



## Timed Negotiations

Sundararaman Akshay, Blaise Genest, Loïc Hérouët, Sharvik Mital

► **To cite this version:**

Sundararaman Akshay, Blaise Genest, Loïc Hérouët, Sharvik Mital. Timed Negotiations. 2019. hal-02337887

**HAL Id: hal-02337887**

**<https://hal.inria.fr/hal-02337887>**

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<sup>1</sup>, Blaise Genest<sup>2</sup>, Loic Helouet<sup>3</sup>, and Sharvik Mital<sup>1</sup>

<sup>1</sup> IIT Bombay, Mumbai, India {akshayss,sharky}@cse.iitb.ac.in

<sup>2</sup> Univ Rennes, CNRS, IRISA, Rennes, France blaise.genest@irisa.fr

<sup>3</sup> Univ Rennes, INRIA, Rennes, France loic.helouet@inria.fr

**Abstract.** Negotiations were introduced in [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 [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 [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 [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 [5, 4].

Recently, [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

41 in PTIME [7]. The decision procedure uses reduction rules. Reduction techniques  
42 were originally proposed for Petri nets [1, 8, 12, 17]. The main idea is to define  
43 transformations rules that produce a model of smaller size w.r.t. the original  
44 model, while preserving the property under analysis. In the context of negotia-  
45 tions, [7, 2] proposed a sound and complete set of soundness-preserving reduction  
46 rules and algorithms to apply these rules efficiently. The question of soundness  
47 for deterministic negotiations was revisited in [9] and showed NLOGSPACE-  
48 complete using anti patterns instead of reduction rules. Further, they show that  
49 the PTIME result holds even when relaxing determinism [9]. Negotiation games  
50 have also been considered to decide whether one particular process can force ter-  
51 mination of a negotiation. While this question is EXPTIME complete in general,  
52 for sound and deterministic negotiations, it becomes PTIME [13].

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

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

75 The first contribution of the paper shows that reachability (whether at least  
76 one run of a negotiation terminates) is NP-complete, already for (untimed) deter-  
77 ministic acyclic negotiations. This implies that computing minimal or maximal  
78 execution time for deterministic (but unsound) acyclic negotiations cannot be  
79 done in PTIME (unless NP=PTIME). We characterize precisely the complex-  
80 ities of different decision variants (threshold, equality, etc.), with complexities  
81 ranging from (co-)NP-complete to  $\Delta_2^P$ .

82 We thus turn to negotiations that are sound but not necessarily determinis-  
83 tic. Our second contribution is a new algorithm, not based on reduction rules,  
84 to compute the maximal execution time in PTIME for sound negotiations. It is  
85 based on computing the maximal execution time of critical paths in the nego-

86 tiations. However, we show that *minimal* execution time cannot be computed  
 87 in PTIME for sound negotiations (unless NP=PTIME): deciding whether the  
 88 minimal execution time is lower than  $T$  is NP-complete, even for  $T$  given in  
 89 unary, using a reduction from a Bin packing problem. This shows that minimal  
 90 execution time is harder to compute than maximal execution time.

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

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

## 115 2 Negotiations: Definitions and Brexit example

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

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

- 120 –  $N$  is a finite set of nodes. Each node is a pair  $n = (P_n, R_n)$  where  $P_n \subseteq P$   
 121 is a non empty set of processes participating in node  $n$ , and  $R_n$  is a finite  
 122 set of outcomes of node  $n$  (also called results), with  $R_{n_f} = \{r_f\}$ . We denote  
 123 by  $R$  the union of all outcomes of nodes in  $N$ .
- 124 –  $n_0$  is the first node of the negotiation and  $n_f$  is the final node. Every process  
 125 in  $P$  participates in both  $n_0$  and  $n_f$ .
- 126 – For all  $n \in N$ ,  $\mathcal{X}_n : P_n \times R_n \rightarrow 2^N$  is a map defining the transition relation  
 127 from node  $n$ , with  $\mathcal{X}_n(p, r) = \emptyset$  iff  $n = n_f, r = r_f$ . We denote  $\mathcal{X} : N \times P \times$

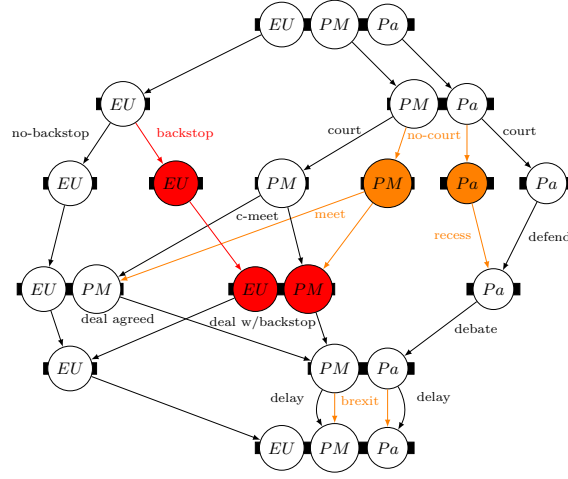


Fig. 1. A (sound but non-deterministic) negotiation modeling Brexit.

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

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

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

144 **Definition 2 (Deterministic negotiations).** A process  $p \in P$  is determinis-  
 145 tic iff, for every  $n \in N$  and every outcome  $r$  of  $n$ ,  $\mathcal{X}(n, p, r)$  is a singleton. A ne-  
 146 gotiation is deterministic iff all its processes are deterministic. It is weakly non-  
 147 deterministic [9] (called weakly deterministic in [2]) iff, for every node  $n$ , one of  
 148 the processes in  $P_n$  is deterministic. Last, it is very weakly non-deterministic [9]  
 149 (called weakly deterministic in [6]) iff, for every  $n$ , every  $p \in P_n$  and every out-  
 150 come  $r$  of  $n$ , there exists a deterministic process  $q$  such that  $q \in P_{n'}$  for every  
 151  $n' \in \mathcal{X}(n, p, r)$ .

152 In deterministic negotiations, once an outcome is chosen, each process knows  
 153 the next node it will be involved in. In (very)-weakly non-deterministic nego-  
 154 tiations, the next node might depend upon the outcome chosen in other nodes  
 155 by other processes. However, once the outcomes have been chosen for all cur-  
 156 rent nodes, there is only one next node possible for each process. Observe that  
 157 the class of deterministic negotiations is isomorphic to the class of free choice  
 158 workflow nets [10]. Coming back to example 1, the Brexit negotiation is non-  
 159 deterministic, because process  $PM$  is non-deterministic. Indeed, consider out-  
 160 comes  $c-meet$ : it allows two nodes, according to whether the backstop is enforced  
 161 or not, which is a decision taken by process  $EU$ . However, the Brexit negotiation  
 162 is very weakly non-deterministic, as the other processes are deterministic.

163 **Semantics:** A *configuration* [2] of a negotiation is a mapping  $M : P \rightarrow 2^N$ .  
 164 Intuitively, it tells for each process  $p$  the set  $M(p)$  of nodes  $p$  is ready to engage in.  
 165 The semantics of a negotiation is defined in terms of moves from a configuration  
 166 to the next one. The *initial*  $M_0$  and *final*  $M_f$  configurations, are given by  $M_0(p) =$   
 167  $\{n_0\}$  and  $M_f(p) = \emptyset$  respectively for every process  $p \in P$ . A configuration  $M$   
 168 *enables* node  $n$  if  $n \in M(p)$  for every  $p \in P_n$ . When  $n$  is enabled, a decision  
 169 at node  $n$  can occur, and the participants at this node choose an outcome  $r \in$   
 170  $R_n$ . The occurrence of  $(n, r)$  produces the configuration  $M'$  given by  $M'(p) =$   
 171  $\mathcal{X}(n, p, r)$  for every  $p \in P_n$  and  $M'(p) = M(p)$  for remaining processes in  $P \setminus P_n$ .  
 172 Moving from  $M$  to  $M'$  after choosing  $(n, r)$  is called a *step*, denoted  $M \xrightarrow{n,r} M'$ . A  
 173 *run* of  $\mathcal{N}$  is a sequence  $(n_1, r_1), (n_2, r_2) \dots (n_k, r_k)$  such that there is a sequence of  
 174 configurations  $M_0, M_1, \dots, M_k$  and every  $(n_i, r_i)$  is a step between  $M_{i-1}$  and  $M_i$ .  
 175 A run starting from the initial configuration and ending in the final configuration  
 176 is called a *final run*. By definition, its last step is  $(n_f, r_f)$ .

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

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

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

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

- 190 1. *A negotiation is sound iff every run from the initial configuration can be*  
 191 *extended to a final run. The problem of soundness is to check if a given*  
 192 *negotiation is sound.*
- 193 2. *The problem of reachability asks if a given negotiation has a final run.*

194 Notice that the Brexit negotiation of Fig.1 is sound (but not deterministic).  
 195 It seems hard to preserve the important features of this negotiation while being  
 196 both sound *and* deterministic. The problem of soundness has received consider-  
 197 able attention. We summarize the results about soundness in the next theorem:

198 **Theorem 1.** *Determining whether a negotiation is sound is PSPACE-Complete.*  
 199 *For (very-)weakly non-deterministic negotiations, it is co-NP-complete [9]. For*  
 200 *acyclic negotiations, it is in DP and co-NP-Hard [6]. Determining whether an*  
 201 *acyclic weakly non-deterministic negotiation is sound is in PTIME [2, 9]. Fi-*  
 202 *nally, deciding soundness for deterministic negotiation is NLOGSPACE-complete [9].*

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

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

*Proof (sketch).* One can easily guess a run of size  $\leq |\mathcal{N}|$  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.  $\square$

## 207 ***k*-Layered Acyclic Negotiations**

208 We introduce a new class of negotiations which has good algorithmic properties,  
 209 namely *k*-layered acyclic negotiations, for *k* fixed. Roughly speaking, nodes of a  
 210 *k*-layered acyclic negotiations can be arranged in layers, and these layers contain  
 211 at most *k* nodes. Before giving a formal definition, we need to define the depth  
 212 of nodes in  $\mathcal{N}$ .

