oct 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Timed Negotiations

S. Akshay\textsuperscript{1}, Blaise Genest\textsuperscript{2}, Loic Helouet\textsuperscript{3}, and Sharvik Mital\textsuperscript{1}

\textsuperscript{1} IIT Bombay, Mumbai, India \{akshayss,sharky\}@cse.iitb.ac.in
\textsuperscript{2} Univ Rennes, CNRS, IRISA, Rennes, France blaise.genest@irisa.fr
\textsuperscript{3} Univ Rennes, INRIA, Rennes, France loic.helouet@inria.fr

Abstract. Negotiations were introduced in \cite{6} as a model for concurrent systems with multiparty decisions. What is very appealing with negotiations is that it is one of the very few non-trivial concurrent models where several interesting problems, such as soundness, i.e. absence of deadlocks, can be solved in PTIME \cite{2}. In this paper, we introduce the model of timed negotiations and consider the problem of computing the minimum and the maximum execution time of a negotiation. The latter can be solved using the algorithm of \cite{10} computing costs in negotiations, but surprisingly minimum execution time cannot.

In this paper, we propose new algorithms to compute both minimum and maximum execution time, that work in much more general classes of negotiations than \cite{10}, that only considered sound and deterministic negotiations. Further, we uncover the precise complexities of these questions, ranging from PTIME to $\Delta_2^P$-complete. In particular, we show that computing the minimum execution time is more complex than computing the maximum execution time in most classes of negotiations we consider.

1 Introduction

Distributed systems are notoriously difficult to analyze, mainly due to the explosion of the number of configurations that have to be considered to answer even simple questions. A challenging task is then to propose models on which analysis can be performed with tractable complexities, preferably within polynomial time. Free choice Petri nets are a classical model of distributed systems that allow for efficient verification, in particular when the nets are 1-safe \cite{5, 4}.

Recently, \cite{6} introduced a new model called negotiations for workflows and business processes. A negotiation describes how processes interact in a distributed system: a subset of processes in a node of the system take a synchronous decisions among several outcomes. The effect of this outcome sends contributing processes to a new set of nodes. The execution of a negotiation ends when processes reach a final configuration. Negotiations can be deterministic (once an outcome is fixed, each process knows its unique successor node) or not.

Negotiations are an interesting model since several properties can be decided with a reasonable complexity. The question of soundness, i.e., deadlock-freedom: whether from every reachable configuration one can reach a final configuration, is PSPACE-complete. However, for deterministic negotiations, it can be decided
in PTIME [7]. The decision procedure uses reduction rules. Reduction techniques were originally proposed for Petri nets [1, 8, 12, 17]. The main idea is to define transformations rules that produce a model of smaller size w.r.t. the original model, while preserving the property under analysis. In the context of negotiations, [7, 2] proposed a sound and complete set of soundness-preserving reduction rules and algorithms to apply these rules efficiently. The question of soundness for deterministic negotiations was revisited in [9] and showed NLOGSPACE-complete using anti patterns instead of reduction rules. Further, they show that the PTIME result holds even when relaxing determinism [9]. Negotiation games have also been considered to decide whether one particular process can force termination of a negotiation. While this question is EXPTIME complete in general, for sound and deterministic negotiations, it becomes PTIME [13].

While it is natural to consider cost or time in negotiations (e.g. think of the Brexit negotiation where time is of the essence, and which we model as running example in this paper), the original model of negotiations proposed by [6] is only qualitative. Recently, [10] has proposed a framework to associate costs to the executions of negotiations, and adapt a static analysis technique based on reduction rules to compute end-to-end cost functions that are not sensitive to scheduling of concurrent nodes. For sound and deterministic negotiations, the end-to-end cost can be computed in $O(n.(C + n))$, where $n$ is the size of the negotiation and $C$ the time needed to compute the cost of an execution. Requiring soundness or determinism seem perfectly reasonable, but asking sound and deterministic negotiations is too restrictive: it prevents a process from waiting for decisions of other processes to know how to proceed.

In this paper, we revisit time in negotiations. We attach time intervals to outcomes of nodes. We want to compute maximal and minimal executions times, for negotiations that are not necessarily sound and deterministic. Since we are interested in minimal and maximal execution time, cycles in negotiations can be either bypassed or lead to infinite maximal time. Hence, we restrict this study to acyclic negotiations. Notice that time can be modeled as a cost, following [10], and the maximal execution time of a sound and deterministic negotiation can be computed in PTIME using the algorithm from [10]. Surprisingly however, we give an example (Example 3) for which the minimal execution time cannot be computed in PTIME by this algorithm.

The first contribution of the paper shows that reachability (whether at least one run of a negotiation terminates) is NP-complete, already for (untimed) deterministic acyclic negotiations. This implies that computing minimal or maximal execution time for deterministic (but unsound) acyclic negotiations cannot be done in PTIME (unless NP=PTIME). We characterize precisely the complexities of different decision variants (threshold, equality, etc.), with complexities ranging from (co-)NP-complete to $\Delta^p_2$.

We thus turn to negotiations that are sound but not necessarily deterministic. Our second contribution is a new algorithm, not based on reduction rules, to compute the maximal execution time in PTIME for sound negotiations. It is based on computing the maximal execution time of critical paths in the nego-
tations. However, we show that minimal execution time cannot be computed in PTIME for sound negotiations (unless NP=PTIME): deciding whether the minimal execution time is lower than $T$ is NP-complete, even for $T$ given in unary, using a reduction from a Bin packing problem. This shows that minimal execution time is harder to compute than maximal execution time.

Our third contribution consists in defining a class in which the minimal execution time can be computed in (pseudo) PTIME. To do so, we define the class of $k$-layered negotiations, for $k$ fixed, that is negotiations where nodes can be organized into layers of at most $k$ nodes at the same depth. These negotiations can be executed without remembering more than $k$ nodes at a time. In this case, we show that computing the maximal execution time is PTIME, even if the negotiation is neither deterministic nor sound. The algorithm, not based on reduction rules, uses the $k$-layer restriction in order to navigate in the negotiation while considering only a polynomial number of configurations. For minimal execution time, we provide a pseudo PTIME algorithm, that is PTIME if constants are given in unary. Finally, we show that computing the maximal execution time of a $k$-layered negotiation is less than $T$ is NP-complete, when $T$ is given in binary. We show this by reducing from a Knapsack problem, yet again emphasizing that the minimal execution time of a negotiation is harder to compute than its maximal execution time.

This paper is organized as follows. Section 2 introduces the key ingredients of negotiations, determinism and soundness, known results in the untimed setting, and provides our running example modeling the Brexit negotiation. Section 3 introduces time in negotiations, gives a semantics to this new model, and formalizes several decision problems on maximal and minimal durations of runs in timed negotiations. We recall the main results of the paper in Section 4. Then, Section 5 considers timed execution problems for deterministic negotiations, Section 6 for sound negotiations, and section 7 for layered negotiations. Proof details for the last three technical sections are given in the Appendices A, B and C.

2 Negotiations: Definitions and Brexit example

In this section, we recall the definition of negotiations, of some subclasses (acyclic and deterministic), as well as important problems (soundness and reachability).

**Definition 1 (Negotiation [6,10]).** A negotiation over a finite set of processes $P$ is a tuple $\mathcal{N} = (N,n_0,n_f,X)$, where:

- $N$ is a finite set of nodes. Each node is a pair $n = (P_n,R_n)$ where $P_n \subseteq P$ is a non empty set of processes participating in node $n$, and $R_n$ is a finite set of outcomes of node $n$ (also called results), with $R_n = \{r_f\}$. We denote by $R$ the union of all outcomes of nodes in $N$.
- $n_0$ is the first node of the negotiation and $n_f$ is the final node. Every process in $P$ participates in both $n_0$ and $n_f$.
- For all $n \in N$, $X_n : P_n \times R_n \rightarrow 2^N$ is a map defining the transition relation from node $n$, with $X_n(p,r) = \emptyset$ iff $n = n_f$, $r = r_f$. We denote $\mathcal{X} : N \times P \times R \rightarrow 2^N$.
Fig. 1. A (sound but non-deterministic) negotiation modeling Brexit.

$R \rightarrow 2^N$ the partial map defined on $\bigcup_{n \in N} \{\{n\} \times P_n \times R_n\}$, with $X(n, p, a) = X_n(p, a)$ for all $p, a$.

Intuitively, at a node $n = (P_n, R_n)$ in a negotiation, all processes of $P_n$ have to agree on a common outcome $r$ chosen from $R_n$. Once this outcome $r$ is chosen, every process $p \in P_n$ is ready to move to any node prescribed by $X(n, p, r)$. A new node $m$ can only start when all processes of $P_m$ are ready to move to $m$.

Example 1. We illustrate negotiations by considering a simplified model of the Brexit negotiation, see Figure 1. There are 3 processes, $P = \{EU, PM, Pa\}$. At first EU decides whether or not to enforce a backstop in any deal (outcome backstop) or not (outcome no-backstop). In the meantime, PM decides to prorog Pa, and Pa can choose or not to appeal to court (outcome court/no court). If it goes to court, then PM and Pa will take some time in court (c-meet, defend), before PM can meet EU to agree on a deal. Otherwise, Pa goes to recess, and PM can meet EU directly. Once EU and PM agreed on a deal, PM tries to convince Pa to vote the deal. The final outcome is whether the deal is voted, or whether Brexit is delayed.

Definition 2 (Deterministic negotiations). A process $p \in P$ is deterministic iff, for every $n \in N$ and every outcome $r$ of $n$, $X(n, p, r)$ is a singleton. A negotiation is deterministic iff all its processes are deterministic. It is weakly non-deterministic [9] (called weakly deterministic in [2]) iff, for every node $n$, one of the processes in $P_n$ is deterministic. Last, it is very weakly non-deterministic [9] (called weakly deterministic in [6]) iff, for every $n$, every $p \in P_n$ and every outcome $r$ of $n$, there exists a deterministic process $q$ such that $q \in P_{n'}$ for every $n' \in X(n, p, r)$.
In deterministic negotiations, once an outcome is chosen, each process knows the next node it will be involved in. In (very)-weakly non-deterministic negotiations, the next node might depend upon the outcome chosen in other nodes by other processes. However, once the outcomes have been chosen for all current nodes, there is only one next node possible for each process. Observe that the class of deterministic negotiations is isomorphic to the class of free choice workflow nets [10]. Coming back to example 1, the Brexit negotiation is non-deterministic, because process PM is non-deterministic. Indeed, consider outcomes c-meet: it allows two nodes, according to whether the backstop is enforced or not, which is a decision taken by process EU. However, the Brexit negotiation is very weakly non-deterministic, as the other processes are deterministic.

**Semantics:** A configuration [2] of a negotiation is a mapping $M : P \rightarrow 2^N$. Intuitively, it tells for each process $p$ the set $M(p)$ of nodes $p$ is ready to engage in. The semantics of a negotiation is defined in terms of moves from a configuration to the next one. The initial $M_0$ and final $M_f$ configurations, are given by $M_0(p) = \{n_0\}$ and $M_f(p) = \emptyset$ respectively for every process $p \in P$. A configuration $M$ enables node $n$ if $n \in M(p)$ for every $p \in P_n$. When $n$ is enabled, a decision at node $n$ can occur, and the participants at this node choose an outcome $r \in R_n$. The occurrence of $(n, r)$ produces the configuration $M'$ given by $M'(p) = X(n, p, r)$ for every $p \in P_n$ and $M'(p) = M(p)$ for remaining processes in $P \setminus P_n$.

Moving from $M$ to $M'$ after choosing $(n, r)$ is called a step, denoted $M \xrightarrow{n,r} M'$. A run of $N$ is a sequence $(n_1, r_1), (n_2, r_2)\ldots(n_k, r_k)$ such that there is a sequence of configurations $M_0, M_1, \ldots, M_k$ and every $(n_i, r_i)$ is a step between $M_{i−1}$ and $M_i$. A run starting from the initial configuration and ending in the final configuration is called a final run. By definition, its last step is $(n_f, r_f)$.

An important class of negotiations in the context of timed negotiations are acyclic negotiations, where infinite sequence of steps are impossible:

**Definition 3 (Acyclic negotiations).** The graph of a negotiation $N$ is the labeled graph $G_N = (V, E)$ where $V = N$, and $E = \{(n, (p, r), n') \mid n' \in X(n, p, r)\}$, with pairs of the form $(p, r)$ being the labels. A negotiation is acyclic iff its graph is acyclic. We denote by $\text{Paths}(G_N)$ the set of paths in the graph of a negotiation. These paths are of the form $\pi = (n_0, (p_0, r_0), n_1) \ldots (n_{k-1}, (p_k, r_k), n_k)$.

The Brexit negotiation of Fig.1 is an example of acyclic negotiation. Despite their apparent simplicity, negotiations may express involved behaviors as shown with the Brexit example. Indeed two important questions in this setting are whether there is some way to reach a final node in the negotiation from (i) the initial node and (ii) any reachable node in the negotiation.

**Definition 4 (Soundness and Reachability).**

1. A negotiation is sound iff every run from the initial configuration can be extended to a final run. The problem of soundness is to check if a given negotiation is sound.
2. The problem of reachability asks if a given negotiation has a final run.
Notice that the Brexit negotiation of Fig.1 is sound (but not deterministic).
It seems hard to preserve the important features of this negotiation while being
both sound and deterministic. The problem of soundness has received consider-
able attention. We summarize the results about soundness in the next theorem:

**Theorem 1.** Determining whether a negotiation is sound is PSPACE-Complete.
For (very-)weakly non-deterministic negotiations, it is co-NP-complete [9]. For
acyclic negotiations, it is in DP and co-NP-Hard [6]. Determining whether an
acyclic weakly non-deterministic negotiation is sound is in PTIME [2, 9]. Fi-

nally, deciding soundness for deterministic negotiation is NLOGSPACE-complete [9].

Checking reachability is NP-complete, even for deterministic acyclic negoti-
ations (surprisingly, we did not find this result stated before in the literature):

**Proposition 1.** Reachability is NP-complete for acyclic negotiations, even if
the negotiation is deterministic.

**Proof (sketch).** One can easily guess a run of size \( \leq |N| \) in polynomial time, and
verify if it reaches \( n_f \), which gives the inclusion in NP. The hardness part comes
from a reduction from 3-CNF-SAT that can be found in the proof of Theorem 3.
\( \square \)

**k-Layered Acyclic Negotiations**
We introduce a new class of negotiations which has good algorithmic properties,
namely \( k \)-layered acyclic negotiations, for \( k \) fixed. Roughly speaking, nodes of a
\( k \)-layered acyclic negotiations can be arranged in layers, and these layers contain
at most \( k \) nodes. Before giving a formal definition, we need to define the depth
of nodes in \( N \).