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

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

220 The Brexit example of Fig.1 is 6-layered. Notice that a layered negotiation  
 221 is necessarily *k*-layered for some  $k \leq |\mathcal{N}| - 2$ . Note also that we can always  
 222 transform an acyclic negotiation  $\mathcal{N}$  into a layered acyclic negotiation  $\mathcal{N}'$ , by  
 223 adding dummy nodes: for every node  $m \in \mathcal{X}(n, p, r)$  with  $\text{depth}(m) > \text{depth}(n) +$   
 224  $1$ , we can add several nodes  $n_1, \dots, n_\ell$  with  $\ell = \text{depth}(m) - (\text{depth}(n) + 1)$ , and  
 225 processes  $P_{n_i} = \{p\}$ . We compute a new relation  $\mathcal{X}'$  such that  $\mathcal{X}'(n, p, r) =$   
 226  $\{n_1\}$ ,  $\mathcal{X}'(n_\ell, p, r) = \{m\}$  and for every  $i \in 1..\ell - 1$ ,  $\mathcal{X}'(n_i, p, r) = n_{i+1}$ . This  
 227 transformation is polynomial: the resulting negotiation is of size up to  $|\mathcal{N}| \times$   
 228  $|\mathcal{X}| \times |P|$ . The proof of the following Theorem can be found in appendix C.

229 **Theorem 2.** *Let  $k \in \mathbb{N}^+$ . Checking reachability or soundness for a *k*-layered*  
 230 *acyclic negotiation  $\mathcal{N}$  can be done in PTIME.*

231 **3 Timed Negotiations**

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

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

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

250 *Example 2.* In the Brexit example, we define the following timed constraints  $\gamma$ .  
 251 We only specify the outcome names, as the timing only depends upon them.  
 252 Backstop and no-backstop both take between 1 and 2 weeks:  $\gamma(\text{backstop}) =$   
 253  $\gamma(\text{no-backstop}) = [1, 2]$ . In case of no-court, recess takes 5 weeks  $\gamma(\text{recess}) =$   
 254  $[5, 5]$ , and PM can meet EU immediatly  $\gamma(\text{meet}) = [0, 0]$ . In case of court ac-  
 255 tion, PM needs to spend 2 weeks in court  $\gamma(\text{c-meet}) = [2, 2]$ , and depending on  
 256 the court delay and decision, Pa needs between 3 (court overrules recess) to 5  
 257 (court confirms recess) weeks,  $\gamma(\text{defend}) = [3, 5]$ . Agreeing on a deal can take  
 258 anywhere from 2 weeks to 2 years (104 weeks):  $\gamma(\text{deal agreed}) = [2, 104]$  - some  
 259 would say infinite time is even possible! It needs more time with the backstop,  
 260  $\gamma(\text{deal w/backstop}) = [5, 104]$ . All others outcomes are assumed to be immedi-  
 261 ate, i.e., associated with  $[0, 0]$ .

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

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

271 A *timed run* is a sequence of steps  $\rho = (M_1, \mu_1) \xrightarrow{e_1} (M_2, \mu_2) \dots (M_k, \mu_k)$   
 272 where each  $(M_i, \mu_i) \xrightarrow{e_i} (M_{i+1}, \mu_{i+1})$  is a timed step. It is *final* if  $M_k = M_f$ . Its  
 273 *execution time*  $\delta(\rho)$  is defined as  $\delta(\rho) = \max_{p \in P} \mu_k(p)$ .



274 Notice that we only attached timing to processes, not to individual steps.  
 275 With our definition of runs, timing on steps may not be monotonous (i.e., non-  
 276 decreasing) along the run, while timing on processes is. Viewed by the lens of  
 277 concurrent systems, the timing is monotonous on the partial orders of the system  
 278 rather than the linearization. It is not hard to restrict paths, if necessary, to have  
 279 a monotonous timing on steps as well. In this paper, we are only interested in  
 280 execution time, which does not depend on the linearization considered.

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

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

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

- 287
- 288 – The mintime problem, i.e., do we have  $\text{mintime}(\mathcal{N}) \leq T$ ?
  - 289 In other words, does there exist a final timed run  $\rho$  with  $\delta(\rho) \leq T$ ?
  - 290 – The maxtime problem, i.e., do we have  $\text{maxtime}(\mathcal{N}) \leq T$ ?
  - 291 In other words, does  $\delta(\rho) \leq T$  for every final timed run  $\rho$ ?

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

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

<sup>4</sup> the authors of [10] acknowledged the issue with their algorithm for mintime.

## 315 4 High Level view of the main results

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

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

331 We first consider (Section 5) negotiations that are deterministic, but with-  
 332 out the soundness restriction. We show that for this class, no timed problem  
 333 we consider can be solved in PTIME (unless NP=PTIME). Further, we show  
 334 that the equality problems ( $maxtime/mintime(\mathcal{N}) = T$ ), are complete for the  
 335 complexity class DP, i.e., at the second level of the Boolean Hierarchy [16].

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

341 Finally, in order to obtain a polytime algorithm to compute the minimum  
 342 execution time, we consider the class of  $k$ -layered negotiations (see Section 7):  
 343 Given  $k \in \mathbb{N}$ , we can show that  $maxtime(\mathcal{N})$  can be computed in PTIME for  
 344  $k$ -layered negotiations. We also show that while the  $mintime(\mathcal{N}) \leq T?$  problem  
 345 is weakly NP-complete for  $k$ -layered negotiations, we can compute  $mintime(\mathcal{N})$   
 346 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) | NP-complete** (Thm. 7) |
|              |                         |                        | 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.

## 347 5 Deterministic Negotiations

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

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

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

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

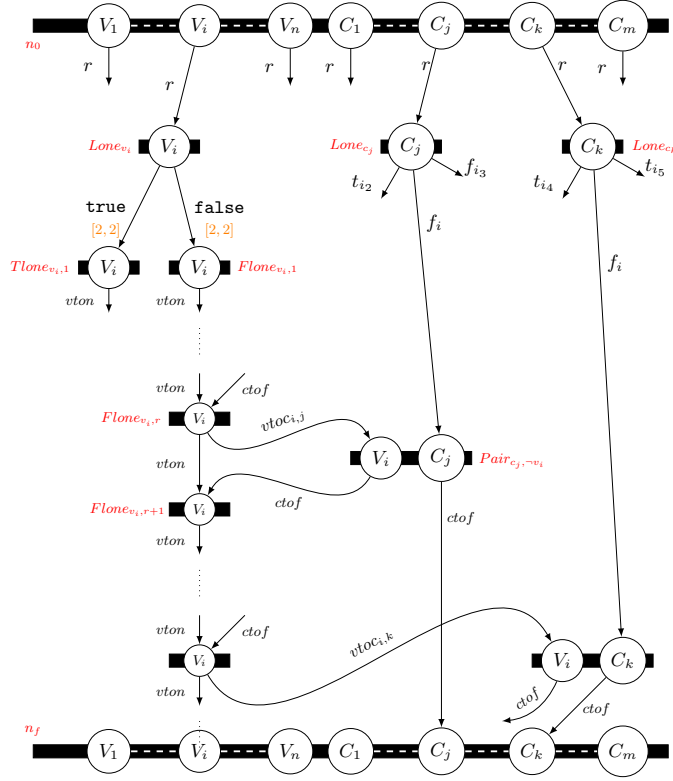
363 Step 1: Reducing 3-CNF-SAT to Reachability problem.

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

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

- 371 – Initial node  $n_0$  has a single outcome  $r$  taking each process  $C_j$  to node  $Lone_{c_j}$ ,  
 372 and each process  $V_i$  to node  $Lone_{v_i}$ .
- 373 –  $Lone_{c_j}$  has three outcomes: if literal  $v_i \in c_j$ , then  $t_i$  is an outcome, taking  
 374  $V_i$  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  
 375  $Pair_{c_j, \neg v_i}$ .
- 376 – The outcomes of  $Lone_{v_i}$  are **true** and **false**. Outcome **true** brings  $v_i$  to  
 377 node  $Tlone_{v_i, 1}$  and outcome **false** brings  $v_i$  to node  $Flone_{v_i, 1}$ .
- 378 – We have a node  $Tlone_{v_i, j}$  for each  $j \leq |S_{i,t}|$  and  $Flone_{v_i, j}$  for each  $j \leq |S_{i,f}|$ ,  
 379 with  $V_i$  as only process. Let  $c_r = S_{i,t}(j)$ . Node  $Tlone_{v_i, j}$  has two outcomes  
 380  $vton$  bringing  $V_i$  to  $Tlone_{v_i, j+1}$  (or  $n_f$  if  $j = |S_{i,t}|$ ), and  $vtoc_{i,r}$  bringing  $V_i$   
 381 to  $Pair_{c_r, v_i}$ . The two outcomes from  $Flone_{v_i, j}$  are similar.
- 382 – Node  $Pair_{c_r, v_i}$  has  $V_i$  and  $C_r$  as its processes and one outcome  $ctof$  which  
 383 takes process  $C_j$  to final node  $n_f$  and process  $V_i$  to  $Tlone_{v_i, j+1}$  (with  $c_r =$   
 384  $S_{i,t}(j)$ ), or to  $n_f$  if  $j = |S_{i,t}|$ . Node  $Pair_{c_r, \neg v_i}$  is defined in the same way  
 385 from  $Flone_{v_i, j}$ .

386 With this we claim that  $\mathcal{N}_\phi$  has a final run iff  $\phi$  is satisfiable which completes  
 387 the first step of the proof. We give a formal proof of this claim in Appendix A.  
 388 Observe that the negotiation  $\mathcal{N}_\phi$  constructed is deterministic and acyclic (but it  
 389 is not sound).