First, a *path* in a negotiation is a sequence of nodes \( n_0 \ldots n_\ell \) such that for
all \( i \in \{1, \ldots, \ell - 1\} \), there exists \( p_i, r_i \) with \( n_{i+1} \in \mathcal{X}(n_i, p_i, r_i) \). The *length* of a
path \( n_0, \ldots, n_\ell \) is \( \ell \). The *depth* \( \text{depth}(n) \) of a node \( n \) is the maximal length of a
path from \( n_0 \) to \( n \) (recall that \( N \) is acyclic, so this number is always finite).

**Definition 5.** An acyclic negotiation is layered if for all node \( n \), every path
reaching \( n \) has length \( \text{depth}(n) \). An acyclic negotiation is \( k \)-layered if it is layered,
and for all \( \ell \in \mathbb{N} \), there are at most \( k \) nodes at depth \( \ell \).

The Brexit example of Fig.1 is 6-layered. Notice that a layered negotiation
is necessarily \( k \)-layered for some \( k \leq |N| - 2 \). Note also that we can always
transform an acyclic negotiation \( N \) into a layered acyclic negotiation \( N' \), by
adding dummy nodes: for every node \( m \in \mathcal{X}(n, p, r) \) with \( \text{depth}(m) > \text{depth}(n) + 1 \),
we can add several nodes \( n_1, \ldots, n_\ell \) with \( \ell = \text{depth}(m) - (\text{depth}(n) + 1) \), and
processes \( P_{n_i} = \{p\} \). We compute a new relation \( \mathcal{X}' \) such that \( \mathcal{X}'(n, p, r) = \{n_1\} \), \( \mathcal{X}'(n_\ell, p, r) = \{m\} \) and for every \( i \in 1..\ell - 1 \), \( \mathcal{X}'(n_i, p, r) = n_{i+1} \). This
transformation is polynomial: the resulting negotiation is of size up to \( |N| \times
|\mathcal{X}| \times |P| \). The proof of the following Theorem can be found in appendix C.

**Theorem 2.** Let \( k \in \mathbb{N}^+ \). Checking reachability or soundness for a \( k \)-layered
acyclic negotiation \( N \) can be done in PTIME.
3 Timed Negotiations

In many negotiations, time is an important feature to take into account. For instance, in the Brexit example, with an initial node starting at the beginning of September 2019, there are 9 weeks to pass a deal till the 31st October deadline.

We extend negotiations by introducing timing constraints on outcomes of nodes, inspired by time Petri nets [15] and by the notion of negotiations with costs [10]. We use time intervals to specify lower and upper bounds for the duration of negotiations. More precisely, we attach time intervals to pairs \((n,r)\) where \(n\) is a node and \(r\) an outcome. In the rest of the paper, we denote by \(\mathcal{I}\) the set of intervals with endpoints that are non-negative integers or \(\infty\). For convenience we only use closed intervals in this paper (except for \(\infty\)), but the results we show can also be extended to open intervals with some notational overhead. Intuitively, outcome \(r\) can be taken at a node \(n\) with associated time interval \([a,b]\) only after \(a\) time units have elapsed from the time all processes contributing to \(n\) are ready to engage in \(n\), and at most \(b\) time units later.

Definition 6. A timed negotiation is a pair \((N,\gamma)\) where \(N\) is a negotiation, and \(\gamma : N \times R \to \mathcal{I}\) associates an interval to each pair \((n,r)\) of node and outcome such that \(r \in R_n\). For a given node \(n\) and outcome \(r\), we denote by \(\gamma^-(n,r)\) (resp. \(\gamma^+(n,r)\)) the lower bound (resp. the upper bound) of \(\gamma(n,r)\).

Example 2. In the Brexit example, we define the following timed constraints \(\gamma\). We only specify the outcome names, as the timing only depends upon them.

Backstop and no-backstop both take between 1 and 2 weeks: \(\gamma(\text{backstop}) = \gamma(\text{no-backstop}) = [1,2]\). In case of no-court, recess takes 5 weeks \(\gamma(\text{recess}) = [5,5]\), and PM can meet EU immediately \(\gamma(\text{meet}) = [0,0]\). In case of court action, PM needs to spend 2 weeks in court \(\gamma(\text{continue}) = [2,2]\), and depending on the court delay and decision, Pa needs between 3 (court overrules recess) to 5 (court confirms recess) weeks, \(\gamma(\text{decision}) = [3,5]\). Agreeing on a deal can take anywhere from 2 weeks to 2 years (104 weeks): \(\gamma(\text{deal agreed}) = [2,104]\) - some would say infinite time is even possible! It needs more time with the backstop, \(\gamma(\text{deal w/backstop}) = [5,104]\). All other outcomes are assumed to be immediate, i.e., associated with \([0,0]\).

Semantics: A timed valuation is a map \(\mu : P \to \mathbb{R}^{\geq 0}\) that associates a non-negative real value to every process. A timed configuration is a pair \((M,\mu)\) where \(M\) is a configuration and \(\mu\) a timed valuation. There is a timed step from \((M,\mu)\) to \((M',\mu')\), denoted \((M,\mu) \xrightarrow{(n,r)} (M',\mu')\), if (i) \(M \xrightarrow{(n,r)} M'\), (ii) \(p \notin P_n\) implies \(\mu'(p) = \mu(p)\) (iii) \(p \in P_n\) implies \((\mu'(p) - \max_{p' \in P_n} \mu(p')) \in \gamma(n,r)\).

Intuitively a timed step \((M,\mu) \xrightarrow{(n,r)} (M',\mu')\) depicts a decision taken at node \(n\), and how long each process of \(P_n\) waited in that node before taking decision \((n,r)\). The last process engaged in \(n\) must wait for a duration contained in \(\gamma(n,r)\). However, other processes may spend a time greater than \(\gamma^+(n,r)\).

A timed run is a sequence of steps \(\rho = (M_1,\mu_1) \xrightarrow{\delta_1} (M_2,\mu_2) \ldots (M_k,\mu_k)\) where each \((M_i,\mu_i) \xrightarrow{\delta_i} (M_{i+1},\mu_{i+1})\) is a timed step. It is final if \(M_k = M_f\). Its execution time \(\delta(\rho)\) is defined as \(\delta(\rho) = \max_{p \in P} \mu_k(p)\).
Notice that we only attached timing to processes, not to individual steps. With our definition of runs, timing on steps may not be monotonous (i.e., non-decreasing) along the run, while timing on processes is. Viewed by the lens of concurrent systems, the timing is monotonous on the partial orders of the system rather than the linearization. It is not hard to restrict paths, if necessary, to have a monotonous timing on steps as well. In this paper, we are only interested in execution time, which does not depend on the linearization considered.

Given a timed negotiation $N$, we can now define the minimum and maximum execution time, which correspond to optimistic or pessimistic views:

**Definition 7.** Let $N$ be a timed negotiation. Its minimum execution time, denoted $\text{mintime}(N)$ is the minimal $\delta(\rho)$ over all final timed run $\rho$ of $N$. We define the maximal execution time $\text{maxtime}(N)$ of $N$ similarly.

Given $T \in \mathbb{N}$, the main problems we consider in this paper are the following:

- The mintime problem, i.e., do we have $\text{mintime}(N) \leq T$?
  - In other words, does there exist a final timed run $\rho$ with $\delta(\rho) \leq T$?
- The maxtime problem, i.e., do we have $\text{maxtime}(N) \leq T$?
  - In other words, does $\delta(\rho) \leq T$ for every final timed run $\rho$?

These questions have a practical interest: in the Brexit example, the question “is there a way to have a vote on a deal within 9 weeks?” is indeed a minimum execution time problem. We also address the equality variant of these decision problems, i.e., $\text{mintime}(N) = T$ : is there a final run of $N$ that terminates in exactly $T$ time units and no other final run takes less than $T$ time units?

Similarly for $\text{maxtime}(N) = T$.

**Example 3.** We use Fig. 1 to show that it is not easy to compute the minimal execution time, and in particular one cannot use the algorithm from [10] to compute it. Consider the node $n$ with $P_n = \{PM, Pa\}$ and $R_n = \{\text{court}, \text{no\_court}\}$. If the outcome is court, then $PM$ needs 2 weeks before he can talk to $EU$ and $Pa$ needs at least 3 weeks before he can debate. However, if the outcome is no_court, then $PM$ need not wait before he can talk to $EU$, but $Pa$ wastes 5 weeks in recession. This means that one needs to remember different alternatives which could be faster in the end, depending on the future. On the other hand, the algorithm from [10] attaches one minimal time to process $Pa$, and one minimal time to process $PM$. No matter the choices (0 or 2 for $PM$ and 3 or 5 for $Pa$), there will be futures in which the chosen number will over or underapproximate the real minimal execution time (this choice is not explicit in [10])\(^4\). For maximum execution time, it is not an issue to attach to each node a unique maximal execution time. The reason for the asymmetry between minimal execution time and maximal execution time of a negotiation is that the execution time of a path is $\max_{p \in P} \mu_k(p)$, for $\mu_k$ the last timed valuation, hence breaking the symmetry between $\min$ and $\max$.

\(^4\) the authors of [10] acknowledged the issue with their algorithm for mintime.
4 High Level view of the main results

In this section, we give a high-level description of our main results. Formal
statements can be found in the sections where they are proved. We gather in
Fig. 2 the precise complexities for the minimal and the maximal execution time
problems for 3 classes of negotiations that we describe in the following. Since we
are interested in minimum and maximum execution time, cycles in negotiations
can be either bypassed or lead to infinite maximal time. Hence, while we define
timed negotiations in general, we always restrict to acyclic negotiations (such as
Brexit) while stating and proving results.

In [10], a PTIME algorithm is given to compute different costs for negoti-
ations that are both sound and deterministic. One limitation of this result is
that it cannot compute the minimum execution time, as explained in Example
3. A second limitation is that the class of sound and deterministic negotiations
is quite restrictive: it cannot model situations where the next node a process
participates in depends on the outcome from another process, as in the Brexit
example. We thus consider classes where one of these restrictions is dropped.

We first consider (Section 5) negotiations that are deterministic, but without
the soundness restriction. We show that for this class, no timed problem
we consider can be solved in PTIME (unless NP=PTIME). Further, we show
that the equality problems (maxtime/mintime(\mathcal{N}) = T), are complete for the
complexity class DP, i.e., at the second level of the Boolean Hierarchy [16].

We then consider (Section 6) the class of negotiations that are sound, but not
necessarily deterministic. We show that maximum execution time can be solved
in PTIME, and propose a new algorithm. However, the minimum execution time
cannot be computed in PTIME (unless NP=PTIME). Again for the mintime
equality problem we have a matching DP-completeness result.

Finally, in order to obtain a polytime algorithm to compute the minimum
execution time, we consider the class of k-layered negotiations (see Section 7):
Given k \in \mathbb{N}, we can show that maxtime(\mathcal{N}) can be computed in PTIME for
k-layered negotiations. We also show that while the mintime(\mathcal{N}) \leq T? problem
is weakly NP-complete for k-layered negotiations, we can compute mintime(\mathcal{N})
in pseudo-PTIME, i.e. in PTIME if constants are given in unary.

|             | Deterministic | Sound           | k-layered       |
|-------------|---------------|-----------------|-----------------|
| Max \leq T  | co-NP-complete (Thm. 3) | PTIME (Prop. 3) | PTIME (Thm. 6)  |
| Max = T     | DP-complete (Prop. 2)     |                 |                 |
| Min \leq T  | NP-complete (Thm. 3)      | NP-complete* (Thm. 5) | pseudo-PTIME (Thm. 8) |
| Min = T     | DP-complete (Prop. 2)     | DP-complete* (Prop. 4) | pseudo-PTIME (Thm. 8) |

Fig. 2. Results for acyclic timed negotiations. DP refers to the complexity class, Difference Polynomial time [16], the second level of the Boolean Hierarchy.
* hardness holds even for very weakly non-deterministic negotiations, and T in unary.
** hardness holds even for sound and very weakly non-deterministic negotiations.
5 Deterministic Negotiations

We start by considering the class of deterministic acyclic negotiations. We show that both maximal and minimal execution time cannot be computed in PTIME (unless NP=PTIME), as the threshold problems are (co-)NP-complete.

**Theorem 3.** The mintime($\mathcal{N}$) $\leq T$ decision problem is NP complete, and the maxtime($\mathcal{N}$) $\leq T$ decision problem is co-NP complete for acyclic deterministic timed negotiations.

Proof. For mintime($\mathcal{N}$) $\leq T$, containment in NP is easy: we just need to guess a run $\rho$ (of polynomial size as $\mathcal{N}$ is acyclic), consider the associated timed run $\rho^{-}$ where all decisions are taken at their earliest possible dates, and check whether $\delta(\rho^{-}) \leq T$, which can be done in time $O(|\mathcal{N}|+\log T)$.

For the hardness, we give the proof in two steps. First, we start with a proof of Proposition 1 that reachability problem is NP-hard using reduction of 3-CNF SAT, i.e., given a formula $\phi$, we build a deterministic negotiation $\mathcal{N}_\emptyset$ s.t. $\phi$ is satisfiable iff $\mathcal{N}_\emptyset$ has a final run. In a second step, we introduce timings on this negotiation and show that mintime($\mathcal{N}_\emptyset$) $\leq T$ iff $\phi$ is satisfiable.

Step 1: Reducing 3-CNF-SAT to Reachability problem.

Given a boolean formula $\phi$ with variables $v_1, 1 \leq i \leq n$ and clauses $c_j, 1 \leq j \leq m$, for each variable $v_i$ we define the sets of clauses $S_{i,t} = \{c_j | v_i \text{ is present in } c_j\}$ and $S_{i,t}^{-} = \{c_j | \neg v_i \text{ is present in } c_j\}$. Clauses in $S_{i,t}$ and $S_{i,t}^{-}$ are naturally ordered: $c_i < c_j$ iff $i < j$. We denote these elements $S_{i,t}^{-}(1) < S_{i,t}^{-}(2) < \ldots$. Similarly for set $S_{i,t}$.

Now, we construct a negotiation $\mathcal{N}_\emptyset$ (as depicted in Figure 3) with a process $V_i$ for each variable $v_i$ and a process $C_j$ for each clause $c_j$:  

- Initial node $n_0$ has a single outcome $r$ taking each process $C_j$ to node Lone$_{c_j}$, and each process $V_i$ to node Lone$_{v_i}$.
- Lone$_{c_j}$ has three outcomes: if literal $v_i \in c_j$, then $t_i$ is an outcome, taking $V_i$ to Pair$_{c_j,v_i}$, and if literal $\neg v_i \in c_j$, then $f_i$ is an outcome, taking $V_i$ to Pair$_{c_j,\neg v_i}$.
- The outcomes of Lone$_{v_i}$ are true and false. Outcome true brings $v_i$ to node Tlonex$_{v_i}$, and outcome false brings $v_i$ to node Flonex$_{v_i}$.
- We have a node Tlonex$_{v_i,j}$ for each $j \leq |S_{i,t}|$ and Flonex$_{v_i,j}$ for each $j \leq |S_{i,t}^{-}|$, with $V_i$ as only process. Let $c_r = S_{i,t}^{-}(j)$. Node Tlonex$_{v_i,j}$ has two outcomes $vt_{on}$ bringing $V_i$ to Tlonex$_{v_i,j+1}$ (or $n_f$ if $j = |S_{i,t}^{-}|$), and $vt_{of_{i,r}}$ bringing $V_i$ to Pair$_{c_r,v_i}$. The two outcomes from Flonex$_{v_i,j}$ are similar.
- Node Pair$_{c_r,v_i}$ has $V_i$ and $C_r$ as its processes and one outcome $ctof$ which takes process $C_j$ to final node $n_f$ and process $V_i$ to Tlonex$_{v_i,j+1}$ (with $c_r = S_{i,t}^{-}(j)$), or to $n_f$ if $j = |S_{i,t}^{-}|$. Node Pair$_{c_r,\neg v_i}$ is defined in the same way from Flonex$_{v_i,j}$.

With this we claim that $\mathcal{N}_\emptyset$ has a final run iff $\phi$ is satisfiable which completes the first step of the proof. We give a formal proof of this claim in Appendix A. Observe that the negotiation $\mathcal{N}_\emptyset$ constructed is deterministic and acyclic (but it is not sound).
Step 2: Before we introduce timing on $\mathcal{N}_\phi$, we introduce a new outcome $r'$ at $n_0$ which takes all processes to $n_f$. Now, the timing function $\gamma$ associated with the $\mathcal{N}_\phi$ is: $\gamma(n_0, r) = [2, 2]$ and $\gamma(n_0, r') = [3, 3]$ and $\gamma(n, r) = [0, 0]$, for all node $n \neq n_0$ and all $r \in R_n$. Then, $\text{mintime}(\mathcal{N}_\phi) \leq 2$ iff $\phi$ has a satisfiable assignment: if $\text{mintime}(\mathcal{N}_\phi) \leq 2$, there is a run with decision $r$ taken at $n_0$ which is final. But existence of any such final run implies satisfiability of $\phi$. For reverse implication, if $\phi$ is satisfiable, then the corresponding run for satisfying assignment takes 2 units time, which means that $\text{mintime}(\mathcal{N}_\phi) \leq 2$.

Similarly, we can prove that the MaxTime problem is co-NP complete by changing $\gamma(n_0, r') = [1, 1]$ and asking if $\text{maxtime}(\mathcal{N}_\phi) > 1$ for the new $\mathcal{N}_\phi$. The answer will be yes iff $\phi$ is satisfiable.

We now consider the related problem of checking if $\text{mintime}(\mathcal{N}) = T$ (or if $\text{maxtime}(\mathcal{N}) = T$). These problems are harder than their threshold variant under usual complexity assumptions: they are DP-complete (Difference Polynomial...
time class, i.e., second level of the Boolean Hierarchy, defined as intersection of
a problem in NP and one in co-NP [16]).

**Proposition 2.** The \( \text{mintime}(\mathcal{N}) = T \) and \( \text{maxtime}(\mathcal{N}) = T \) decision problems are DP-complete for acyclic deterministic negotiations.

**Proof.** We only give the proof for \( \text{mintime} \) (the proof for \( \text{maxtime} \) is given in
Appendix A). Indeed, it is easy to see that this problem is in DP, as it can be
written as \( \text{mintime}(\mathcal{N}) \leq T \) which is in NP and \( \neg (\text{mintime}(\mathcal{N}) \leq T - 1) \),
which is in co-NP. To show hardness, we use the negotiation constructed in the
above proof as a gadget, and show a reduction from the SAT-UNSAT problem
(a standard DP-complete problem).

The SAT-UNSAT Problem asks given two Boolean expressions \( \phi \) and \( \phi' \), both
in CNF forms with three literals per clause, is it true that \( \phi \) is satisfiable and \( \phi' \)
is unsatisfiable? SAT-UNSAT is known to be DP-complete [16]. We reduce this
problem to \( \text{mintime}(\mathcal{N}) = T \).

Given \( \phi, \phi' \), we first make the corresponding negotiations \( \mathcal{N}_\phi \) and \( \mathcal{N}_{\phi'} \) as in
the previous proof. Let \( n_0 \) and \( n_f \) be the initial and final nodes of \( \mathcal{N}_\phi \) and \( n'_0 \)
and \( n'_f \) be the initial and final nodes of \( \mathcal{N}_{\phi'} \). (Similarly, for other nodes we write
' above the nodes to signify they belong to \( \mathcal{N}_{\phi'} \)).

In the negotiation \( \mathcal{N}_{\phi'} \), we introduce a new node \( n_{all} \), in which all the pro-
cesses participate (see Figure 4). The node \( n_{all} \) has a single outcome \( r'_{all} \) which
sends all the processes to \( n_f \). Also, for node \( n_0' \), apart from the outcome \( r \) which sends all processes to different nodes, there is another outcome \( r_{all} \) which sends all the processes to \( n_{all} \). Now we merge the nodes \( n_f \) and \( n_0 \) and call the merged node \( n_{sep} \). Also nodes \( n_0 \) and \( n_f' \) now have all the processes of \( N_\phi \) and \( N_{\phi'} \) participating in them. This merged process gives us a new negotiation \( N_{\phi',\phi'} \) in which the structure above \( n_{sep} \) is same as \( N_\phi \) while below it is same as \( N_{\phi'} \). Node \( n_{sep} \) now has all the processes of \( N_\phi \) and \( N_{\phi'} \) participating in it. The outcomes of \( n_{sep} \) will be same as that of \( n_0' \) (\( r_{all}, r \)). For both the outcomes of \( n_{sep} \) the processes corresponding to \( N_\phi \) directly go to \( n_f \) of the \( N_{\phi',\phi'} \). Similarly \( n_0 \) of \( N_{\phi',\phi'} \) which is same \( n_0 \) of \( N_\phi \), sends processes corresponding to \( N_{\phi',\phi'} \) directly to \( n_{sep} \) for all its outcomes. We now define timing function \( \gamma \) for \( N_{\phi',\phi'} \) which is as follows: \( \gamma(Lone_{v_i}, r) = [1, 1] \) for all \( v_i \in \phi' \) and \( r \in \{\text{true, false}\} \), \( \gamma(n_{all}, r_{all}) = [2, 2] \) and \( \gamma(n, r) = [0, 0] \) for all other outcomes of nodes. With this construction, one can conclude that \( \min time(N_{\phi',\phi'}) = 2 \) iff \( \phi \) is satisfiable and \( \phi' \) is unsatisfiable (see Appendix for details). This completes the reduction and hence proves DP-hardness.

Finally, we consider a related problem of computing the min and max time.

To consider the decision variant, we rephrase this problem as checking whether an arbitrary bit of the minimum execution time is 1. Perhaps surprisingly, we obtain that this problem goes even beyond DP, the second level of the Boolean Hierarchy and is in fact hard for \( \Delta^P_2 \) (second level of the polynomial hierarchy), which contains the entire Boolean Hierarchy. Formally,

**Theorem 4.** Given an acyclic deterministic timed negotiation and a positive integer \( k \), computing the \( k \)th bit of the maximum/minimum execution time is \( \Delta^P_2 \)-complete.

Finally, we remark that if we were interested in the optimization variant and not the decision variant of the problem, the above proof can be adapted to show that these variants are OptP-complete (as defined in [14]). But as optimization is not the focus of this paper, we avoid formal details of this proof.

## 6 Sound Negotiations

Sound negotiations are negotiations in which every run can be extended to a final run, as in Fig. 1. In this section, we show that \( \max time(N) \) can be computed in PTIME for sound negotiations, hence giving PTIME complexities for the \( \max time(N) \leq T \) and \( \max time(N) = T \) questions. However, we show that \( \min time(N) \leq T \) is NP-complete for sound negotiations, and that \( \min time(N) = T \) is DP-complete, even if \( T \) is given in unary.

Consider the graph \( G_N \) of a negotiation \( N \). Let \( \pi = (n_0, (p_0, r_0), n_1) \cdots (n_k, (p_k, r_k), n_{k+1}) \) be a path of \( G_N \). We define the maximal execution time of a path \( \pi \) as the value \( \delta^+(\pi) = \sum_{i \in 0..k} \gamma^+(n_i, r_i) \). We say that a path \( \pi = (n_0, (p_0, r_0), n_1) \cdots (n_k, (p_k, r_k), n_{k+1}) \) is a path of some run \( \rho = (M_1, \mu_1) \mapsto (n_1, r_1') \cdots (M_k, \mu_k) \) if \( r_0, \ldots, r_k \) is a subword of \( r_1', \ldots, r_k' \).
Lemma 1. Let $\mathcal{N}$ be an acyclic and sound timed negotiation. Then $\text{maxtime}(\mathcal{N}) = \max_{\pi \in \text{Paths}(\mathcal{N}_\pi)} \delta^+(\pi) + \gamma^+(n_f, r_f)$.

Proof. Let us first prove that $\text{maxtime}(\mathcal{N}) \geq \max_{\pi \in \text{Paths}(\mathcal{G}_\pi)} \delta^+(\pi) + \gamma^+(n_f, r_f)$.

Consider any path $\pi$ of $G_{\mathcal{N}}$, ending in some node $n$. First, as $\mathcal{N}$ is sound, we can compute a run $\rho_\pi$ such that $\pi$ is a path of $\rho_\pi$, and $\rho_\pi$ ends in a configuration in which $n$ is enabled. We associate with $\rho_\pi$ the timed run $\rho_\pi^+$ which associates to every node the latest possible execution date. We have easily $\delta(\rho_\pi^+) \geq \delta(\pi)$, and then we obtain $\max_{\pi \in \text{Paths}(\mathcal{G}_\pi)} \delta(\rho_\pi^+) \geq \max_{\pi \in \text{Paths}(\mathcal{G}_\pi)} \delta(\pi)$. As $\text{maxtime}(\mathcal{N})$ is the maximal execution time over all runs, it is hence necessarily greater than $\max_{\pi \in \text{Paths}(\mathcal{G}_\pi)} \delta(\rho_\pi^+) + \gamma^+(n_f, r_f)$.

We now prove that $\text{maxtime}(\mathcal{N}) \leq \max_{\pi \in \text{Paths}(\mathcal{G}_\pi)} \delta^+(\pi) + \gamma^+(n_f, r_f)$. Take any timed run $\rho = (M_1, \mu_1) (n_1, r_1) \cdots (M_k, \mu_k)$ of $\mathcal{N}$ with a unique maximal node $n_k$. We show that there exists a path $\pi$ of $\rho$ such that $\delta(\rho) \leq \delta^+(\pi)$ by induction on the length $k$ of $\rho$. The initialization is trivial for $k = 1$. Let $k \in \mathbb{N}$. Because $n_k$ is the unique maximal node of $\rho$, we have $\delta(\rho) = \max_{p \in P_{n_k}^+} \mu_{k-1}(p) + \gamma^+(n_k, r_k)$.

We choose one $p_{k-1}$ maximizing $\mu_{k-1}(p)$. Let $\ell < k$ be the maximal index of a decision involving process $p_{k-1}$ (i.e. $p_{k-1} \in P_{n_\ell}$). Now, consider the timed run $\rho'$ subword of $\rho$, but with $n_\ell$ as a new maximal node (that is, it is $\rho$ where nodes $n_i, i > \ell$ has been removed, but also where some nodes $n_i, i < \ell$ have been removed if they are not causally before $n_\ell$ (in particular, $P_{n_i} \cap P_{n_\ell} = \emptyset$).

By definition, we have that $\delta(\rho) = \delta(\rho') + \gamma^+(n_\ell, r_\ell) + \gamma^+(n_k, r_k)$. We apply the induction hypothesis on $\rho'$, and obtain a path $\pi'$ of $\rho'$ ending in $n_\ell$ such that $\delta(\rho') + \gamma^+(n_\ell, r_\ell) \leq \delta^+(\pi')$. It suffices to consider the path $\pi' = \pi'.(n_\ell, (p_{k-1}, r_\ell), n_k)$ to prove the inductive step $\delta(\rho) \leq \delta^+(\pi) + \gamma^+(n_k, r_k)$.

Thus $\text{maxtime}(\mathcal{N}) = \max \delta(\rho) \leq \max_{\pi \in \text{Paths}(\mathcal{G}_\pi)} \delta^+(\pi) + \gamma^+(n_f, r_f)$.  

Lemma 1 gives a way to evaluate the maximal execution time. This amounts to finding a path of maximal weight in an acyclic graph, which is a standard PTIME problem that can be solved using standard max-cost calculation.

Proposition 3. Computing the maximal execution time for an acyclic sound negotiation $\mathcal{N} = (\mathcal{N}, n_0, n_f, \mathcal{X})$ can be done in time $O(|\mathcal{N}| + |\mathcal{X}|)$.

A direct consequence is that $\text{maxtime}(\mathcal{N}) \leq T$ and $\text{maxtime}(\mathcal{N}) = T$ problems can be solved in polynomial time when $\mathcal{N}$ is sound. Notice that if $\mathcal{N}$ is deterministic but not sound, then Lemma 1 does not hold: we only have an inequality.

We now turn to $\text{mintime}(\mathcal{N})$. We show that it is strictly harder to compute for sound negotiations than $\text{maxtime}(\mathcal{N})$.

Theorem 5. $\text{mintime}(\mathcal{N}) \leq T$ is NP-complete in the strong sense for sound acyclic negotiations, even if $\mathcal{N}$ is very weakly non-deterministic.

Proof (sketch). First, we can decide $\text{mintime}(\mathcal{N}) \leq T$ in NP. Indeed, one can guess a final (untimed) run $\rho$ of size $\leq |\mathcal{N}|$, consider $\rho^-$ the timed run corresponding to $\rho$ where all outcomes are taken at the earliest possible dates, and compute in linear time $\delta(\rho^-)$, and check that $\delta(\rho^-) \leq T$. 

14
The hardness part is obtained by reduction from the Bin Packing problem. The reduction is similar to Knapsack, that we will present in Thm. 7. The difference is that we use \( \ell \) bins in parallel, rather than 2 process, one for the weight and one for the value. The hardness is thus strong, but the negotiation is not \( k \)-layered for a bounded \( k \) (It is \( 2\ell + 1 \) bounded, with \( \ell \) depending on the input). A detailed proof is given in Appendix B. \( \square \)

We show that \( \text{mintime}(N) = T \) is harder to decide than \( \text{mintime}(N) \leq T \), with a proof similar to Prop. 2.