**Fig. 3.** A part of  $\mathcal{N}_\phi$  where clause  $c_j$  is  $(i_2 \vee \neg i \vee \neg i_3)$  and clause  $c_k$  is  $(i_4 \vee \neg i \vee i_5)$ . Timing is  $[0, 0]$  wherever not mentioned

390 Step 2: Before we introduce timing on  $\mathcal{N}_\phi$ , we introduce a new outcome  $r'$   
 391 at  $n_0$  which takes all processes to  $n_f$ . Now, the timing function  $\gamma$  associated  
 392 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  
 393 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  
 394 assignment: if  $\text{mintime}(\mathcal{N}_\phi) \leq 2$ , there is a run with decision  $r$  taken at  $n_0$   
 395 which is final. But existence of any such final run implies satisfiability of  $\phi$ . For  
 396 reverse implication, if  $\phi$  is satisfiable, then the corresponding run for satisfying  
 397 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.  $\square$

398 We now consider the related problem of checking if  $\text{mintime}(\mathcal{N}) = T$  (or if  
 399  $\text{maxtime}(\mathcal{N}) = T$ ). These problems are harder than their threshold variant un-  
 400 der usual complexity assumptions: they are DP-complete (Difference Polynomial

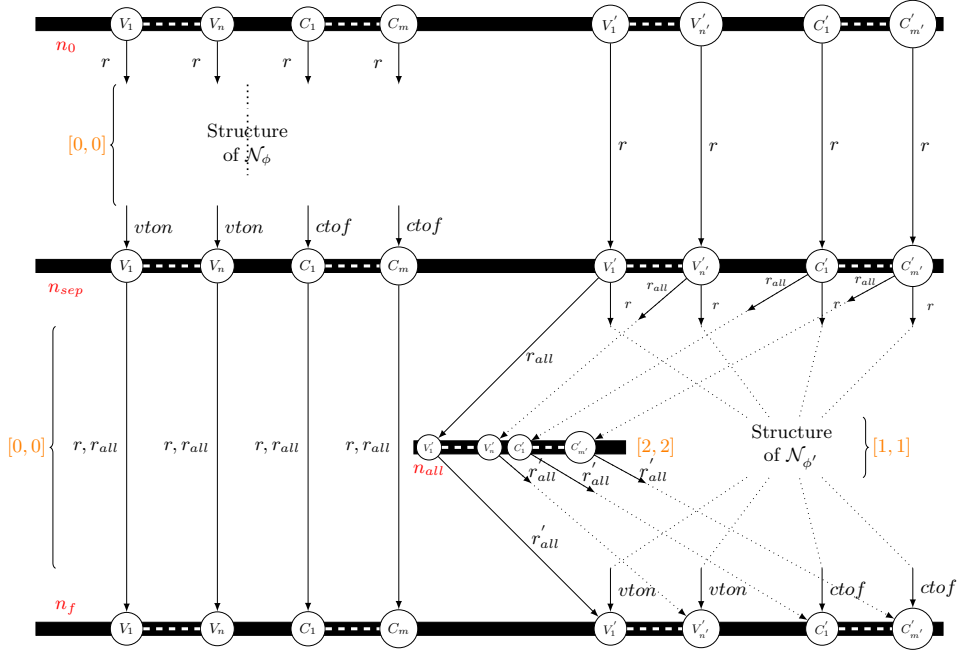


Fig. 4. Structure of  $\mathcal{N}_{\phi, \phi'}$

401 time class, i.e., second level of the Boolean Hierarchy, defined as intersection of  
 402 a problem in NP and one in co-NP [16]).

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

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

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

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

In the negotiation  $\mathcal{N}_{\phi, \phi'}$ , 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  $\mathcal{N}_\phi$  and  $\mathcal{N}_{\phi'}$  participating in them. This merged process gives us a new negotiation  $\mathcal{N}_{\phi, \phi'}$  in which the structure above  $n_{sep}$  is same as  $\mathcal{N}_\phi$  while below it is same as  $\mathcal{N}_{\phi'}$ . Node  $n_{sep}$  now has all the processes of  $\mathcal{N}_\phi$  and  $\mathcal{N}_{\phi'}$  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  $\mathcal{N}_\phi$  directly go to  $n_f$  of the  $\mathcal{N}_{\phi, \phi'}$ . Similarly  $n_0$  of  $\mathcal{N}_{\phi, \phi'}$  which is same  $n_0$  of  $\mathcal{N}_\phi$ , sends processes corresponding to  $\mathcal{N}_{\phi'}$  directly to  $n_{sep}$  for all its outcomes. We now define timing function  $\gamma$  for  $\mathcal{N}_{\phi, \phi'}$  which is as follows:  $\gamma(Lone'_{v_i}, r) = [1, 1]$  for all  $v_i \in \phi'$  and  $r \in \{\mathbf{true}, \mathbf{false}\}$ ,  $\gamma(n_{all}, r'_{all}) = [2, 2]$  and  $\gamma(n, r) = [0, 0]$  for all other outcomes of nodes. With this construction, one can conclude that  $mintime(\mathcal{N}_{\phi, \phi'}) = 2$  iff  $\phi$  is satisfiable and  $\phi'$  is unsatisfiable (see Appendix for details). This completes the reduction and hence proves DP-hardness.  $\square$

419 Finally, we consider a related problem of computing the min and max time.  
 420 To consider the decision variant, we rephrase this problem as checking whether  
 421 an arbitrary bit of the minimum execution time is 1. Perhaps surprisingly, we  
 422 obtain that this problem goes even beyond DP, the second level of the Boolean  
 423 Hierarchy and is in fact hard for  $\Delta_2^P$  (second level of the *polynomial* hierarchy),  
 424 which contains the entire Boolean Hierarchy. Formally,

425 **Theorem 4.** *Given an acyclic deterministic timed negotiation and a positive*  
 426 *integer  $k$ , computing the  $k^{\text{th}}$  bit of the maximum/minimum execution time is*  
 427  *$\Delta_2^P$ -complete.*

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

## 432 6 Sound Negotiations

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

439 Consider the graph  $G_{\mathcal{N}}$  of a negotiation  $\mathcal{N}$ . Let  $\pi = (n_0, (p_0, r_0), n_1) \cdots$   
 440  $(n_k, (p_k, r_k), n_{k+1})$  be a path of  $G_{\mathcal{N}}$ . We define the *maximal execution time* of  
 441 a path  $\pi$  as the value  $\delta^+(\pi) = \sum_{i \in 0..k} \gamma^+(n_i, r_i)$ . We say that a path  $\pi =$   
 442  $(n_0, (p_0, r_0), n_1) \cdots (n_\ell, (p_\ell, r_\ell), n_{\ell+1})$  is a path of some run  $\rho = (M_1, \mu_1) \xrightarrow{(n_1, r'_1)}$   
 443  $\cdots (M_k, \mu_k)$  if  $r_0, \dots, r_\ell$  is a subword of  $r'_1, \dots, r'_k$ .

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

446 *Proof.* Let us first prove that  $\text{maxtime}(\mathcal{N}) \geq \max_{\pi \in \text{Paths}(G_{\mathcal{N}})} \delta^+(\pi) + \gamma^+(n_f, r_f)$ .  
 447 Consider any path  $\pi$  of  $G_{\mathcal{N}}$ , ending in some node  $n$ . First, as  $\mathcal{N}$  is sound, we can  
 448 compute a run  $\rho_\pi$  such that  $\pi$  is a path of  $\rho_\pi$ , and  $\rho_\pi$  ends in a configuration  
 449 in which  $n$  is enabled. We associate with  $\rho_\pi$  the timed run  $\rho_\pi^+$  which asso-  
 450 ciates to every node the latest possible execution date. We have easily  $\delta(\rho_\pi^+) \geq$   
 451  $\delta(\pi)$ , and then we obtain  $\max_{\pi \in \text{Paths}(G_{\mathcal{N}})} \delta(\rho_\pi^+) \geq \max_{\pi \in \text{Paths}(G_{\mathcal{N}})} \delta(\pi)$ . As  
 452  $\text{maxtime}(\mathcal{N})$  is the maximal duration over all runs, it is hence necessarily greater  
 453 than  $\max_{\pi \in \text{Paths}(G_{\mathcal{N}})} \delta(\rho_\pi^+) + \gamma^+(n_f, r_f)$ .

454 We now prove that  $\text{maxtime}(\mathcal{N}) \leq \max_{\pi \in \text{Paths}(G_{\mathcal{N}})} \delta^+(\pi) + \gamma^+(n_f, r_f)$ . Take  
 455 any timed run  $\rho = (M_1, \mu_1) \xrightarrow{(n_1, r_1)} \dots (M_k, \mu_k)$  of  $\mathcal{N}$  with a unique maximal node  
 456  $n_k$ . We show that there exists a path  $\pi$  of  $\rho$  such that  $\delta(\rho) \leq \delta^+(\pi)$  by induction  
 457 on the length  $k$  of  $\rho$ . The initialization is trivial for  $k = 1$ . Let  $k \in \mathbb{N}$ . Because  $n_k$   
 458 is the unique maximal node of  $\rho$ , we have  $\delta(\rho) = \max_{p \in P_{n_k}} \mu_{k-1}(p) + \gamma^+(n_k, r_k)$ .  
 459 We choose one  $p_{k-1}$  maximizing  $\mu_{k-1}(p)$ . Let  $\ell < k$  be the maximal index of a  
 460 decision involving process  $p_{k-1}$  (i.e.  $p_{k-1} \in P_{n_\ell}$ ). Now, consider the timed run  
 461  $\rho'$  subword of  $\rho$ , but with  $n_\ell$  as unique maximal node (that is, it is  $\rho$  where  
 462 nodes  $n_i, i > \ell$  has been removed, but also where some nodes  $n_i, i < \ell$  have been  
 463 removed if they are not causally before  $n_\ell$  (in particular,  $P_{n_i} \cap P_{n_\ell} = \emptyset$ ).

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

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

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

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

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

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

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

481 *Proof (sketch).* First, we can decide  $\text{mintime}(\mathcal{N}) \leq T$  in NP. Indeed, one can  
 482 guess a final (untimed) run  $\rho$  of size  $\leq |N|$ , consider  $\rho^-$  the timed run corre-  
 483 sponding to  $\rho$  where all outcomes are taken at the earliest possible dates, and  
 484 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. 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.  $\square$

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

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

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

## 492 7 $k$ -Layered Negotiations

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

### 499 7.1 Algorithmic properties

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

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

511 *Proof (Sketch).* The first step is to compute  $S_i$  layer by layer, by following its  
 512 inductive definition. The set  $S_i$  is of size at most  $2^k$ , as  $|N_i| < k$  by definition  
 513 of  $k$ -layeredness. Knowing  $S_i$ , it is easy to build  $S_{i+1}$  by induction. This takes  
 514 time in  $O(|P||\mathcal{N}|^{k+1})$ : We need to consider all  $k$ -tuple of outcomes for each layer.  
 515 There can be  $|\mathcal{N}|^k$  such tuples. We need to do that for all processes ( $|P|$ ), and  
 516 for all layers (at most  $|\mathcal{N}|$ ).