**Proposition 4.** The \( \text{mintime}(N) = T \) decision problem is DP-complete for sound acyclic negotiations, even if it is very weakly non-deterministic.

An open question is whether the minimal execution time can be computed in PTIME if the negotiation is both sound and deterministic. The reduction from Bin Packing does not work with deterministic (and sound) negotiations.

### 7 \( k \)-Layered Negotiations

In the previous sections, we have considered sound negotiations, and deterministic negotiations. For both classes, computing the minimal execution time cannot be done in PTIME (unless \( \text{NP} = \text{PTIME} \)), even if constants are given in unary.

In this section, we consider \( k \)-layeredness (see Section 2), a syntactic property that can be efficiently verified (it suffices to compute the depth of each node, which can be done in polynomial time).

#### 7.1 Algorithmic properties

Let \( k \) be a fixed integer. We first show that the maximum execution time can be computed in PTIME for \( k \)-layered negotiations. Let \( N_i \) be the set of nodes at layer \( i \). We define for every layer \( i \) the set \( S_i \) of subsets of nodes \( X \subseteq N_i \) which can be jointly enabled and such that for every process \( p \), there is exactly one node \( n(X,p) \) in \( X \) with \( p \in n(X,p) \). Formally, we define \( S_i \) inductively. We start with \( S_0 = \{ n_0 \} \). We then define \( S_{i+1} \) from the contents of layer \( S_i \): we have \( Y \in S_{i+1} \iff \bigcup_{n \in Y} P_n = P \) and there exist \( X \in S_i \) and an outcome \( r_m \in R_m \) for every \( m \in X \), such that \( n \in X(n(X,p),p,r_m) \) for each \( n \in Y \) and \( p \in P_n \).

**Theorem 6.** Let \( k \in \mathbb{N}^+ \). Computing the maximum execution time for a \( k \)-layered acyclic negotiation \( N \) can be done in PTIME. More precisely, the worst-case time complexity is \( O(|P| \cdot |N|^{k+1}) \).

**Proof (Sketch).** The first step is to compute \( S_i \) layer by layer, by following its inductive definition. The set \( S_i \) is of size at most \( 2^k \), as \( |N_i| < k \) by definition of \( k \)-layeredness. Knowing \( S_i \), it is easy to build \( S_{i+1} \) by induction. This takes time in \( O(|P| \cdot |N|^{k+1}) \). We need to consider all \( k \)-uple of outcomes for each layer. There can be \( |N|^k \) such tuples. We need to do that for all processes \(|P|\), and for all layers (at most \(|N|\)).
We then keep for each subset $X \in S_i$ and each node $n \in X$, the maximal time $f_i(n,X) \in \mathbb{N}$ associated with $n$ and $X$. From $S_{i+1}$ and $f_i$, we inductively compute $f_{i+1}$ in the following way: for all $X \in S_i$ with successor $Y \in S_{i+1}$ for outcomes $(r_p)_{p \in P}$, we denote $f_{i+1}(Y,n,X) = \max_{r_p \in P(n)} f_i(X,n(X,p)) + \gamma^+(n(X,p), r_p)$. If there are several choices of $(r_p)_{p \in P}$ leading to the same $Y$, we take $r_p$ with the maximal $f_i(X,n(X,p)) + \gamma^+(n(X,p), r_p)$. We then define $f_{i+1}(Y,n) = \max_{X \in S_i} f_{i+1}(Y,n,X)$. Again, the initialization is trivial, with $f_0(\{n_0\}, n_0) = 0$. The maximal execution time of $\mathcal{N}$ is $f(\{n_f\}, n_f)$.

We can bound the complexity precisely by $O(d(\mathcal{N}) \cdot C(\mathcal{N}) \cdot ||R||^k)$, with:

- $d(\mathcal{N}) \leq |\mathcal{N}|$ the depth of $n_f$, that is the number of layers of $\mathcal{N}$, and $||R||$ is the maximum number of outcomes of a node,
- $C(\mathcal{N}) = \max_i |S_i| \leq 2^k$, which we will call the number of contexts of $\mathcal{N}$, and which is often much smaller than $2^k$.
- $k^* = \max_{X \in \cup_j S_i} |X| \leq k$. We say that $\mathcal{N}$ is $k^*$-thread bounded, meaning that there cannot be more than $k^*$ nodes in the same context $X$ of any layer. Usually, $k^*$ is strictly smaller than $k = \max_i |N_i|$, as $N_i = \bigcup_{X \in S_i} X$.

Consider again the Brexit example Figure 1. We have $(k+1) = 7$, while we have the depth $d(\mathcal{N}) = 6$, the negotiation is $k^* = 3$-thread bounded ($k^*$ is bounded by the number of processes), $||R|| = 2$, and the number of contexts is at most $C(\mathcal{N}) = 4$ (EU chooses to enforce backstop or not, and Pa chooses to go to court or not).

### 7.2 Minimal Execution Time

As with sound negotiations, computing minimal time is much harder than computing the maximal time for $k$-layered negotiations:

**Theorem 7.** Let $k \geq 6$. The Min $\leq T$ problem is NP-Complete for $k$-layered acyclic negotiations, even if the negotiation is sound and very weakly non-deterministic.

**Proof.** One can guess in polynomial time a final run of size $\leq |\mathcal{N}|$. If the execution time of this final run is smaller than $T$ then we have found a final run witnessing $\text{mintime}(\mathcal{N}) \leq T$. Hence the problem is in NP.

Let us now show that the problem is NP-hard. We proceed by reduction from the Knapsack decision problem. Let us consider a set of items $U = \{u_1, \ldots, u_n\}$ of respective values $v_1, \ldots, v_n$ and weight $w_1, \ldots, w_n$ and a knapsack of maximal capacity $W$. The knapsack problem asks, given a value $V$ whether there exists a subset of items $U' \subseteq U$ such that $\sum_{u_i \in U'} v_i \geq V$ and such that $\sum_{u_i \in U'} w_i \leq W$.

We build a negotiation with $2n$ processes $P = \{p_1, \ldots, p_{2n}\}$, as shown in Fig. 5. Intuitively, $p_i, i \leq n$ will serve to encode the value of selected items as timing, while $p_i, i > n$ will serve to encode the weight of selected items as timing.

Concerning timing constraints for outcomes we do the following: Outcomes 0, yes and no are associated with $[0,0]$. Outcome $c_i$ is associated with $[w_i, w_i]$, the weight of $u_i$. Last, outcome $b_i$ is associated with a more complex function,
such that $\sum_i b_i \leq W$ iff $\sum_i v_i \geq V$. For that, we set $[\frac{(v_{\text{max}} - v_i)W}{n \cdot v_{\text{max}} - v_i}, \frac{v_{\text{max}} W}{n \cdot v_{\text{max}} - v_i}]$ for outcome $b_i$, where $v_{\text{max}}$ is the largest value of an item, and $V$ is the total value we want to reach at least. Also, we set $[\frac{(v_{\text{max}} W)}{n \cdot v_{\text{max}} - v_i}, \frac{v_{\text{max}} W}{n \cdot v_{\text{max}} - v_i}]$ for outcome $a_i$. We set $T = W$, the maximal weight of the knapsack.

Now, consider a final run $\rho$ in $\mathcal{N}$. The only choices in $\rho$ are outcomes $\text{yes}$ or $\text{no}$ from $C_1, \ldots, C_n$. Let $I$ be the set of indices such that $\text{yes}$ is the outcome from all $C_i$ in this path. We obtain $\delta(\rho) = \max(\sum_{i \in I} a_i + \sum_{i \in I} b_i, \sum_{i \in I} c_i)$. We have $\delta(\rho) \leq T = W$ iff $\sum_{i \in I} a_i + \sum_{i \in I} b_i \leq W$, that is the sum of the weights is lower than $W$, and $\sum_{i \in I} \frac{(v_{\text{max}} - v_i)W}{n \cdot v_{\text{max}} - v_i} \leq W$. That is, $n \cdot v_{\text{max}} - \sum_{i \in I} v_i \leq n \cdot v_{\text{max}} - V$, i.e. $\sum_{i \in I} v_i \geq V$. Hence, there exists a path $\rho$ with $\delta(\rho) \leq T = W$ iff there exists a set of items of weight less than $W$ and of value more than $V$. \hfill $\Box$

It is well known that Knapsack is weakly NP-hard, that is, it is NP-hard only when weights/values are given in binary. This means that Thm. 7 shows that minimum execution time $\leq T$ is NP-hard only when $T$ is given in binary. We can actually show that for $k$-layered negotiations, the $\text{mintime}(\mathcal{N}) \leq T$ problem can be decided in PTIME if $T$ is given in unary (i.e. if $T$ is not too large):

**Theorem 8.** Let $k \in \mathbb{N}$. Given a $k$-layered negotiation $\mathcal{N}$ and $T$ written in unary, one can decide in PTIME whether the minimum execution time of $\mathcal{N}$ is $\leq T$. The worst-case time complexity is $O(|\mathcal{N}| \cdot |P| \cdot (T \cdot |\mathcal{N}|)^k)$.
Proof. We will remember for each layer \( i \) a set \( T_i \) of functions \( \tau \) from nodes \( N_i \) of layer \( i \) to a value in \{1, \ldots, T, \bot\}. Basically, we have \( \tau \in T_i \) if there exists a path \( \rho \) reaching \( X = \{ n \in N_i \mid f(n) \neq \bot \} \), and this path reaches node \( n \in X \) after \( \tau(n) \) time units. As for \( S_i \), for all \( p \), we should have a unique node \( n(\tau, p) \) such that \( p \in n(f, p) \) and \( \tau(n(\tau, p)) \neq \bot \). Again, it is easy to initialize \( T_0 = \{ \tau_0 \} \), with \( \tau_0(n_0) = 0 \), and \( \tau_0(n) = \bot \) for all \( n \neq n_0 \).

Inductively, we build \( T_{i+1} \) in the following way: \( \tau_{i+1} \in T_{i+1} \) iff there exists a \( \tau_i \in T_i \) and \( r_p \in R_{n(\tau_i, p)} \) for all \( p \in P \) such that for all \( n \) with \( \tau_{i+1}(n) \neq \bot \), we have \( \tau_{i+1}(n) = \max_p \tau_i(n(\tau_i, p)) + \gamma(n(\tau_i, p), r_p) \).

We have that the minimum execution time for \( N \) is \( \min_{\tau \in T_{i+1}} \tau(n_f) \), for \( n \) the depth of \( n_f \). There are at most \( T^k \) functions \( \tau \) in any \( T_i \), and there are at most \(|N|\) layers to consider, giving the complexity. \( \square \)

As with Thm. 6, we can more accurately state the complexity as \( O(d(N) \cdot C(N) \cdot ||R||^{k \cdot T^{k-1}}) \). The \( k^* - 1 \) is because we only need to remember minimal functions \( \tau \in T_i \); if \( \tau'(n) \geq \tau(n) \) for all \( n \), then we do not need to keep \( \tau' \) in \( T_i \).

In particular, for the knapsack encoding in the proof of Thm. 7, we have \( k^* = 3 \), \( ||R|| = 2 \) and \( C(N) = 4 \).

Notice that if \( k \) is part of the input, then the problem is strongly NP-hard, even if \( T \) is given in unary, as e.g. encoding bin packing with \( \ell \) bins result to a \( 2\ell + 1 \)-layered negotiations.

8 Conclusion

In this paper, we considered timed negotiations. We believe that time is of the essence in negotiations, as exemplified by the Brexit negotiation. It is thus important to be able to compute in a tractable way the minimal and maximal execution time of negotiations.

We showed that we can compute in PTIME the maximal execution time for acyclic negotiations that are either sound or \( k \)-layered, for \( k \) fixed. We showed that we cannot compute in PTIME the maximal execution time for negotiations that are not sound nor \( k \)-layered, even if they are deterministic and acyclic (unless NP=PTIME). We also showed that surprisingly, computing the minimal execution time is much harder, with strong NP-hardness results in most of the classes of negotiations, contradicting a claim in [10]. We came up with a new reasonable class of negotiations, namely \( k \)-layered negotiations, which enjoys a pseudo PTIME algorithm to compute the minimal execution time. That is, the algorithm is PTIME when the timing constants are given in unary. We showed that this restriction is necessary, as the problem becomes NP-hard for constants given in binary, even when the negotiation is sound and very weakly non-deterministic. The problem to know whether the minimal execution time can be computed in PTIME for deterministic and sound negotiation remains open.
References

1. Jörg Desel. Reduction and design of well-behaved concurrent systems. In CONCUR '90, Theories of Concurrency: Unification and Extension, Amsterdam, The Netherlands, August 27-30, 1990, Proceedings, volume 458 of Lecture Notes in Computer Science, pages 166–181. Springer, 1990.

2. Jörg Desel, Javier Esparza, and Philipp Hoffmann. Negotiation as concurrency primitive. Acta Inf., 56(2):93–159, 2019.

3. Knuth (Donald E.). Fundamental Algorithms, volume 1 of The Art of Computer Programming. Addison-Wesley, 1973.

4. J. Esparza and Jörg Desel. Free Choice Petri nets. Cambridge University Press, 1995.

5. Javier Esparza. Decidability and complexity of petri net problems - an introduction. In Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, Dagstuhl, September 1996, volume 1491 of Lecture Notes in Computer Science, pages 374–428. Springer, 1998.

6. Javier Esparza and Jörg Desel. On negotiation as concurrency primitive. In CONCUR 2013 - Concurrency Theory - 24th International Conference, CONCUR 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings, volume 8052 of Lecture Notes in Computer Science, pages 440–454. Springer, 2013.

7. Javier Esparza and Jörg Desel. On negotiation as concurrency primitive II: deterministic cyclic negotiations. In FOSSACS'14, volume 8412 of Lecture Notes in Computer Science, pages 258–273. Springer, 2014.

8. Javier Esparza and Philipp Hoffmann. Reduction rules for colored workflow nets. In Fundamental Approaches to Software Engineering - 19th International Conference, FASE 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings, volume 9633 of Lecture Notes in Computer Science, pages 342–358. Springer, 2016.

9. Javier Esparza, Denis Kuperberg, Anca Muscholl, and Igor Walukiewicz. Soundness in negotiations. Logical Methods in Computer Science, 14(1), 2018.

10. Javier Esparza, Anca Muscholl, and Igor Walukiewicz. Static analysis of deterministic negotiations. In 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017, pages 1–12, 2017.

11. Michael R. Garey and David S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA, 1979.

12. Serge Haddad. A reduction theory for coloured nets. In Advances in Petri Nets 1989, volume 424 of Lecture Notes in Computer Science, pages 209–235. Springer, 1990.

13. Philipp Hoffmann. Negotiation games. In Javier Esparza and Enrico Tronci, editors, Proceedings Sixth International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2015, Genoa, Italy, 21-22nd September 2015., volume 193 of EPTCS, pages 31–42, 2015.