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  $f_{i+1}(Y, n, X) = \max_{p \in P(n)} f_i(X, n(X, p)) + \gamma^+(n(X, p), r_p)$ . 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  $f(\{n_f\}, n_f)$ .  $\square$

517 We can bound the complexity precisely by  $O(d(\mathcal{N}) \cdot C(\mathcal{N}) \cdot \|R\|^{k^*})$ , with:  
518 –  $d(\mathcal{N}) \leq |\mathcal{N}|$  the depth of  $n_f$ , that is the number of layers of  $\mathcal{N}$ , and  $\|R\|$  is  
519 the maximum number of outcomes of a node,  
520 –  $C(\mathcal{N}) = \max_i |S_i| \leq 2^k$ , which we will call the *number of contexts of  $\mathcal{N}$* , and  
521 which is often much smaller than  $2^k$ .  
522 –  $k^* = \max_{X \in \cup_i S_i} |X| \leq k$ . We say that  $\mathcal{N}$  is  *$k^*$ -thread bounded*, meaning  
523 that there cannot be more that  $k^*$  nodes in the same context  $X$  of any layer.  
524 Usually,  $k^*$  is strictly smaller than  $k = \max_i |N_i|$ , as  $N_i = \bigcup_{X \in S_i} X$ .

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

## 530 7.2 Minimal Execution Time

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

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

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

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

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

546 Concerning timing constraints for outcomes we do the following: Outcomes  
547 0, yes and no are associated with  $[0, 0]$ . Outcome  $c_i$  is associated with  $[w_i, w_i]$ ,  
548 the weight of  $u_i$ . Last, outcome  $b_i$  is associated with a more complex function,

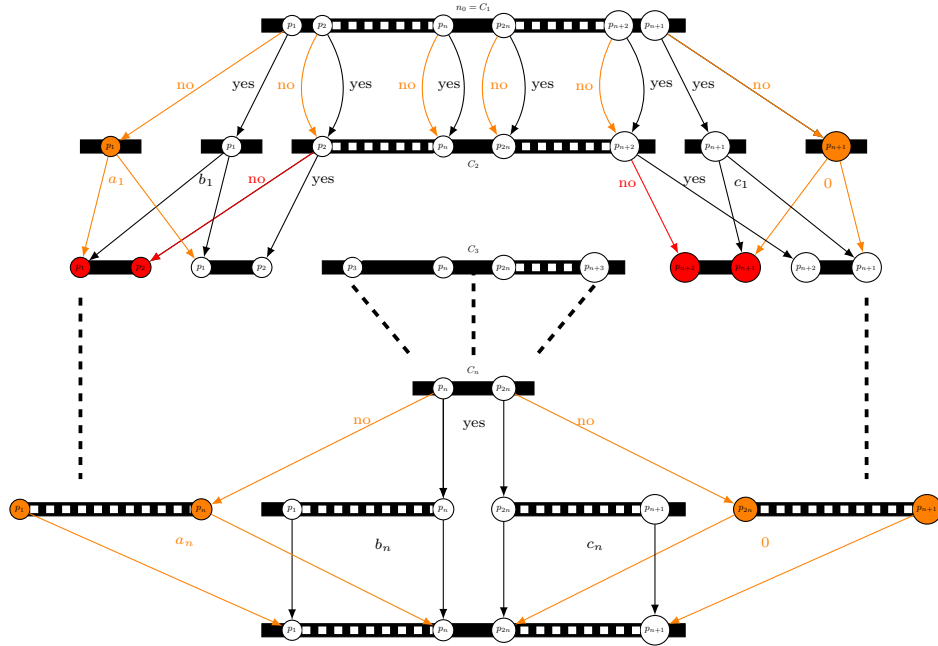


Fig. 5. The negotiation encoding Knapsack

549 such that  $\sum_i b_i \leq W$  iff  $\sum_i v_i \geq V$ . For that, we set  $[\frac{(v_{max}-v_i)W}{n \cdot v_{max}-V}, \frac{v_{max}W}{n \cdot v_{max}-v_i}]$  for  
550 outcome  $b_i$ , where  $v_{max}$  is the largest value of an item, and  $V$  is the total value  
551 we want to reach at least. Also, we set  $[\frac{(v_{max})W}{n \cdot v_{max}-V}, \frac{v_{max}W}{n \cdot v_{max}-v_i}]$  for outcome  $a_i$ . We  
552 set  $T = W$ , the maximal weight of the knapsack.

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

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

565 **Theorem 8.** *Let  $k \in \mathbb{N}$ . Given a  $k$ -layered negotiation  $\mathcal{N}$  and  $T$  written in*  
566 *unary, one can decide in PTIME whether the minimum execution time of  $\mathcal{N}$  is*  
567  *$\leq T$ . The worst-case time complexity is  $O(|\mathcal{N}| \cdot |P| \cdot (T \cdot |\mathcal{N}|)^k$ ).*

568 *Proof.* We will remember for each layer  $i$  a set  $\mathcal{T}_i$  of functions  $\tau$  from nodes  $N_i$   
569 of layer  $i$  to a value in  $\{1, \dots, T, \perp\}$ . Basically, we have  $\tau \in \mathcal{T}_i$  if there exists a  
570 path  $\rho$  reaching  $X = \{n \in N_i \mid f(n) \neq \perp\}$ , and this path reaches node  $n \in X$   
571 after  $\tau(n)$  time units. As for  $S_i$ , for all  $p$ , we should have a unique node  $n(\tau, p)$   
572 such that  $p \in n(f, p)$  and  $\tau(n(\tau, p)) \neq \perp$ . Again, it is easy to initialize  $\mathcal{T}_0 = \{\tau_0\}$ ,  
573 with  $\tau_0(n_0) = 0$ , and  $\tau_0(n) = \perp$  for all  $n \neq n_0$ .

574 Inductively, we build  $\mathcal{T}_{i+1}$  in the following way:  $\tau_{i+1} \in \mathcal{T}_{i+1}$  iff there exists a  
575  $\tau_i \in \mathcal{T}_i$  and  $r_p \in R_{n(\tau_i, p)}$  for all  $p \in P$  such that for all  $n$  with  $\tau_{i+1}(n) \neq \perp$ , we  
576 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  $\mathcal{N}$  is  $\min_{\tau \in \mathcal{T}_n} \tau(n_\tau)$ , for  $n$  the  
depth of  $n_f$ . There are at most  $T^k$  functions  $\tau$  in any  $\mathcal{T}_i$ , and there are at most  
 $|\mathcal{N}|$  layers to consider, giving the complexity.  $\square$

577 As with Thm. 6, we can more accurately state the complexity as  $O(d(\mathcal{N}) \cdot$   
578  $C(\mathcal{N}) \cdot \|R\|^{k^*} \cdot T^{k^* - 1})$ . The  $k^* - 1$  is because we only need to remember minimal  
579 functions  $\tau \in \mathcal{T}_i$ : if  $\tau'(n) \geq \tau(n)$  for all  $n$ , then we do not need to keep  $\tau'$  in  $\mathcal{T}_i$ .  
580 In particular, for the knapsack encoding in the proof of Thm. 7, we have  $k^* = 3$ ,  
581  $\|R\| = 2$  and  $C(\mathcal{N}) = 4$ .

582 Notice that if  $k$  is part of the input, then the problem is strongly NP-hard,  
583 even if  $T$  is given in unary, as e.g. encoding bin packing with  $\ell$  bins result to a  
584  $2\ell + 1$ -layered negotiations.

## 585 8 Conclusion

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

590 We showed that we can compute in PTIME the maximal execution time for  
591 acyclic negotiations that are either sound or  $k$ -layered, for  $k$  fixed. We showed  
592 that we cannot compute in PTIME the maximal execution time for negotiations  
593 that are not sound nor  $k$ -layered, even if they are deterministic and acyclic  
594 (unless NP=PTIME). We also showed that surprisingly, computing the minimal  
595 execution time is much harder, with strong NP-hardness results in most of the  
596 classes of negotiations, contradicting a claim in [10]. We came up with a new  
597 reasonable class of negotiations, namely  $k$ -layered negotiations, which enjoys  
598 a pseudo PTIME algorithm to compute the minimal execution time. That is,  
599 the algorithm is PTIME when the timing constants are given in unary. We  
600 showed that this restriction is necessary, as the problem becomes NP-hard for  
601 constants given in binary, even when the negotiation is sound and very weakly  
602 non-deterministic. The problem to know whether the minimal execution time  
603 can be computed in PTIME for deterministic and sound negotiation remains  
604 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*

- 654 on *Theory of Computing*, STOC '82, pages 255–260, New York, NY, USA, 1982.  
655 ACM.  
656 17. Robert H. Sloan and Ugo A. Buy. Reduction rules for time petri nets. *Acta Inf.*,  
657 33(7):687–706, 1996.

## 658 Appendix A: Deterministic Negotiations

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

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

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

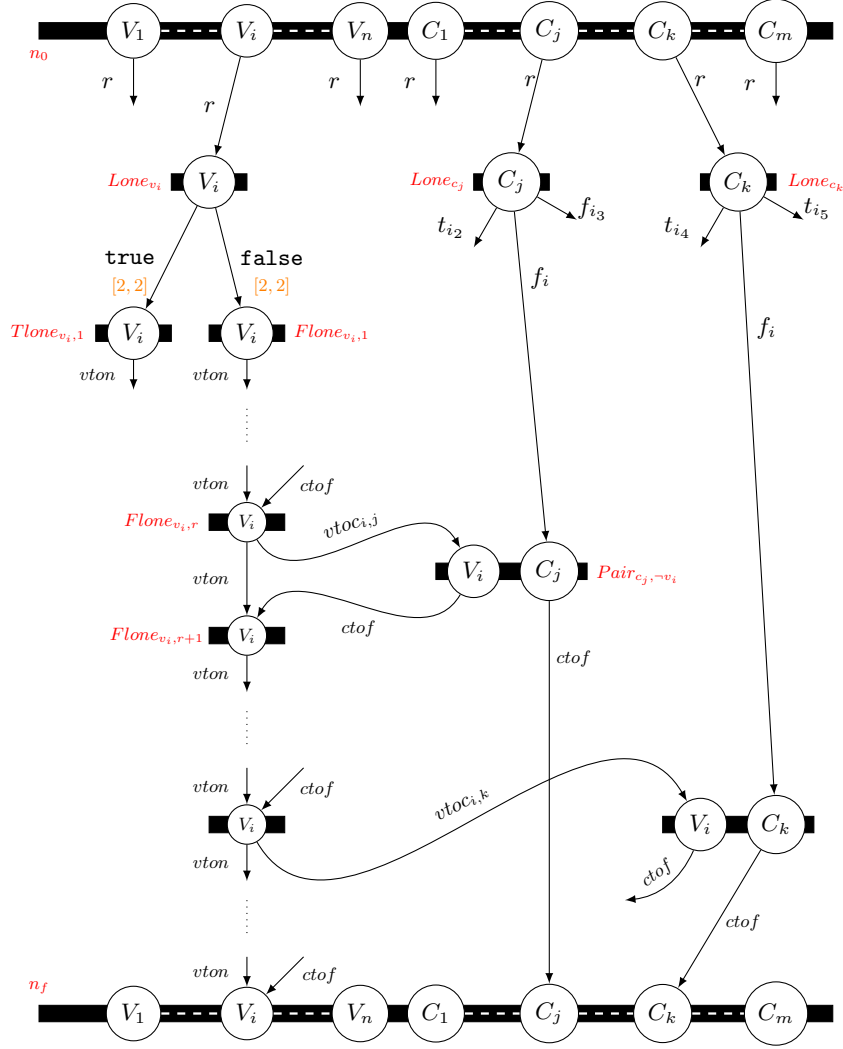
669 For the hardness, we give the proof in two steps. First, we start with a proof  
670 of Proposition 1 that reachability problem is NP-hard using reduction of 3-CNF  
671 SAT, i.e., given a formula  $\phi$ , we build a deterministic negotiation  $\mathcal{N}_\phi$  s.t.  $\phi$  is  
672 satisfiable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

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

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

680 Now, we construct a negotiation  $\mathcal{N}_\phi$  with a process  $V_i$  for each variable  $v_i$   
681 and a process  $C_j$  for each clause  $c_j$ :

- 682 – Initial node  $n_0$  has a single outcome  $r$  taking each process  $C_j$  to node  $\text{Lone}_{c_j}$ ,  
683 and each process  $V_i$  to node  $\text{Lone}_{v_i}$ .
- 684 –  $\text{Lone}_{c_j}$  has three outcomes: if literal  $v_i \in c_j$ , then  $t_i$  is an outcome, taking  
685  $V_i$  to  $\text{Pair}_{c_j, v_i}$ , and if literal  $\neg v_i \in c_j$ , then  $f_i$  is an outcome, taking  $V_i$  to  
686  $\text{Pair}_{c_j, \neg v_i}$ .
- 687 – The outcomes of  $\text{Lone}_{v_i}$  are **true** and **false**. Outcome **true** brings  $v_i$  to  
688 node  $\text{Tlone}_{v_i, 1}$  and outcome **false** brings  $v_i$  to node  $\text{Flone}_{v_i, 1}$ .
- 689 – We have a node  $\text{Tlone}_{v_i, j}$  for each  $j \leq |S_{i,\text{t}}|$  and  $\text{Flone}_{v_i, j}$  for each  $j \leq |S_{i,\text{f}}|$ ,  
690 with  $V_i$  as only process. Let  $c_r = S_{i,\text{t}}(j)$ . Node  $\text{Tlone}_{v_i, j}$  has two outcomes  
691  $\text{vton}$  bringing  $V_i$  to  $\text{Tlone}_{v_i, j+1}$  (or  $n_f$  if  $j = |S_{i,\text{t}}|$ ), and  $\text{vtoc}_{i,r}$  bringing  $V_i$   
692 to  $\text{Pair}_{c_r, v_i}$ . The two outcomes from  $\text{Flone}_{v_i, j}$  are similar.
- 693 – Node  $\text{Pair}_{c_r, v_i}$  has  $V_i$  and  $C_r$  as its processes and one outcome  $\text{ctof}$  which  
694 takes process  $C_j$  to final node  $n_f$  and process  $V_i$  to  $\text{Tlone}_{v_i, j+1}$  (with  $c_r =$   
695  $S_{i,\text{t}}(j)$ ), or to  $n_f$  if  $j = |S_{i,\text{t}}|$ . Node  $\text{Pair}_{c_r, \neg v_i}$  is defined in the same way  
696 from  $\text{Flone}_{v_i, j}$ .



**Fig. 3.** A part of  $\mathcal{N}_\phi$  where clause  $c_j$  is  $(i_2 \vee \neg i \vee \neg i_3)$  and clause  $c_k$  is  $(i_4 \vee \neg i \vee i_5)$ . Timing is  $[0, 0]$  wherever not mentioned

697 *Claim.*  $\mathcal{N}_\phi$  has a final run iff  $\phi$  is satisfiable.

698 *Proof.* First we show that if there is a run  $\rho$  from  $n_0$  to  $n_f$  then  $\phi$  is satisfiable.  
 699 In  $\rho$ , all processes reached  $n_f$ . So each process  $V_i$  takes either outcome **true** or  
 700 **false** in  $\rho$ . Let  $val$  the valuation associated each variable  $v_i$  with the choice  
 701 **true** or **false** by  $V_i$ . We now show that all clause  $c_r$  have at least one literal  
 702 true in  $val$ . In  $\rho$ , process  $C_r$  reaches the final node  $n_f$ : it must have gone via  
 703 one node either  $Pair_{c_r, v_i}$  or  $Pair_{c_r, \neg v_i}$ , for some  $i$ . Wlog, let us assume that  $C_r$   
 704 went to  $Pair_{c_r, \neg v_i}$ . The only way it is possible is for process  $V_i$  to have been in  
 705  $Flone_{v_i, j}$ , with  $c_r = S_{i, f}(j)$ . This is possible only if  $V_i$  decided outcome **false** at  
 706  $Lone_{v_i}$ . So this implies that literal  $\neg v_i$  of  $c_j$  is true in  $val$ . Hence  $\phi$  is satisfiable.

707 Conversely, we show that if  $\phi$  is satisfiable then  $\mathcal{N}_\phi$  has final run. Let  $val$  a  
 708 satisfiable assignment  $val : V \rightarrow \{\mathbf{true}, \mathbf{false}\}$  for  $\phi$ . We build a run  $\rho$  which is  
 709 final. After reaching  $Lone_{v_i}$ ,  $V_i$  will decide the outcome according to the value  
 710 of  $val(v_i)$  and reach  $Flone_{v_i, 1}$  or  $Tlone_{v_i, 1}$  accordingly. Let  $G_i(val)$  be the set  
 711 of clause  $c_j$  such that  $i$  is the minimal literal of  $c_j$  true under  $val$ . When there is  
 712 a choice between two outcomes  $vtoc$  and  $vtoc_{i, k}$  for process  $V_i$ , the run chooses  
 713  $vtoc_{i, k}$  iff  $k \in G_i(val)$ . Concerning  $C_j$ , it appears in exactly one  $G_i(val)$ , because  
 714  $val$  satisfies  $\phi$ . If  $val(v_i) = \mathbf{true}$ , run  $\rho$  chooses outcome  $t_i$  for  $V_i$  in node  $Lone_{c_j}$ ,  
 715 and outcome  $f_i$  if  $val(v_i) = \mathbf{false}$ . Observe that the same variable  $v_i$  can be  
 716 associated with several clauses  $c_j$ , but then all these clauses go to the same type  
 717 of nodes i.e.  $Pair_{c_j, v_i}$  if  $val(v_i) = \mathbf{true}$  and  $Pair_{c_j, \neg v_i}$  if  $val(v_i) = \mathbf{false}$ .

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

718 With this we claim that  $\mathcal{N}_\phi$  has a final run iff  $\phi$  is satisfiable which com-  
 719 pletes the first step of the proof. Observe that the negotiation  $\mathcal{N}_\phi$  constructed  
 720 is deterministic and acyclic (but it is not sound).

721 **Step 2:** Before we introduce timing on  $\mathcal{N}_\phi$ , we introduce a new outcome  $r'$   
 722 at  $n_0$  which takes all processes to  $n_f$ . Now, the timing function  $\gamma$  associated  
 723 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  
 724 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  
 725 assignment: if  $\text{mintime}(\mathcal{N}_\phi) \leq 2$ , there is a run with decision  $r$  taken at  $n_0$   
 726 which is final. But existence of any such final run implies satisfiability of  $\phi$ . For  
 727 reverse implication, if  $\phi$  is satisfiable, then the corresponding run for satisfying  
 728 assignment takes 2 units time, which means that  $\text{mintime}(\mathcal{N}_\phi) \leq 2$ .