14. Mark W Krentel. The complexity of optimization problems. Journal of computer and system sciences, 36(3):490–509, 1988.

15. P.M. Merlin. A Study of the Recoverability of Computing Systems. PhD thesis, University of California, Irvine, CA, USA, 1974.

16. C. H. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). In Proceedings of the Fourteenth Annual ACM Symposium
Appendix A: Deterministic Negotiations

We start by considering the class of deterministic acyclic negotiations. We show that both maximal and minimal execution time cannot be computed in PTIME (unless NP=PTIME), as the threshold problems are (co-)NP-complete.

Theorem 3. The \( \text{mintime}(N) \leq T \) decision problem is NP complete, and the \( \text{maxtime}(N) \leq T \) decision problem is co-NP complete for acyclic deterministic timed negotiations.

Proof. For \( \text{mintime}(N) \leq T \), containment in NP is easy; we just need to guess a run \( \rho \) (of polynomial size as \( N \) is acyclic), consider the associate timed run \( \rho^- \) where all decisions are taken at their earliest possible dates, and check whether \( \delta(\rho^-) \leq T \), which can be done in time \( O(|N|+\log T) \).

For the hardness, we give the proof in two steps. First, we start with a proof of Proposition 1 that reachability problem is NP-hard using reduction of 3-CNF SAT, i.e., given a formula \( \phi \), we build a deterministic negotiation \( N_\phi \) s.t. \( \phi \) is satisfiable iff \( N_\phi \) has a final run. In a second step, we introduce timings on this negotiation and show that \( \text{mintime}(N_\phi) \leq T \) iff \( \phi \) is satisfiable.

Step 1: Reducing 3-CNF-SAT to Reachability problem.

Given a boolean formula \( \phi \) with variables \( v_i, 1 \leq i \leq n \) and clauses \( c_j, 1 \leq j \leq m \), for each variable \( v_i \) we define the sets of clauses \( S_{i,t} = \{ c_j | v_i \text{ is present in } c_j \} \) and \( S_{i,t} = \{ c_j | \neg v_i \text{ is present in } c_j \} \). Clauses in \( S_{i,t} \) and \( S_{i,f} \) are naturally ordered: \( c_i < c_j \) iff \( i < j \). We denote these elements \( S_{i,t}(1) < S_{i,t}(2) < \ldots \). Similarly for set \( S_{i,f} \).

Now, we construct a negotiation \( N_\phi \) with a process \( V_i \) for each variable \( v_i \) and a process \( C_j \) for each clause \( c_j \):

- Initial node \( n_0 \) has a single outcome \( r \) taking each process \( C_j \) to node \( Lone_{v_i} \), and each process \( V_i \) to node \( Lone_{v_i} \).
- \( Lone_{v_i} \) has three outcomes: if literal \( v_i \in c_j \), then \( t_i \) is an outcome, taking \( V_i \) to \( Pair_{c_i,v_i} \), and if literal \( \neg v_i \in c_j \), then \( f_i \) is an outcome, taking \( V_i \) to \( Pair_{c_i,\neg v_i} \).
- The outcomes of \( Lone_{v_i} \) are true and false. Outcome true brings \( v_i \) to node \( Tlone_{v_i,1} \) and outcome false brings \( v_i \) to node \( Tlone_{v_i,1} \).
- We have a node \( Tlone_{v_i,j} \) for each \( j \leq |S_{v_i,t}| \) and \( Flone_{v_i,j} \) for each \( j \leq |S_{v_i,t}| \), with \( V_i \) as only process. Let \( c_r = S_{v_i,t}(j) \). Node \( Tlone_{v_i,j} \) has two outcomes \( vton \) bringing \( V_i \) to \( Tlone_{v_i,j+1} \) (or \( n_f \) if \( j = |S_{v_i,t}| \)), and \( vtoc_{v_i} \) bringing \( V_i \) to \( Pair_{v_i} \). The two outcomes from \( Flone_{v_i,j} \) are similar.
- Node \( Pair_{c_r,v_i} \) has \( V_i \) and \( C_r \) as its processes and one outcome \( ctof \) which takes process \( C_j \) to final node \( n_f \) and process \( V_i \) to \( Tlone_{v_i,j+1} \) (with \( c_r = S_{v_i,t}(j) \), or to \( n_f \) if \( j = |S_{v_i,t}| \)). Node \( Pair_{c_r,\neg v_i} \) is defined in the same way from \( Flone_{v_i,j} \).
Fig. 3. A part of $N_\varphi$ where clause $c_j$ is $(i_2 \lor \neg i \lor \neg i_3)$ and clause $c_k$ is $(i_4 \lor \neg i \lor i_5)$. Timing is $[0, 0]$ wherever not mentioned.
Claim. \( N_\phi \) has a final run iff \( \phi \) is satisfiable.

Proof. First we show that if there is a run \( \rho \) from \( n_0 \) to \( n_f \) then \( \phi \) is satisfiable. In \( \rho \), all processes reached \( n_f \). So each process \( V_i \) takes either outcome \text{true} or \text{false} in \( \rho \). Let \( val \) the valuation associated each variable \( v_i \) with the choice \text{true} or \text{false} by \( V_i \). We now show that all clause \( c_r \) have at least one literal true in \( val \). In \( \rho \), process \( C_r \) reaches the final node \( n_f \); it must have gone via one node either \( \text{Pair}_{c_r,v_i} \) or \( \text{Pair}_{c_r,\neg v_i} \), for some \( i \). Wlog, let us assume that \( C_r \) went to \( \text{Pair}_{c_r,v_i} \). The only way it is possible is for process \( V_i \) to have been in \( \text{Flone}_{v_i,j} \), with \( c_r = S_{i,t}(j) \). This is possible only if \( V_i \) decided outcome \text{false} at \( \text{Lone}_v \). So this implies that literal \( \neg v_i \) of \( c_j \) is true in \( val \). Hence \( \phi \) is satisfiable.

Conversely, we show that if \( \phi \) is satisfiable then \( N_\phi \) has final run. Let \( val \) a satisfiable assignment \( val : V \rightarrow \{ \text{true}, \text{false} \} \) for \( \phi \). We build a run \( \rho \) which is final. After reaching \( \text{Lone}_{v_i} \), \( V_i \) will decide the outcome according to the value of \( val(v_i) \) and reach \( \text{Flone}_{v_i,1} \) or \( \text{Tlone}_{v_i,1} \) accordingly. Let \( G_i(val) \) be the set of clause \( c_j \) such that \( i \) is the minimal literal of \( c_j \) true under \( val \). When there is a choice between two outcomes \( vton \) and \( vtoc_{i,k} \) for process \( V_i \), the run chooses \( vtoc_{i,k} \) iff \( k \in G_i(val) \). Concerning \( C_j \), it appears in exactly one \( G_i(val) \), because \( val \) satisfies \( \phi \). If \( val(v_i) = \text{true} \), run \( \rho \) chooses outcome \( t_i \) for \( V_i \) in node \( \text{Lone}_{c_j} \), and outcome \( f_i \) if \( val(v_i) = \text{false} \). Observe that the same variable \( v_i \) can be associated with several clauses \( c_j \), but then all these clauses go to the same type of nodes i.e. \( \text{Pair}_{c_j,v_i} \) if \( val(v_i) = \text{true} \) and \( \text{Pair}_{c_j,\neg v_i} \) if \( val(v_i) = \text{false} \).

This run \( \rho \) is final: Every process \( C_j \) reaches \( n_f \) after participating in exactly one node \( \text{Pair}_{c_j,v_i} \) or \( \text{Pair}_{c_j,\neg v_i} \). Every process \( V_i \) reaches \( n_f \) after participating in zero or more node \( \text{Pair}_{c_j,v_i} \) or \( \text{Pair}_{c_j,\neg v_i} \) (it participates in exactly \( |G_i| \) such nodes).

With this we claim that \( N_\phi \) has a final run iff \( \phi \) is satisfiable which completes the first step of the proof. Observe that the negotiation \( N_\phi \) constructed is deterministic and acyclic (but it is not sound).

Step 2: Before we introduce timing on \( N_\phi \), we introduce a new outcome \( r' \) at \( n_0 \) which takes all processes to \( n_f \). Now, the timing function \( \gamma \) associated with the \( N_\phi \) is: \( \gamma(n_0,r) = [2,2] \) and \( \gamma(n_0,r') = [3,3] \) and \( \gamma(n,r) = [0,0] \), for all node \( n \neq n_0 \) and all \( r \in R_n \). Then, \( \text{mintime}(N_\phi) \leq 2 \) iff \( \phi \) has a satisfiable assignment: if \( \text{mintime}(N_\phi) \leq 2 \), there is a run with decision \( r \) taken at \( n_0 \) which is final. But existence of any such final run implies satisfiability of \( \phi \). For reverse implication, if \( \phi \) is satisfiable, then the corresponding run for satisfying assignment takes \( 2 \) units time, which means that \( \text{mintime}(N_\phi) \leq 2 \).

Similarly, we can prove that the MaxTime problem is Co-NP complete by changing \( \gamma(n_0,r') = [1,1] \) and asking if \( \text{maxtime}(N_\phi) > 1 \) for the new \( N_\phi \). The answer will be yes iff \( \phi \) is satisfiable.

As a side note, we observe that the NP-hardness for \( \text{mintime} \) could also have been proved without introducing the new result \( r' \) but then it would have been possible that \( N_\phi \) had no final run making \( \text{mintime}(N_\phi) \leq 2 \) vacuous.

We now consider the related problem of checking if \( \text{mintime}(N) = T \) (or if \( \text{maxtime}(N) = T \)). These problems are harder than their threshold variant un-
under usual complexity assumptions: they are DP-complete (Difference Polynomial
time class, i.e., second level of the Boolean Hierarchy, defined as intersection of
a problem in NP and co-NP [16]).

**Proposition 2.** The \( \minTime(N) = T \) and \( \maxTime(N) = T \) decision prob-
lems are DP-complete for acyclic deterministic negotiations.

**Proof.** Indeed, it is easy to see that this problem is in DP, as it can be written
as \( \minTime(N) \leq T \) which is in NP and \( \neg(\minTime(N) \leq T - 1) \), which is in
co-NP. To show hardness, we use the negotiation constructed in the above proof
as a gadget, and show a reduction from the SAT-UNSAT problem (a standard
DP-complete problem).

SAT-UNSAT Problem : Given two Boolean expressions \( \phi \) and \( \phi' \), both in
CNF forms with three literals per clause, is it true that \( \phi \) is satisfiable and \( \phi' \) is
unsatisfiable? SAT-UNSAT is known to be DP-Complete [16]. We reduce this
problem to \( \minTime(N) = T \).

Given \( \phi, \phi' \), we first make the corresponding negotiations \( N_\phi \) and \( N_{\phi'} \) as in
the previous proof. Let \( n_0 \) and \( n_f \) be the initial and final nodes of \( N_\phi \) and \( n_0 \) and
\( n_f \) be the initial and final nodes of \( N_{\phi'} \). (Similarly, for other nodes we write \( ' \) above
the nodes to signify they belong to \( N_{\phi'} \)). In the negotiation \( N_{\phi'} \), we introduce
a new node \( n_{all} \) (see Figure 4), in which all the processes participate. The node
\( n_{all} \) has a single outcome \( r'_{all} \) which sends all the processes to \( n_f \). Also, for node

Fig. 4. Structure of \( N_{\phi,\phi'} \)
n'_0, apart from the outcome r which sends all processes to different nodes, there
is another outcome r_{all} which sends all the processes to n_{all}.
Now we merge the nodes n_f and n'_0 and call the merged node n_{sep}. Also nodes
n_0 and n'_f now have all the processes of N_0 and N'_0 participating in them.
This merged process gives us a new negotiation N'_{0,0'} in which the structure
above n_{sep} is same as N_0 while below it is same as N'_0. Node n_{sep} now has all the
processes of N_0 and N'_0 participating in it. The outcomes of n_{sep} will be same
as that of n'_0 (r_{all}, r). For both the outcomes of n_{sep} the processes corresponding
to N_0 directly go to n_f of the N'_{0,0'}. Similarly n_0 of N'_{0,0'} which is same n_0 of
N_0 sends processes corresponding to N'_0 directly to n_{sep} for all its outcomes.
We now define timing function \gamma for N'_{0,0'} which is as follows:
\begin{itemize}
\item \gamma(L_{0,v_i}, r) = [1, 1] for all v_i \in \phi' and r \in \{true, false\},
\item \gamma(n_{all}, r_{all}') = [2, 2] and
\item \gamma(n, r) = [0, 0] for all other outcomes of nodes.
\end{itemize}

The claim is that

Claim. mintime(N'_{0,0'}) = 2 iff \phi is satisfiable and \phi' is unsatisfiable.

Proof. If mintime(N'_{0,0'}) = 2, this implies that \phi is satisfiable, for if it was not
satisfiable then for no run, all the processes corresponding to \phi could reach n_{sep}
and therefore the negotiation could not complete and hence MinTime would be
infinite. Also \phi' is unsatisfiable because if it would have been satisfiable then
there would have been a final run in which the processes after reaching n_{sep}
choose the outcome r from n_{sep} and complete the negotiation. The time for that
run would be 1 unit and therefore the mintime(N'_{0,0'}) \neq 2.

For the other side of the implication, we can argue similarly that if \phi is satisfiable then the processes of N_0 would complete the structure above n_{sep}
and reach n_{sep} in 0 units of time. From there the processes would have to choose
the outcome r_{all} to reach n_f because otherwise, the run would not be final. The
time taken for the path would be 2 units. So total time associated will this run
will be 2 units which will also be the mintime(N'_{0,0'}). \hfill \Box

For equality decision problem of MaxTime, the proof is similar; only the
\gamma(L_{0,v_i}, r) = [2, 2] for all v_i \in \phi', \gamma(n_{all}, r_{all}') = [1, 1] and \gamma(n, r) = [0, 0] for
all other nodes. The question asked is maxtime(N'_{0,0'}) = 2 which is true if only
if \phi is satisfiable and \phi' is unsatisfiable. \hfill \Box

Finally, we consider a related problem of deciding if a bit of mintime(N) is
1 (or similarly with maxtime(N)). Perhaps surprisingly, we obtain that these
problems goes even beyond DP (the second level of the Boolean Hierarchy) and
is in fact hard for \Delta^P_2, which contains the whole Boolean Hierarchy:

**Theorem 4.** Given an acyclic deterministic timed negotiation and a positive
integer k, computing the k^{th} bit of the maximum/minimum execution time is
\Delta^P_2 complete.
Paritcipation changes are the following: The node $\text{Lone}_i$ from some change in arcs and participation of processes in nodes. $N_i$ has the same structure as that of $v_{true}$ at written as $2$ that lexicorgraphic value is same, first of all the observe that time taken $V_i$ node is essentially the one whose lexicographic value is same as $t_i$ final and takes time $\phi$ which there is no change). We now define timing function $Tseitin$ transformation. Let the new variables introduced be called $\text{maxtime}, \text{mintime}$ decision problem of $x$ in lexicographic value will be $2$ which is same as $t_i$ for which there is no change. We first show that $\text{maxtime}/\text{mintime}$ show the problem of whether $\text{maxtime}$ $\text{ mintime}$ complete. $\Delta$ because odd or even is the same as the last bit. We first notice that it suffices to complete $\Delta_2$ hard. This is because odd or even is the same as the last bit. We first show that $\text{maxtime}(N)$ $= \text{odd}$ is $\Delta_2$ complete.

Consider the following problem: Given a Boolean formula $\phi(x_1, x_2, ... x_n)$, is $x_n = 1$ in the lexicographically largest satisfying assignment of $\phi$?

The above problem is known to be $\Delta_2$ complete [14] and we reduce it to the decision problem of $\text{maxtime}(N) = \text{odd}$. First, we convert $\phi$ to $3$-CNF form using Tseitin transformation. Let the new variables introduced be called $t_1, t_2, ..., t_k$. So $\phi(x_1, x_2, ..., x_n)$ is equisatisfiable to $3$-CNF $\phi'(v_1, v_2, ..., v_n, v_{n+1}, ..., v_{n+k})$ where $v_i = x_i$ for $i \leq n$ and $v_i = t_i$ for $i > n$. We convert $\phi$ to a negotiation $N_0, N_0'$. $N_0'$ has the same structure as that of $N_0$ which was constructed in Theorem 3 apart from some change in arcs and participation of processes in nodes. Participation changes are the following: The node $\text{Lone}_{v_i}$ associated with each variable $v_i$ of $\phi'$ now involve two processes namely $V_i$ and $V_{i-1}$. ($\text{Lone}_{v_i}$ has only $V_i$ as process). Both of the outcomes, $\text{true}$ and $\text{false}$ associated with $\text{Lone}_{v_i}$ take $V_{i-1}$ to $n_i$ while $\text{true}$ takes $V_i$ to $T\text{Lone}_{v_i,1}$ and $\text{false}$ takes $V_i$ to $F\text{Lone}_{v_i,1}$. Change in arcs is the following: The outcome $v_{ton}$ of $F\text{Lone}_{v_i, r}$ where $r = |S_i, t|$ and $T\text{Lone}_{v_i, r'}$ where $r' = |S_i, t|$ takes $V_i$ to $\text{Lone}_{v_i+1}$. (Except for $i = n + k$ for which there is no change). We now define timing function $\gamma$ as follows:

- $\gamma(\text{Lone}_{v_i}, \text{true}) = [2^{n-i}, 2^{n-i}]$ for all $i \leq n$ and
- $\gamma(n, r) = 0$ for all other combination of nodes and outcomes.

The claim is that $\text{maxtime}(N_0') = \text{odd}$ iff $x_n = 1$ in the lexicographically largest satisfying assignment of $\phi$.

To prove the claim, we prove a stronger outcome that there is a run which is final and takes time $t$ iff there is a satisfying assignment to $\phi$ whose lexicographic value is same as $t$ in binary.

To prove the forward implication, consider any run $\sigma$ which is final. Now, just like the proof in 3, the process $V_i$ must have choosen either $\text{true}$ or $\text{false}$ at node $\text{Lone}_{v_i}$. The assignment $f$, corresponding to this outcome chosen by each $V_i$ is essentially the one whose lexicographic value is same as $t$. The fact that this assignment is satisfiable follows from the proof of theorem 3. To show that that lexicographic value is same, first of all the observe that time taken $f$ can be written as $2^{n-i_1} + 2^{n-i_2} + ... + 2^{n-i_k}$ where $V_i$ are those processes which chose $\text{true}$ at $\text{Lone}_{v_i}$. Moreover $i_j \leq n$, which implies that all these variables are also present in $\phi$. Also the the contribution of a variable $x_{ij}$ (which is same as $v_{ij}$) in lexicographic value will be $2^{n-i_j}$ which is same as its contribution in $t$. Hence the forward implication.
Fig. 5. A part of $\mathcal{N}_\phi$. Here if $i > n$, then timing with arcs $\text{true}$ and $\text{false}$ will be $[0, 0]$. 
For backward implication, consider any satisfiable assignment \( f \) of \( \phi \). Since \( \phi \) and \( \phi' \) are equisatisfiable, hence there will exist an satisfiable assignment \( f' \) to \( \phi' \), such that \( f'(x_i) = f(v_i) \) for \( i \leq n \). Now following the proof of Thm. 3, it is easy see that the run \( \sigma \) corresponding to the assignment \( f' \) will be final. Moreover the time taken for the path will be \( 2^{n-i_1} + 2^{n-i_2} + \ldots + 2^{n-i_k} \) where \( f'(v_i) = \text{true} \). Since all these \( i_j \leq n \), these variables will also be present in \( \phi \) and their contribution in lexicographic value of \( f \) would also be \( 2^{n-i_j} \). And hence the backward implication.

This proves the claim and shows that \( \maxtime(N_{\phi'}) = \text{odd} \) if \( x_n = 1 \) in the lexicographically largest satisfying assignment of \( \phi \). \( \square \)

Finally, we note that if we were interested in the optimization and not the decision variant of the problem, the above proof can be adapted to show that the optimization variants are \( \textbf{OptP-Complete} \) (as defined in [14]).

**Appendix B: Sound Negotiations**

Sound negotiations are negotiations in which every run can be extended to a final run, as in Fig. 1. In this section, we show that \( \maxtime(N) \) can be computed in PTIME for sound negotiations, hence giving PTIME complexities for the \( \maxtime(N) \leq T? \) and \( \maxtime(N) = T? \) questions. However, we show that \( \mintime(N) \leq T \) is NP-complete for sound negotiations, and that \( \mintime(N) = T \) is DP-complete, even if \( T \) is given in unary.

Consider the graph \( G_N \) of a negotiation \( N \). Let \( \pi = (n_0, (p_0, r_0), n_1) \cdots (n_k, (p_k, r_k), n_{k+1}) \) be a path of \( G_N \). We define the **maximal execution time** of a path \( \pi \) as the value \( \delta^+(\pi) = \sum_{i \in 0..k} \gamma^+(n_i, r_i) \). We say that a path \( \pi = (n_0, (p_0, r_0), n_1) \cdots (n_k, (p_k, r_k), n_{k+1}) \) is a path of some run \( \rho = (M_1, \mu_1) (n_1, r_1^*) \cdots (M_k, \mu_k) \) if \( r_0, \ldots, r_k \) is a subword of \( r_1^*, \ldots, r_k^* \).

**Lemma 1.** Let \( N \) be an acyclic and sound timed negotiation. Then \( \maxtime(N) = \max_{\pi \in \text{Paths}(G_N)} \delta^+(\pi) + \gamma^+(n_f, r_f) \).

**Proof.** Let us first prove that \( \maxtime(N) \geq \max_{\pi \in \text{Paths}(G_N)} \delta^+(\pi) + \gamma^+(n_f, r_f) \).

Consider any path \( \pi \) of \( G_N \), ending in some node \( n \). First, as \( N \) is sound, we can compute a run \( \rho_\pi \) such that \( \pi \) is a path of \( \rho_\pi \), and \( \rho_\pi \) ends in a configuration in which \( n \) is enabled. We associate with \( \rho_\pi \) the timed run \( \rho_\pi^+ \) which associates to every node the latest possible execution date. We have easily \( \delta(\rho_\pi^+) \geq \delta(\pi) \), and then we obtain \( \max_{\pi \in \text{Paths}(G_N)} \delta(\rho_\pi^+) \geq \max_{\pi \in \text{Paths}(G_N)} \delta(\pi) \). As \( \maxtime(N) \) is the maximal duration over all runs, it is hence necessarily greater than \( \max_{\pi \in \text{Paths}(G_N)} \delta(\rho_\pi^+) + \gamma^+(n_f, r_f) \).

We now prove that \( \maxtime(N) \leq \max_{\pi \in \text{Paths}(G_N)} \delta^+(\pi) + \gamma^+(n_f, r_f) \).

Take any timed run \( \rho = (M_1, \mu_1) (n_1, r_1^*) \cdots (M_k, \mu_k) \) of \( N \) with a unique maximal node \( n_k \). We show that there exists a path \( \pi \) of \( \rho \) such that \( \delta(\rho) \leq \delta^+(\pi) \) by induction on the length \( k \) of \( \rho \). The initialization is trivial for \( k = 1 \). Let \( k \in \mathbb{N} \).

Because \( n_k \) is the unique maximal node of \( \rho \), we have \( \delta(\rho) = \max_{p \in P_{n_k}} \mu_{k-1}(p) + \)}
\(\gamma^+(n_k, r_k)\). We choose one \(p_{k-1}\) maximizing \(\mu_{k-1}(p)\). Let \(\ell < k\) be maximal index of a decision involving process \(p_{k-1}\) (i.e. \(p_{k-1} \in P_{n_k}\)). Now, consider the timed run \(\rho'\) subword of \(\rho\), but with \(n_\ell\) as unique maximal node (that is, it is \(\rho\) where nodes \(n_i, i > \ell\) has been removed, but also where some nodes \(n_i, i < \ell\) have been removed if they are not causally before \(n_\ell\) (in particular, \(P_{n_i} \cap P_{n_\ell} = \emptyset\)). By definition, we have that \(\delta(\rho) = \delta(\rho') + \gamma^+(n_\ell, r_\ell) + \gamma^+(n_k, r_k)\). We apply the induction hypothesis on \(\rho'\), and obtain a path \(\pi'\) of \(\rho'\) ending in \(n_\ell\) such that \(\delta(\rho') + \gamma^+(n_\ell, r_\ell) \leq \delta^+(\pi')\). It suffices to consider the path \(\pi = \pi'(n_\ell, (p_{k-1}, r_\ell, n_k))\) to prove the inductive step \(\delta(\rho) \leq \delta^+(\pi) + \gamma^+(n_k, r_k)\).

Thus \(\text{maxtime}(\mathcal{N}) = \max \delta(\rho) \leq \max_{\pi \in \text{Paths}(\mathcal{G}_N)} \delta^+(\pi) + \gamma^+(n_f, r_f). \quad \Box\)

Lemma 1 gives a way to evaluate the maximal execution time. This amounts to finding a path of maximal weight, which is a standard PTIME graph problem that can be solved using standard max-cost calculation.

**Proposition 3.** Computing the maximal execution time for an acyclic sound negotiation \(\mathcal{N} = (\mathcal{N}, n_0, n_f, \mathcal{X})\) can be done in time \(O(|\mathcal{N}| + |\mathcal{X}|)\).

**Proof.** First of all, we compute a topological order \(\prec\) on nodes of the graph \(\mathcal{G}_N\), that is for all \(n' \in \mathcal{X}(n, r)\), we have \(n < n'\). This can be done in \(O(|\mathcal{N}| + |\mathcal{X}|)\) [3]. Then, we follow the total order \(\prec\) on nodes of \(\mathcal{G}_N\) and attach to each node \(n\) a maximal time \(\delta^+(n)\) for runs ending at node \(n\) in the following way: \(\delta^+(n_0) = 0\) and for each node \(n\), we let \(\delta^+(n) = \max_{n' \in \mathcal{X}(n, r)} \delta^+(n')\). It is easy to see that \(\delta^+(n)\) is the maximal \(\delta(\pi)\) over all paths \(\pi\) from \(n_0\) to \(n\). As every transition of \(\mathcal{G}_N\) is considered only once, the computation of \(\delta^+\) can be done in \(O(|\mathcal{N}| + |\mathcal{X}|)\). It then suffices to return \(\delta^+(n_f) + \gamma^+(n_f, r_f)\). \(\Box\)

A direct consequence is that \(\text{maxtime}(\mathcal{N}) \leq T\) and \(\text{maxtime}(\mathcal{N}) = T\) problems can be solved in polynomial time when \(\mathcal{N}\) is. Notice that if \(\mathcal{N}\) is deterministic but not sound, then Lemma 1 does not hold: we only have an inequality. We now turn to \(\text{mintime}(\mathcal{N})\). We show that it is strictly harder to compute for sound negotiations than \(\text{maxtime}(\mathcal{N})\).

**Theorem 5.** \(\text{mintime}(\mathcal{N}) \leq T\) is NP-complete in the strong sense for sound acyclic negotiations, even if \(\mathcal{N}\) is very weakly non-deterministic.

**Proof.** First, we can decide \(\text{mintime}(\mathcal{N}) \leq T\) in NP. Indeed, one can guess a final (untimed) run \(\rho\) of size \(\leq |\mathcal{N}|\), consider \(\rho^-\) the timed run corresponding to \(\rho\) where all outcomes are taken at the earliest possible dates, and compute in linear time \(\delta(\rho^-)\), and check that \(\delta(\rho^-) \leq T\).

The hardness part is obtained by reduction from the Bin Packing problem. We give a set \(U\) of items, a size \(s(u) \in \mathbb{N}\) for each \(u \in U\), a positive integer \(B\) defining a bin capacity. The bin packing problem asks whether there exists a partition of \(U\) into \(k\) disjoint subsets \(U_1, U_2...U_k\) such that the sum of sizes of items in each \(U_i\) is smaller or equal to \(B\). Bin Packing is known to be NP-Complete [11] in the strong sense, that is even if the constants are given in unary. Let us now show that every instance of Bin Packing can be reduced to a min-time problem for very-weakly non-deterministic sound negotiations.
Given a set $U$ of items, a bin capacity $B$ and number $k$ of bins, we build a
timed negotiation $N_{U,k}$ with $k$ processes $u_{i,1}, u_{i,2}, \ldots, u_{i,k}$ for each item $u_i \in U$,
and $k$ additional processes $v_1, v_2, \ldots, v_k$. The timing of a process $v_j$ will encode the
total size of items put in the bin $i$. We then show that Bin Packing with items
$U$, $k$ bins, and a bound $B$ has a solution iff $\text{mintime}(N_{U,k}) \leq B$.

We describe the negotiation $N_{U,k}$ layer by layer. In total we will have $|U|+1$
layers: intuitively, we will consider one item in each layer, and make one global
decision to decide in which bin this item goes. The first layer has only the initial
node $n_0$. The set of processes involved in $n_0$ is the set of all processes. The
outcomes from the initial node are $r_{1,1}, \ldots, r_{1,k}$, which tell in which bin $1, \ldots, k$
the first item is placed. Outcome $r_{1,i}$ leads process $u_{i,1}$ and $v_i$ to node YES$^1_i$.