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

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

732 We now consider the related problem of checking if  $\text{mintime}(\mathcal{N}) = T$  (or if  
 733  $\text{maxtime}(\mathcal{N}) = T$ ). These problems are harder than their threshold variant un-

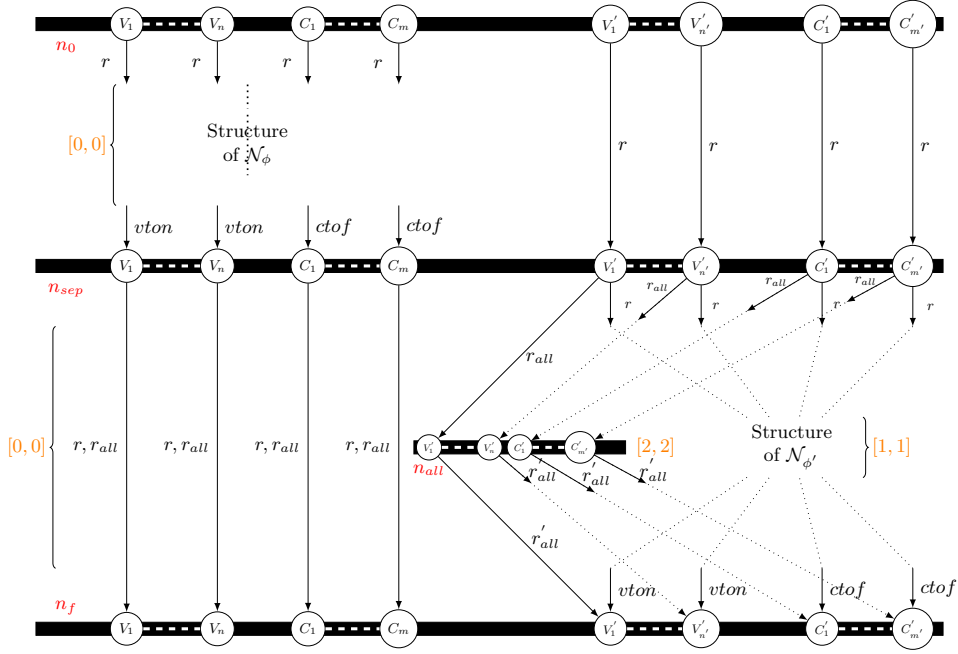


Fig. 4. Structure of  $\mathcal{N}_{\phi, \phi'}$

734 der usual complexity assumptions: they are DP-complete (Difference Polynomial  
 735 time class, i.e., second level of the Boolean Hierarchy, defined as intersection of  
 736 a problem in NP and co-NP [16]).

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

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

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

748 Given  $\phi, \phi'$ , we first make the corresponding negotiations  $\mathcal{N}_\phi$  and  $\mathcal{N}_{\phi'}$  as in  
 749 the previous proof. Let  $n_0$  and  $n_f$  be the initial and final nodes of  $\mathcal{N}_\phi$  and  $n'_0$  and  
 750  $n'_f$  be the initial and final nodes of  $\mathcal{N}_{\phi'}$ . (Similarly, for other nodes we write '  
 751 above the nodes to signify they belong to  $\mathcal{N}_{\phi'}$ ). In the negotiation  $\mathcal{N}_{\phi, \phi'}$ , we introduce  
 752 a new node  $n_{all}$  (see Figure 4), in which all the processes participate. The node  
 753  $n_{all}$  has a single outcome  $r'_{all}$  which sends all the processes to  $n_f$ . Also, for node



754  $n'_0$ , apart from the outcome  $r$  which sends all processes to different nodes, there  
 755 is another outcome  $r_{all}$  which sends all the processes to  $n_{all}$ .

756 Now we merge the nodes  $n_f$  and  $n'_0$  and call the merged node  $n_{sep}$ . Also nodes  
 757  $n_0$  and  $n'_f$  now have all the processes of  $\mathcal{N}_\phi$  and  $\mathcal{N}_{\phi'}$  participating in them.

758 This merged process gives us a new negotiation  $\mathcal{N}_{\phi,\phi'}$  in which the structure  
 759 above  $n_{sep}$  is same as  $\mathcal{N}_\phi$  while below it is same as  $\mathcal{N}_{\phi'}$ . Node  $n_{sep}$  now has all the  
 760 processes of  $\mathcal{N}_\phi$  and  $\mathcal{N}_{\phi'}$  participating in it. The outcomes of  $n_{sep}$  will be same  
 761 as that of  $n'_0$  ( $r_{all}, r$ ). For both the outcomes of  $n_{sep}$  the processes corresponding  
 762 to  $\mathcal{N}_\phi$  directly go to  $n_f$  of the  $\mathcal{N}_{\phi,\phi'}$ . Similarly  $n_0$  of  $\mathcal{N}_{\phi,\phi'}$  which is same  $n_0$  of  
 763  $\mathcal{N}_\phi$ , sends processes corresponding to  $\mathcal{N}_{\phi'}$  directly to  $n_{sep}$  for all its outcomes.  
 764 We now define timing function  $\gamma$  for  $\mathcal{N}_{\phi,\phi'}$  which is as follows:

- 765 -  $\gamma(Lone'_{v_i,r}) = [1, 1]$  for all  $v_i \in \phi'$  and  $r \in \{\mathbf{true}, \mathbf{false}\}$ ,
- 766 -  $\gamma(n_{all}, r'_{all}) = [2, 2]$  and
- 767 -  $\gamma(n, r) = [0, 0]$  for all other outcomes of nodes.

768 The claim is that

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

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

For the other side of the implication, we can argue similarly that if  $\phi$  is  
 satisfiable then the processes of  $\mathcal{N}_\phi$  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 with this run  
 will be 2 units which will also be the  $mintime(\mathcal{N}_{\phi,\phi'})$ .  $\square$

For equality decision problem of MaxTime, the proof is similar; only the  
 $\gamma(Lone'_{v_i,r}) = [2, 2]$  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  $maxtime(\mathcal{N}_{\phi,\phi'}) = 2$  which is true if only  
 if  $\phi$  is satisfiable and  $\phi'$  is unsatisfiable.  $\square$

777 Finally, we consider a related problem of deciding if a bit of  $mintime(\mathcal{N})$  is  
 778 1 (or similarly with  $maxtime(\mathcal{N})$ ). Perhaps surprisingly, we obtain that these  
 779 problems goes even beyond DP (the second level of the Boolean Hierarchy) and  
 780 is in fact hard for  $\Delta_2^P$ , which contains the whole Boolean Hierarchy:

781 **Theorem 4.** *Given an acyclic deterministic timed negotiation and a positive*  
 782 *integer  $k$ , computing the  $k^{th}$  bit of the maximum/minimum execution time is*  
 783  *$\Delta_2^P$  complete.*

784 *Proof.* Containment is again relatively easy. Given an acyclic deterministic timed  
785 negotiation, we can compute the largest possible time attainable as a function  
786 of the number of nodes and maximal constant in each node. Now guess the  
787 min/max time (in binary) and then check it using NP-oracle or equivalently  
788 Co-NP oracle calls.

789 The hardness is not so simple to obtain. We first notice that it suffices to  
790 show the problem of whether  $\text{maxtime}/\text{mintime}(\mathcal{N}) = \text{odd} ?$  is  $\Delta_2^P$  hard. This is  
791 because odd or even is the same as the last bit. We first show that  $\text{maxtime}(\mathcal{N})$   
792  $= \text{odd}$  is  $\Delta_2^P$  complete.

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

795 The above problem is known to be  $\Delta_2^P$  complete [14] and we reduce it to the  
796 decision problem of  $\text{maxtime}(\mathcal{N}) = \text{odd} ?$  First, we convert  $\phi$  to 3-CNF form us-  
797 ing Tseitin transformation. Let the new variables introduced be called  $t_1, t_2, \dots, t_k$ .  
798 So  $\phi(x_1, x_2, \dots, x_n)$  is equisatisfiable to 3-CNF  $\phi'(v_1, v_2, \dots, v_n, v_{n+1}, \dots, v_{n+k})$  where  
799  $v_i = x_i$  for  $i \leq n$  and  $v_i = t_i$  for  $i > n$ . We convert  $\phi'$  to a negotiation  $\mathcal{N}_{\phi'}$ .  $\mathcal{N}_{\phi'}$   
800 has the same structure as that of  $\mathcal{N}_{\phi}$  which was constructed in Theorem 3 apart  
801 from some change in arcs and participation of processes in nodes.

802 Participation changes are the following : The node  $Lone_{v_i}$  associated with each  
803 variable  $v_i$  of  $\phi'$  now involve two processes namely  $V_i$  and  $V_{i-1}$ . ( $Lone_{v_1}$  has only  
804  $V_1$  as process). Both of the outcomes, **true** and **false** associated with  $Lone_{v_i}$   
805 take  $V_{i-1}$  to  $n_f$  while **true** takes  $V_i$  to  $TLone_{v_i,1}$  and **false** takes  $V_i$  to  $Flone_{v_i,1}$ .  
806 Change in arcs is the following: The outcome  $v_{ton}$  of  $FLone_{v_i,r}$  where  $r = |S_{i,t}|$   
807 and  $TLone_{v_i,r'}$  where  $r' = |S_{i,t}|$  takes  $V_i$  to  $Lone_{v_{i+1}}$  (Except for  $i = n + k$  for  
808 which there is no change). We now define timing function  $\gamma$  as follows:

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

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

813

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

817

818 To prove the forward implication, consider any run  $\sigma$  which is final. Now,  
819 just like the proof in 3, the process  $V_i$  must have chosen either **true** or **false** at  
820 node  $Lone_{v_i}$ . The assignment  $f$ , corresponding to this outcome chosen by each  
821  $V_i$  is essentially the one whose lexicographic value is same as  $t$ . The fact that  
822 this assignment is satisfiable follows from the proof of theorem 3. To show that  
823 that lexicographic value is same, first of all the observe that time taken  $t$  can be  
824 written as  $2^{n-i_1} + 2^{n-i_2} + \dots + 2^{n-i_k}$  where  $V_{i_j}$  are those processes which chose  
825 **true** at  $Lone_{v_i}$ . Moreover  $i_j \leq n$ , which implies that all these variables are also  
826 present in  $\phi$ . Also the contribution of a variable  $x_{i_j}$  (which is same as  $v_{i_j}$ )  
827 in lexicographic value will be  $2^{n-i_j}$  which is same as its contribution in  $t$ . Hence  
828 the forward implication.

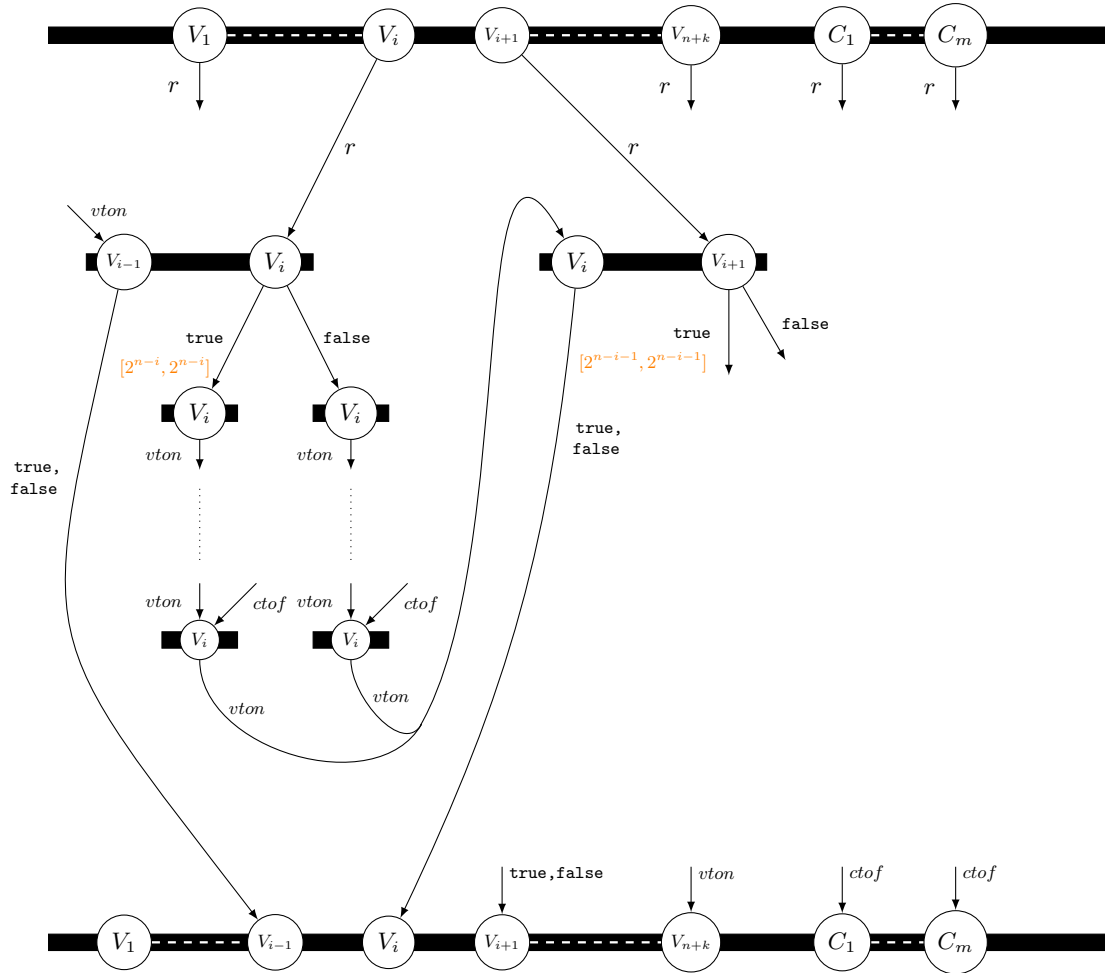


Fig. 5. A part of  $\mathcal{N}_\phi$ . Here if  $i > n$ , then timing with arcs **true** and **false** will be  $[0, 0]$ .

829 For backward implication, consider any satisfiable assignment  $f$  of  $\phi$ . Since  
 830  $\phi$  and  $\phi'$  are equisatisfiable hence there will exist a satisfiable assignment  $f'$  to  
 831  $\phi'$ , such that  $f'(x_i) = f(v_i)$  for  $i \leq n$ . Now following the proof of

832 Thm. 3, it is easy to see that the run  $\sigma$  corresponding to the assignment  $f'$  will  
 833 be final. Moreover the time taken for the path will be  $2^{n-i_1} + 2^{n-i_2} + \dots + 2^{n-i_k}$   
 834 where  $f'(v_{i_k}) = \mathbf{true}$ . Since all these  $i_j \leq n$ , these variables will also be present  
 835 in  $\phi$  and their contribution in lexicographic value of  $f$  would also be  $2^{n-i_j}$ . And  
 836 hence the backward implication.

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

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

## 840 Appendix B: Sound Negotiations

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

847 Consider the graph  $G_{\mathcal{N}}$  of a negotiation  $\mathcal{N}$ . Let  $\pi = (n_0, (p_0, r_0), n_1) \dots$   
 848  $(n_k, (p_k, r_k), n_{k+1})$  be a path of  $G_{\mathcal{N}}$ . We define the *maximal execution time* of  
 849 a path  $\pi$  as the value  $\delta^+(\pi) = \sum_{i \in 0..k} \gamma^+(n_i, r_i)$ . We say that a path  $\pi =$   
 850  $(n_0, (p_0, r_0), n_1) \dots (n_\ell, (p_\ell, r_\ell), n_{\ell+1})$  is a path of some run  $\rho = (M_1, \mu_1) \xrightarrow{(n_1, r'_1)}$   
 851  $\dots (M_k, \mu_k)$  if  $r_0, \dots, r_\ell$  is a subword of  $r'_1, \dots, r'_k$ .

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

854 *Proof.* Let us first prove that  $\text{maxtime}(\mathcal{N}) \geq \max_{\pi \in \text{Paths}(G_{\mathcal{N}})} \delta^+(\pi) + \gamma^+(n_f, r_f)$ .  
 855 Consider any path  $\pi$  of  $G_{\mathcal{N}}$ , ending in some node  $n$ . First, as  $\mathcal{N}$  is sound, we can  
 856 compute a run  $\rho_\pi$  such that  $\pi$  is a path of  $\rho_\pi$ , and  $\rho_\pi$  ends in a configuration  
 857 in which  $n$  is enabled. We associate with  $\rho_\pi$  the timed run  $\rho_\pi^+$  which asso-  
 858 ciates to every node the latest possible execution date. We have easily  $\delta(\rho_\pi^+) \geq$   
 859  $\delta(\pi)$ , and then we obtain  $\max_{\pi \in \text{Paths}(G_{\mathcal{N}})} \delta(\rho_\pi^+) \geq \max_{\pi \in \text{Paths}(G_{\mathcal{N}})} \delta(\pi)$ . As  
 860  $\text{maxtime}(\mathcal{N})$  is the maximal duration over all runs, it is hence necessarily greater  
 861 than  $\max_{\pi \in \text{Paths}(G_{\mathcal{N}})} \delta(\rho_\pi^+) + \gamma^+(n_f, r_f)$

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

863 Take any timed run  $\rho = (M_1, \mu_1) \xrightarrow{(n_1, r'_1)} \dots (M_k, \mu_k)$  of  $\mathcal{N}$  with a unique  
 864 maximal node  $n_k$ . We show that there exists a path  $\pi$  of  $\rho$  such that  $\delta(\rho) \leq \delta^+(\pi)$   
 865 by induction on the length  $k$  of  $\rho$ . The initialization is trivial for  $k = 1$ . Let  $k \in \mathbb{N}$ .  
 866 Because  $n_k$  is the unique maximal node of  $\rho$ , we have  $\delta(\rho) = \max_{p \in P_{n_k}} \mu_{k-1}(p) +$

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

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

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

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

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

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

889 A direct consequence is that  $\maxtime(\mathcal{N}) \leq T$  and  $\maxtime(\mathcal{N}) = T$  prob-  
890 lems can be solved in polynomial time when  $\mathcal{N}$  is. Notice that if  $\mathcal{N}$  is determi-  
891 nistic but not sound, then Lemma 1 does not hold: we only have an inequality.

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

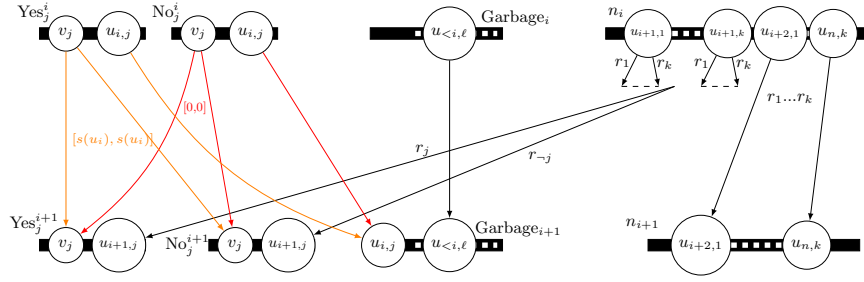
894 **Theorem 5.**  *$\mintime(\mathcal{N}) \leq T$  is NP-complete in the strong sense for sound*  
895 *acyclic negotiations, even if  $\mathcal{N}$  is very weakly non-deterministic.*

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

900 The hardness part is obtained by reduction from the **Bin Packing** problem.  
901 We give a set  $U$  of items, a size  $s(u) \in \mathbb{N}$  for each  $u \in U$ , a positive integer  
902  $B$  defining a bin capacity. The bin packing problem asks whether there exists  
903 a partition of  $U$  into  $k$  disjoint subsets  $U_1, U_2 \dots U_k$  such that the sum of sizes  
904 of items in each  $U_i$  is smaller or equal to  $B$ . Bin Packing is known to be NP-  
905 Complete [11] in the strong sense, that is even if the constants are given in  
906 unary. Let us now show that every instance of Bin Packing can be reduced to a  
907 min-time problem for very-weakly non-deterministic sound negotiations.

908 Given a set  $U$  of items, a bin capacity  $B$  and number  $k$  of bins, we build a  
 909 timed negotiation  $\mathcal{N}_{U,k}$  with  $k$  processes  $u_{i,1}, u_{i,2}, \dots, u_{i,k}$  for each item  $u_i \in U$ ,  
 910 and  $k$  additional processes  $v_1, v_2, \dots, v_k$ . The timing of a process  $v_i$  will encode the  
 911 total size of items put in the bin  $i$ . We then show that Bin Packing with items  
 912  $U$ ,  $k$  bins, and a bound  $B$  has a solution iff  $\text{mintime}(\mathcal{N}_{U,k}) \leq B$ .