It leads processes $v_{j,i}$ and $v_j$ to NO$^1_j$ for every $j \neq i$. Last, it leads all other
processes in $\{u_{j,m} \mid j > 1, 1 \leq m \leq k\}$ to node $n_1$. Intuitively, moving to node
YES$^1_i$ means that item $u_1$ is placed in bin $i$. The second layer has $2k + 1$ nodes:
YES$^2_{1} \ldots$ YES$^2_{k}$, NO$^2_{1} \ldots$ NO$^2_{k}$ and $n_1$. The timing of outcome $r_{1,i}$ from node $n_0$
is $\gamma(n_0, r_{1,i}) = [0, 0]$.

Inductively, layer $i$ is defined as in Fig 6. Node $n_i$ contains processes $u_{j,\ell}$ for
all $j > i$ and all $\ell$. It is similar to $n_0$, with outcome $r_{i+1,1}, \ldots, r_{i+1,k}$. Outcome
$r_{i+1,\ell}$ leads process $u_{i+1,\ell}$ to node YES$^{i+1}_\ell$, and process $u_{i+1,j}$ to NO$^{i+1}_j$ for all
$j \neq \ell$. Other processes $u_{i',j}$ with $i' > i + 1$ are sent to $n_{i+1}$. The associated
timings are $[0, 0]$.

Node Garbage$_i$ collects all nodes $u_{\ell,j}$ with $\ell < i$. There is a unique outcome,
with associated timing $[0, 0]$, leading all processed to Garbage$_{i+1}$.

Node YES$^i_j$ has a unique outcome $r$, with timing $\gamma(\text{YES}^i_j, r) = [s(u_i), s(u_i)]$, and with $\mathcal{A}(\text{YES}^i_j, r) = \{\text{YES}^{i+1}_j, \text{NO}^{i+1}_j\}$. That is, node YES$^i_j$ is non determinis-
tic, and it awaits the decision from $u_{i+1,j}$ to known whether it will go to YES$^{i+1}_j$
or to NO$^{i+1}_j$. Last, $u_{i,j}$ is sent to node Garbage$_{i+1}$. This allows each nodes to
have at least one deterministic process, as $v_i$ only are non-deterministic.

In the same way, NO$^i_j$ has a unique outcome $r$, timed with $\gamma(\text{NO}^i_j, r) = [0, 0]$, and with $\mathcal{A}(\text{NO}^i_j, r) = \{\text{YES}^{i+1}_j, \text{NO}^{i+1}_j\}$. It sends process $u_{j,i}$ to Garbage$_{i+1}$.
The last layer has only node $n_f$. Nodes $Yes_i^j$ and $No_i^j$ both have a single outcome which take all their processes to $n_f$. The timing function $\gamma$ is defined as follows: $\gamma(Yes_i^j, r_i) = [s(u_i), s(u_i)]$ and $\gamma(n, r) = [0, 0]$ for all other node and outcome $r$.

We now prove that $\text{MinTime}(N_{U,k}) \leq B$ iff the answer to Bin Packing is positive. The maximal execution time over runs $\rho$ of $N_{U,k}$ is the maximal value of all valuations $\mu(v_j)$ and $\mu(u_{i,j})$, with $i \in 1...|U|$, $j \in 1..k$. Take the valuation $\mu$ at the last step before $(n_f, r_f)$. Consider $t = \max_{j} \mu(v_j)$. We have easily that $\mu(u_{i,j}) \leq t$ for all $i, j$ by construction, because each $u_{i,j}$ had the same timing as $v_j$ before reaching a garbage node. Now, we have $\mu(v_j) = \sum_{\gamma(Yes_i^j, r_i) \in \rho} s(u_i)$. Hence, $\delta(\rho) = \max_{j \in 1..k} \mu(v_j)$. That is, $\text{mintime}(N(U, B, k) \leq B$ iff there is a path $\rho$ such that $\mu(v_j) = \sum_{\gamma(Yes_i^j, r_i) \in \rho} s(u_i) \leq B$ for all $j$, i.e. there exists a valuation such that each item is in one bin, and no bin exceeds its bound $B$.

Last, we now show that $N_{U,k}$ is a very weakly non-deterministic, sound and layered negotiations. First, the only processes that have non-deterministic transitions are processes $v_1, \ldots, v_k$, from $Yes_i^j$ and $No_i^j$ nodes. However, both nodes also have the same deterministic process $u_i^j$. Thus $N_{U,k}$ is very weakly non-deterministic. Let us now prove soundness. The only choices are made from node $n_1$, the rest just follow in a unique way. From any configuration $M$, let $i$ such that $M(u_{i+1,j}) = \{n_i\}$ for some $j$. By construction, $i$ is unique. We can then do steps $r_{i+1,1} \ldots r_{m,1}$, that is choosing to place items $i+1, \ldots, n$ to the first bin. The steps from other processes are uniquely derived, and all processes reach $n_f$. The layeredness comes from the definition. Actually, $N_{U,k}$ is $2k + 2$-layered, for $k$ the number of bins. However, as $k$ is part of the input, it does not fall in our $k$-layered restriction. \(\square\)

We show that $\text{mintime}(N) = T$ is harder to decide than $\text{mintime}(N) \leq T$:

**Proposition 4.** The $\text{mintime}(N) = T$? decision problem is DP-complete for sound acyclic negotiations, even if it is very weakly non-deterministic.

**Proof.** The reduction is very similar to proof of Proposition 2. First, we define the complement of Bin-Packing Problem, **Non-Bin-Packing Problem**: Given a set $U$ of items, a size $s(u) \in \mathbb{N}$ for each $u \in U$, a positive integer bin capacity $B$, does for any partition $U$ into $k$ disjoint subsets $U_1, U_2 \ldots U_k$ there exist a subset $U_i$ such that the sum of sizes of the items in $U_i$ is more than $B$?

Since the Bin-Packing Problem is NP-Complete, so the Non-Bin-Problem is co-NP Complete. Now consider the following **Bin-Non-Bin Problem**:

Given two instances of Bin-Packing parameters, $P_1 = (U_1, s_1, B_1, k_1)$ and $P_2 = (U_2, s_2, B_2, k_2)$, does $P_1$ satisfy Bin-Packing Problem and $P_2$ satisfy Non-Bin-Packing Problem?

Bin-Non-Bin Problem is DP-Complete, so we reduce it to our equality decision problem of min time. First, we construct the negotiations $N_{U_1', B_1', k_1}$ and $N_{U_2', B_2', k_2}$ like in proof of Theorem 5, but only after tripling each $s(u)$ in $U_1$ and doubling each $s(u)$ in $U_2$. Likewise we triple $B_1$ and double $B_2$, so that new
\[ B'_1 = 3 \times B_1 \text{ and } B'_2 = 2 \times B_2. \]

In \( \mathcal{N}'_{U'_1, B'_1, k_1} \), we add a new node \( n_0 \) with a single outcome \( r \) which now acts as the first node. The older \( n_0 \) is now called \( n'_0 \). We also add a new process \( a_1 \), which goes to another new node \( n_{a_1} \) (has only \( a_1 \) as process) from \( n_0 \) for its single outcome \( r \). Outcome \( r \) sends all other processes from \( n_0 \) to \( n'_0 \). Node \( n_{a_1} \) has a single outcome \( r_1 \) which takes \( a_1 \) to \( n_f \). Also, \( \gamma(n_{a_1}, r_1) = [3 \times B_1 + 1, 3 \times B_1 + 1] \) while \( \gamma(n_0, r) = [0, 0] \).

Similarly in \( \mathcal{N}'_{U'_2, B'_2, k_2} \), we add a new node \( n_0 \) with two outcomes \( r \) and \( r_{new} \) which now acts as the first node. The older \( n_0 \) is now called \( n'_0 \). We also add a new process \( a_2 \), which goes to another new node \( n_{a_2} \) (has only \( a_2 \) as process) from \( n_0 \) for its outcome \( r \). Outcome \( r \) sends all other processes from \( n_0 \) to \( n'_0 \). Node \( n_{a_2} \) has a single outcome \( r_2 \) which takes \( a_2 \) to \( n_f \). Also, \( \gamma(n_{a_2}, r_2) = [2 \times B_1, 2 \times B_1] \) while \( \gamma(n_0, r) = [0, 0] \). For outcome \( r_{new} \) of \( n_0 \), all processes (including \( a_2 \)) directly go to \( n_f \). Also, \( \gamma(n_0, r_{new}) = [2 \times B_2 + 1, 2 \times B_2 + 1] \).

Now we merge the two negotiations \( \mathcal{N}'_{U'_1, B'_1, k_1} \) and \( \mathcal{N}'_{U'_2, B'_2, k_2} \) in the same way as we merged in Corollary 2, merging the \( n_f \) of \( \mathcal{N}'_{U'_1, B'_1, k_1} \) with \( n_0 \) of \( \mathcal{N}'_{U'_2, B'_2, k_2} \) and making other similar changes we did in Corollary 2. We call this new negotiation \( \mathcal{N}'_{p'_1, p'_2} \). Note the negotiation \( \mathcal{N}'_{p'_1, p'_2} \) is sound as well as very weakly non-deterministic.

The claim is that \( \text{mintime}(\mathcal{N}'_{p'_1, p'_2}) = 3 \times B_1 + 2 \times B_2 + 2 \) iff \((P_1, P_2)\) satisfy Bin-Non-Bin Problem.

We first show the reverse implication i.e if \((P_1, P_2)\) satisfy Bin-Non-Bin Problem, then \( \text{mintime}(\mathcal{N}'_{p'_1, p'_2}) = 3 \times B_1 + 2 \times B_2 + 2 \). Since \( P_1 \) is satisfiable, so the min-time to complete the structure above \( n_{sep} \) of \( \mathcal{N}'_{p'_1, p'_2} \) is \( 3 \times B_1 + 1 \). This is because all the processes corresponding to \( \mathcal{N}'_{U'_1, B'_1, k_1} \) take \(( \leq 3 \times B \) time to reach \( n_{sep} \) (because \( P_1 \) is satisfies Bin-Packing) while \( a_1 \) takes \( 3 \times B_1 + 1 \) units of time. After reaching \( n_{sep} \), processes can now take either outcome \( r_2 \) or \( r_{new} \). If processes choose outcome \( r_2 \), then the time taken by any final run will be \(( \geq 2 \times (B_2 + 1) \) because \( P_2 \) satisfies Non-Bin-Packaging. On the other hand, if processes choose \( r_{new} \) to reach \( n_f \), then the time taken will be \( 2 \times B_2 + 1 \). So it is clear min-time for part below \( n_{sep} \) is \( 2 \times B_2 + 1 \). Thus, overall the \( \text{mintime}(\mathcal{N}'_{p'_1, p'_2}) = 3 \times B_1 + 2 \times B_2 + 2 \).

For forward implication, we consider all four scenarios of \((P_1, P_2)\) and argue that \( P_1 \) satisfies Bin-Packaging and \( P_2 \) satisfies Non-Bin-Packaging is the only possibility. First let’s assume that \( P_1 \) does not satisfy Bin-Packaging. Then the min-time to complete the structure above \( n_{sep} \) is \(( \geq 3 \times (B_1 + 1) \) This is because processes corresponding to \( \mathcal{N}'_{U'_1, B'_1, k_1} \) take at least \( 3 \times (B_1 + 1) \) time to reach \( n_{sep} \) while \( a_1 \) take \( 3 \times B_1 + 1 \). Now since the min-time which can be taken to reach \( n_f \) from \( n_{sep} \) in either case whether \( P_2 \) satisfies Non-Bin-Packaging or not is \(( \geq 2 \times B_2 \) so the min time to complete \( \mathcal{N}'_{p'_1, p'_2} \) \( \geq 3 \times B_1 + 2 \times B_2 + 3 \). Hence this shows that \( P_1 \) satisfies Bin-Packaging. This also shows the final run corresponding to min-time of \( \mathcal{N}'_{p'_1, p'_2} \) takes exactly \( 3 \times B_1 + 1 \) units of time to reach \( n_{sep} \) from \( n_0 \) (i.e. all processes have reached \( n_{sep} \)) if \( \text{mintime}(\mathcal{N}'_{p'_1, p'_2}) = 3 \times B_1 + 2 \times B_2 + 2 \).

Now if we assume the \( P_2 \) does not satisfy Non-Bin-Packaging, then the min-time to reach \( n_f \) from \( n_{sep} \) is \( 2 \times B_2 \). And we already know that min-time to reach \( n_{sep} \)
from \( n_0 \) is \( 3 \cdot B_1 + 1 \). So \( \text{mintime}(N_{P_1', P_2'}) = 2 \cdot B_2 + 3 \cdot B_1 + 1 \). Hence this leaves us with the only case when \( P_1 \) satisfies Bin-Packing and \( P_2 \) satisfies Non-Bin-Packing for which we already know that the min time taken is \( 3 \cdot B_1 + 2 \cdot B_2 + 2 \) from the reverse implication. □

An open question is whether the minimal execution time can be computed in PTIME if the negotiation is both sound and deterministic. The reduction to bin packing does not work with deterministic (and sound) negotiations.

Appendix C: \( k \)-Layered Negotiations

In the previous sections, we have considered sound negotiations, and deterministic negotiations. For both classes, computing the minimal execution time cannot be done in PTIME (unless \( \text{NP} = \text{PTIME} \)), even if constants are given in unary. In this section, we consider \( k \)-layeredness (see Section 2), a syntactic property that can be efficiently verified (it suffices to compute the depth of each node, which can be done in polynomial time).

8.1 Algorithmic properties

Let \( k \) be a fixed integer. We first show that Reachability, Soundness and maximum execution time can be checked in PTIME for \( k \)-layered negotiations (the two first results were stated in Section 2). Let \( N_i \) be the set of nodes at layer \( i \). We define for every layer \( i \) the set \( S_i \) of subsets of nodes \( X \subseteq N_i \) which can be jointly enabled and such that for every process \( p \), there is exactly one node \( n(X, p) \) in \( X \) with \( p \in n(X, p) \). Formally, we define \( S_i \) inductively. We start with \( S_0 = \{ n_0 \} \). We then define \( S_{i+1} \) from the contents of layer \( S_i \): we have \( Y \in S_{i+1} \) iff \( \bigcup_{n \in Y} P_n = P \) and there exist \( X \in S_i \) and an outcome \( r_m \in R_m \) for every \( m \in X \), such that \( n \in X(n(X, p), p, r_m) \) for each \( n \in Y \) and \( p \in P_n \).

**Theorem 6.** Let \( k \in \mathbb{N}^+ \). Checking reachability, soundness and computing the maximum execution time for a \( k \)-layered acyclic negotiation \( N \) can be done in PTIME. More precisely, the worst-case time complexity is \( O(|P| \cdot |N|^{k+1}) \).

**Proof (Sketch of Proof).** The algorithm has the same form for all problems. The basis is to compute \( S_0 \) layer by layer, by following its inductive definition. The set \( S_i \) is of size at most \( 2^k \), as \( |N_i| < k \) by definition of \( k \)-layerness. Knowing \( S_i \), it is easy to build \( S_{i+1} \) by induction. This takes time at most \( O(|P| |N|^{k+1}) \). We need to consider all \( k \)-uple of outcomes for each layer. There can be \( |N|^k \) such tuples. We need to do that for all processes (\( |P| \)), and for all layers (at most \( |N| \)).

For reachability, we just need to check whether layer \( S_d = \{ n_f \} \), where \( d \) is the depth of \( n_f \).

For soundness, let us denote by \( \text{Next}(X, (r_n)_{n \in X}) \) the set of nodes that are successors of nodes in \( X \) after outcomes \( (r_n)_{n \in X} \). We need to check that for all layer \( i \), for all set \( X \in S_i \) and all tuple of outcomes \( (r_n)_{n \in X} \), there...
is a $Y \subseteq \text{Next}(X, (r_n)_{n \in X})$ such that every process $p$ is in exactly one node $n(Y, p)$ of $Y$. All nodes of $\text{Next}(X, (r_n)_{n \in X})$ are at depth $i+1$, and thus there are at most $k$ nodes in $\text{Next}(X, (r_n)_{n \in X})$. There are thus at most $2^k$ subset $Y \subseteq \text{Next}(X, (r_n)_{n \in X})$ and we can check them one by one.

For maximal execution time, we keep for each subset $X \in S_i$ and each node $n \in X$, the maximal time $f_i(n, X) \in \mathbb{N}$ associated with $n$ and $X$. From $S_{i+1}$ and $f_i$, we inductively compute $f_{i+1}$ in the following way: for all $X \in S_i$ with successor $Y \in S_{i+1}$ for outcomes $(r_p)_{p \in P}$, we denote $f_{i+1}(Y, n, X) = \max_{p \in P(n)} f_i(X, n(X, p)) + \gamma^+ (n(X, p), r_p)$. If there are several choices of $(r_p)_{p \in P}$ leading to the same $Y$, we take $r_p$ with the maximal $f_i(X, n(X, p)) + \gamma^+ (n(X, p), r_p)$.

We then define $f_{i+1}(Y, n) = \max_{X \in S_i} f_{i+1}(Y, n, X)$. Again, the initialization is trivial, with $f_0(\emptyset, \emptyset) = 0$. The maximal execution time of $\mathcal{N}$ is $f(\{f_j\}, n_f)$. That is, for all nodes (at most $|\mathcal{N}|$), we have to consider every $k$-uple of outcomes, and there are at most $\mathcal{N}^k$ of them, and every process to compute the max, and the complexity is still in $O(|P| \cdot |\mathcal{N}|^{k+1})$.

We can bound the complexity precisely by $O(d|\mathcal{N}| \cdot C(\mathcal{N}) \cdot ||R||^{k^*})$, with:

- $d(\mathcal{N}) \leq |\mathcal{N}|$ the depth of $n_f$, that is the number of layers of $\mathcal{N}$, and $||R||$ is the maximum number of outcomes of a node,
- $C(\mathcal{N}) = \max_i |S_i| \leq 2^k$, which we will call the number of contexts of $\mathcal{N}$, and which is often much smaller than $2^k$.
- $k^* = \max_{X \in \bigcup_i S_i} |X| \leq k$. We say that $\mathcal{N}$ is $k^*$-thread bounded, meaning that there cannot be more than $k^*$ nodes in the same context $X$ of any layer.

Usually, $k^*$ is strictly smaller than $k = \max_i |N_i|$, as $N_i = \bigcup_{X \in S_i} X$.

Consider again the Brexit example Figure 1. We have $(k+1) = 7$, while we have the depth $d(\mathcal{N}) = 6$, the negotiation is $k^*$, $3$-thread bounded ($k^*$ is bounded by the number of processes), and the number of contexts is at most $C(\mathcal{N}) = 4$ (EU chooses to enforce backstop or not, and Pa chooses to go to court or not).

### 8.2 Minimal Execution Time

As with sound negotiations, computing minimal time is much harder than computing the maximal time for $k$-layered negotiations:

**Theorem 7.** Let $k \geq 6$. The $\text{Min} \leq T$ problem is NP-Complete for $k$-layered acyclic negotiations, even if the negotiation is sound and very weakly non-deterministic.

**Proof.** One can guess in polynomial time a final run of size $\leq |\mathcal{N}|$. If the execution time of this final run is smaller than $T$ then we have found a final run witnessing $\text{Min}(\mathcal{N}) \leq T$. Hence the problem is in NP.

Let us now show that the problem is NP-hard. We proceed by reduction from the knapsack decision problem. Let us consider a set of items $U = \{u_1, \ldots, u_n\}$ of respective values $v_1, \ldots, v_n$ and weight $w_1, \ldots, w_n$ and a knapsack of maximal capacity $W$. The knapsack problem asks, given a value $V$ whether there exists a subset of items $U' \subseteq U$ such that $\sum_{u_i \in U'} v_i \geq V$ and such that $\sum_{u_i \in U'} w_i \leq W$. 

33
We build a negotiation with \(2n\) processes \(P = \{p_1, \ldots, p_{2n}\}\). Intuitively, \(p_i, i \leq n\) will serve to encode the value as timing, while \(p_i, i > n\) will serve to encode the weight as timing. We set the set of nodes \(N = \{n_0, n_f\} \cup \{C_i \mid i \in 1..n\} \cup \{n_L,0,i, n_L,1,i, n_R,0,i, n_R,1,i \mid i \in 1..n\}\). Intuitively, node \(nL,1,i\) (resp. \(nR,1,i\)) will be used to remember that item \(i\) is placed in the knapsack and that its value (resp. weight) needs to be added. For all \(i\), node \(nL,1,i\) (resp. \(nR,1,i\)) has a unique possible outcome, \(b_i\) (resp. \(c_i\)). Nodes of the form \(nL,0,i\) remember that item \(i\) has not been placed in the knapsack, and they have outcome \(a_i\). Nodes of the form \(nR,0,i\) remember that item \(i\) has not been placed in the knapsack, and they all have outcome \(0\). This outcome does not change the execution time, matching the fact that the current weight and value of the knapsack is not increased.

Last, nodes of the form \(C_i\) will just remember the items that have already been considered. These nodes have two outputs, yes and no, telling whether the item \(i\) should be placed in the knapsack or not, consistently for weight and value processes.

We set \(P_{n_0} = P_{n_f} = P\), and for other nodes \(nL,0,i, P_{nL,0,i} = P_{nL,1,i} = \{p_1, \ldots, p_i\}\) and \(P_{nR,0,i} = P_{nR,1,i} = \{p_{n+1}, \ldots, p_{2n}\}\). Last \(P_{n_f} = \{p_1, \ldots, p_n, p_{n+1}, \ldots, p_{2n}\}\).

We define the transition relation as follows: \(X(n_0, yes, p_1) = \{nL,1,1\}\), and \(X(n_0, no, p_1) = \{nL,0,1\}\), such that process \(p_1\) remembers that the item is picked/notpicked.

In the same way, \(X(n_0, no, p_{n+1}) = \{nR,0,1\}\) and \(X(n_0, yes, p_{n+1}) = \{nR,1,1\}\) for process \(p_{n+1}\). Hence both process \(p_1, p_{n+1}\) will have the same information about whether the first item is picked or not. Finally, for every \(k \in 2..n\), we define \(X(n_0, no, p_k) = X(n_0, no, p_{k+n}) = X(n_0, yes, p_k) = X(n_0, yes, p_{k+n}) = \{C_1\}\).

Other layers are similar: for \(i \in 1..n\), we have \(X(C_i, no, p_1) = \{nL,0,i+1\}\)
\(X(C_i, yes, p_1) = \{nL,1,i+1\}\). Similarly, for every \(i \in 1..n\), \(X(C_i, no, p_{n+i}) = \{nR,0,i+1\}\), and \(X(C_i, yes, p_{n+i}) = \{nR,1,i+1\}\). We set \(X(C_i, no, p_j) = X(C_i, yes, p_j) = \{C_{i+1}\}\) for every \(j \in [i+1, n-1] \cup [n+i+1, 2n]\).

The most interesting set of transitions are to interface \(nL,0,i, nL,1,i, nR,0,i, nR,1,i\) with the next layer, in a non deterministic way because they don't know whether the next item will be picked or not:
\(X(nL,0,i, a_i, p_1) = X(nL,1,i, b_i, p_1) = \{nL,0,i+1, nL,1,i+1\}\)
for \(j \in 1..i\) and, \(X(nR,0,i, 0, p_j) = X(nR,1,i, c_i, p_j) = \{nR,0,i+1, nR,1,i+1\}\) for \(j \in n+i+1..n+i+n\).

Last, all processes synchronize on \(n_f\) by setting \(X(nL,0,n, 0, p_f) = X(nL,1,n, b_n, p_f) = X(nR,0,n, 0, p_f) = X(nR,1,n, c_n, p_f) = \{n_f\}\)
\(X(nR,0,n, 0, p_f) = X(nR,1,n, c_n, p_f) = \{n_f\}\)

We now have to set timing constraints for outcomes. Outcomes 0, yes and no are associated with \([0, 0]\). Outcome \(c_i\) is associated with \([w_i, w_i]\), the weight of \(w_i\). Last, outcome \(b_i\) is associated with a more complex function, such that \(\sum_i b_i \leq W\) iff \(\sum_i v_i \geq V\). For that, we set \([v_{max} - v_i, V] \cdot n \cdot W\) for outcome \(b_i\), where \(v_{max}\) is the largest value of an item, and \(V\) is the total value we want to reach at least. Also, we set \([v_{max} - v_i, V] \cdot n \cdot W\) for outcome \(a_i\). We set \(T = W\), the maximal weight of the knapsack.

Now, consider a final run \(\rho\) in \(N\). The only choice is about yes, no from \(C_i\). Let \(I\) be the set of indices such that yes is the outcome from all \(C_i\) in this path. We obtain \(\delta(\rho) = \max(\sum_{i \in I} a_i + \sum_{i \in I} b_i, \sum_{i \in I} c_i)\). We have \(\delta(\rho) \leq T = W\) iff \(\sum_{i \in I} w_i \leq W\), that is the sum of the weight are lower than \(W\), and
\[
\sum_{i \in I} (v_{\text{max}} - v_i) W + \sum_{i \in I} (v_{\text{max}} - v_i) W \leq W. \]
i.e. \[
\sum_{i \in I} v_i \geq V. \]
Hence, there exists a path \( \delta(\rho) \leq T = W \) iff there exists a set of items of weight less than \( W \) and of value more than \( V \).

So, given a knapsack of size \( n \), a value \( V \) and a weight limit \( W \) one can build a negotiation \( N^K_{\text{napp}} \) with \( O(3n + 2) \) nodes. We can encode all weights with \( \log(v_{\text{max}} \cdot W) + \log(n \cdot v_{\text{max}}) \) bits. One can notice that \( N^K_{\text{napp}} \) is 5-layered and sound.

However, it is not (weakly) non-deterministic because of nodes \( n_{L,0,i}, n_{L,1,i}, n_{R,0,i}, n_{R,1,i} \).
It is easy to add two processes \( V \) (resp. \( W \)), present in all nodes \( n_{L,0,i}, n_{L,1,i} \) (resp \( n_{R,0,i}, n_{R,1,i} \)), and make process \( P_i \) (resp. \( P_{n+i} \)) leave these nodes, deterministically leading to a new node garbage_{i+1} at layer \( i+1 \). Then the negotiation is very weakly deterministic, and 6-layered.

Following the same lines as for the proofs of Propositions 2 and 4, a consequence of Theorem 7 is that the \( \text{Min} = T \) problem is in \( \text{DP} \) for \( k \)-layered acyclic negotiations.
It is well known that Knapsack is weakly \( \text{NP} \)-hard, that is it \( \text{NP} \)-hard only when weights/values are given in binary. This means that Thm. 7 shows that minimum execution time \( \leq T \) is \( \text{NP} \)-hard only when \( T \) is given in binary. We can actually show that for \( k \)-layered negotiations, the \( \text{mintime}(N) \leq T \) problem can be decided in \( \text{PTIME} \) if \( T \) is given in unary (i.e. if \( T \) is not too large):
Theorem 8. Let $k \in \mathbb{N}$. Given a $k$-layered negotiation $N$ and $T$ written in unary, one can decide in PTIME whether the minimum execution time of $N$ is $\leq T$. The worst-case time complexity is $O(|N| \cdot |P| \cdot (T \cdot |N|)^k)$.

Proof. We will remember for each layer $i$ a set $T_i$ of functions $\tau$ from nodes $N_i$ of layer $i$ to a value in $\{1, \ldots, T, \perp\}$. Basically, we have $\tau \in T_i$ if there exists a path $\rho$ reaching $X = \{n \in N_i \mid f(n) \neq \perp\}$, and this path reaches node $n \in X$ after $\tau(n)$ time units. As for $S_i$, for all $p$, we should have a unique node $n(\tau, p)$ such that $p \in n(f, p)$ and $\tau(n(\tau, p)) \neq \perp$. Again, it is easy to initialize $T_0 = \{\tau_0\}$, with $\tau_0(n_0) = 0$, and $\tau_0(n) = \perp$ for all $n \neq n_0$.

Inductively, we build $T_{i+1}$ in the following way: $\tau_{i+1} \in T_{i+1}$ iff there exists a $\tau_i \in T_i$ and $r_p \in R_{n(\tau, p)}$ for all $p \in P$ such that for all $n$ with $\tau_{i+1}(n) \neq \perp$, we have $\tau_{i+1}(n) = \max_p \tau_i(n(\tau, p)) + \gamma(n(\tau, p), r_p)$.

We have that the minimum execution time for $N$ is $\min_{\tau \in T_i} \tau(n_f)$, for $n$ the depth of $n_f$. There are at most $T^k$ functions $\tau$ in any $T_i$, and there are at most $|N|$ layers to consider, giving the complexity. $\square$

As with Thm. 6, we can more accurately state the complexity as $O(d(N) \cdot C(N) \cdot ||R||^{k^*} \cdot T^{k^*-1})$. The $k^* - 1$ is because we only need to remember minimal functions $\tau \in T_i$; if $\tau'(n) \geq \tau(n)$ for all $n$, then we do not need to keep $\tau'$ in $T_i$.

In particular, for the knapsack encoding in the proof of Thm. 7, we have $k^* = 3$, $||R|| = 2$ and $C(N) = 4$.

Notice that if $k$ is part of the input, then the problem is strongly NP-hard, even if $T$ is given in unary, as e.g. encoding bin packing with $k$ bins result to a $k + 1$-layered negotiations.