913 We describe the negotiation  $\mathcal{N}_{U,k}$  layer by layer. In total we will have  $|U| + 1$   
 914 layers: intuitively, we will consider one item in each layer, and make one global  
 915 decision to decide in which bin this item goes. The first layer has only the initial  
 916 node  $n_0$ . The set of processes involved in  $n_0$  is the set of all processes. The  
 917 outcomes from the initial node are  $r_{1,1}, \dots, r_{1,k}$ , which tell in which bin  $1, \dots, k$   
 918 the first item is placed. Outcome  $r_{1,i}$  leads process  $u_{i,1}$  and  $v_i$  to node  $\text{YES}_i^1$ .  
 919 It leads processes  $u_{j,1}$  and  $v_j$  to  $\text{NO}_j^1$  for every  $j \neq i$ . Last, it leads all other  
 920 processes in  $\{u_{j,m} \mid j > 1, 1 \leq m \leq k\}$  to node  $n_1$ . Intuitively, moving to node  
 921  $\text{YES}_i^1$  means that item  $u_1$  is placed in bin  $i$ . The second layer has  $2k + 1$  nodes:  
 922  $\text{YES}_1^2 \dots \text{YES}_k^2, \text{NO}_1^2 \dots \text{NO}_k^2$  and  $n_1$ . The timing of outcome  $r_{1,i}$  from node  $n_0$   
 923 is  $\gamma(n_0, r_{1,i}) = [0, 0]$ .



**Fig. 6.** Layer  $i$  of the very weakly non-deterministic  $\mathcal{N}(U, k)$

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

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

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

936 In the same way,  $\text{NO}_j^i$  has a unique outcome  $r$ , timed with  $\gamma(\text{NO}_j^i, r) = [0, 0]$ ,  
 937 and with  $\mathcal{X}(\text{NO}_j^i, r) = \{\text{YES}_j^{i+1}, \text{NO}_j^{i+1}\}$ . It sends process  $u_{j,i}$  to  $\text{Garbage}_{i+1}$ .

938 The last layer has only node  $n_f$ . Nodes  $Yes_i^k$  and  $No_i^k$  both have a single  
 939 outcome which take all their processes to  $n_f$ .

940 The timing function  $\gamma$  is defined as follows:  $\gamma(Yes_j^i, r_i) = [s(u_i), s(u_i)]$  and  
 941  $\gamma(n, r) = [0, 0]$  for all other node and outcome  $r$ .

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

951 Last, we now show that  $\mathcal{N}_{U,k}$  is a very weakly non-deterministic, sound and  
 952 layered negotiations. First, the only processes that have non-deterministic transi-  
 953 tions are processes  $v_1, \dots, v_k$ , from  $Yes_j^i$  and  $No_j^i$  nodes. However, both nodes  
 954 also have the same deterministic process  $u_j^i$ . Thus  $\mathcal{N}_{U,k}$  is very weakly non-  
 955 deterministic. Let us now prove soundness. The only choices are made from  
 956 node  $n_i$ , the rest just follow in a unique way. From any configuration  $M$ , let  $i$   
 957 such that  $M(u_{i+1}, j) = \{n_i\}$  for some  $j$ . By construction,  $i$  is unique. We can  
 958 then do steps  $r_{i+1,1} \dots r_{n,1}$ , that is choosing to place items  $i+1, \dots, n$  to the first  
 959 bin. The steps from other processes are uniquely derived, and all processes reach  
 960  $n_f$ . The layeredness comes from the definition. Actually,  $\mathcal{N}_{U,k}$  is  $2k+2$ -layered,  
 961 for  $k$  the number of bins. However, as  $k$  is part of the input, it does not fall in  
 962 our  $k$ -layered restriction.  $\square$

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

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

966 *Proof.* The reduction is very similar to proof of Proposition 2. First, we define  
 967 the complement of Bin-Packing Problem, **Non-Bin-Packing Problem**:

968 Given a set  $U$  of items, a size  $s(u) \in \mathbb{N}$  for each  $u \in U$ , a positive integer bin  
 969 capacity  $B$ , does for any partition  $U$  into  $k$  disjoint subsets  $U_1, U_2 \dots U_k$  there  
 970 exist a subset  $U_i$  such that the sum of sizes of the items in  $U_i$  is more than  $B$ ?  
 971 Since the Bin-Packing Problem is NP-Complete, so the Non-Bin-Problem is co-  
 972 NP Complete. Now consider the following **Bin-Non-Bin Problem** :

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

976 Bin-Non-Bin Problem is DP-Complete, so we reduce it to our equality deci-  
 977 sion problem of min time. First, we construct the negotiations  $\mathcal{N}_{U_1, B_1, k_1}'$  and  
 978  $\mathcal{N}_{U_2, B_2, k_2}'$  like in proof of Theorem 5, but only after tripling each  $s(u)$  in  $U_1$  and  
 979 doubling each  $s(u)$  in  $U_2$ . Likewise we triple  $B_1$  and double  $B_2$ , so that new

980  $B'_1 = 3 * B_1$  and  $B'_2 = 2 * B_2$ .  
981 In  $\mathcal{N}_{U'_1, B'_1, k_1}$ , we add a new node  $n_0$  with a single outcome  $r$  which now acts  
982 as the first node. The older  $n_0$  is now called  $n'_0$ . We also add a new process  $a_1$ ,  
983 which goes to another new node  $n_{a_1}$  (has only  $a_1$  as process) from  $n_0$  for its single  
984 outcome  $r$ . Outcome  $r$  sends all other processes from  $n_0$  to  $n'_0$ . Node  $n_{a_1}$  has a  
985 single outcome  $r_1$  which takes  $a_1$  to  $n_f$ . Also,  $\gamma(n_{a_1}, r_1) = [3 * B_1 + 1, 3 * B_1 + 1]$   
986 while  $\gamma(n_0, r) = [0, 0]$ .  
987 Similarly in  $\mathcal{N}_{U'_2, B'_2, k_2}$ , we add a new node  $n_0$  with two outcomes  $r$  and  $r_{new}$   
988 which now acts as the first node. The older  $n_0$  is now called  $n'_0$ . We also add a  
989 new process  $a_2$ , which goes to another new node  $n_{a_2}$  (has only  $a_2$  as process) from  
990  $n_0$  for its outcome  $r$ . Outcome  $r$  sends all other processes from  $n_0$  to  $n'_0$ . Node  
991  $n_{a_2}$  has a single outcome  $r_2$  which takes  $a_2$  to  $n_f$ . Also,  $\gamma(n_{a_2}, r_2) = [2 * B_1, 2 * B_1]$   
992 while  $\gamma(n_0, r) = [0, 0]$ . For outcome  $r_{new}$  of  $n_0$ , all processes (including  $a_2$ ) di-  
993 rectly go to  $n_f$ . Also,  $\gamma(n_0, r_{new}) = [2 * B_2 + 1, 2 * B_2 + 1]$ .  
994 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  
995 as we merged in Corollary 2, merging the  $n_f$  of  $\mathcal{N}_{U'_1, B'_1, k_1}$  with  $n_0$  of  $\mathcal{N}_{U'_2, B'_2, k_2}$   
996 and making other similar changes we did in Corollary 2. We call this new ne-  
997 gotiation  $\mathcal{N}_{P'_1, P'_2}$ . Note the negotiation  $\mathcal{N}_{P'_1, P'_2}$  is sound as well as very weakly  
998 non-deterministic.

999 The claim is that  $\text{mintime}(\mathcal{N}_{P'_1, P'_2}) = 3 * B_1 + 2 * B_2 + 2$  iff  $(P_1, P_2)$  satisfy  
1000 Bin-Non-Bin Problem.

1001 We first show the reverse implication i.e if  $(P_1, P_2)$  satisfy Bin-Non-Bin Problem,  
1002 then  $\text{mintime}(\mathcal{N}_{P'_1, P'_2}) = 3 * B_1 + 2 * B_2 + 2$ . Since  $P_1$  is satisfiable, so the mintime  
1003 to complete the structure above  $n_{sep}$  of  $\mathcal{N}_{P'_1, P'_2}$  is  $3 * B_1 + 1$ . This is because all the  
1004 processes corresponding to  $\mathcal{N}_{U'_1, B'_1, k_1}$  take  $(\leq 3 * B)$  time to reach  $n_{sep}$  (because  
1005  $P_1$  is satisfies Bin-Packing) while  $a_1$  takes  $3 * B_1 + 1$  units of time. After reaching  
1006  $n_{sep}$ , processes can now take either outcome  $r_2$  or  $r_{new}$ . If processes choose  
1007 outcome  $r_2$ , then the timetaken by any final run will be  $(\geq 2 * (B_2 + 1))$  because  
1008  $P_2$  satisfies Non-Bin-Packing. On the other hand, if processes choose  $r_{new}$  to  
1009 reach  $n_f$ , then the time taken will be  $2 * B_2 + 1$ . So it is clear mintime for part  
1010 below  $n_{sep}$  is  $2 * B_2 + 1$ . So, overall the  $\text{mintime}(\mathcal{N}_{P'_1, P'_2}) = 3 * B_1 + 2 * B_2 + 2$ .

1011 For forward implication, we consider all four scenerios of  $(P_1, P_2)$  and argue that  
1012  $P_1$  satisfies Bin-Packing and  $P_2$  satisfies Non-Bin-Packing is the only possibility.  
1013 First let's assume that  $P_1$  does not satisfy Bin-Packing. Then the mintime to  
1014 complete the structure above  $n_{sep}$  is  $(\geq 3 * (B_1 + 1))$ . This is beacuse processes  
1015 corresponding to  $\mathcal{N}_{U'_1, B'_1, k_1}$  take at least  $3 * (B_1 + 1)$  time to reach  $n_{sep}$  while  
1016  $a_1$  take  $3 * B_1 + 1$ . Now since the mintime which can be taken to reach  $n_f$  from  
1017  $n_{sep}$  in either case whether  $P_2$  satisfies Non-Bin-Packing or not is  $(\geq 2 * B_2)$  so  
1018 the min time to complete  $(\mathcal{N}_{P'_1, P'_2}) \geq 3 * B_1 + 2 * B_2 + 3$ . Hence this shows that  
1019  $P_1$  satisfies Bin-Packing. This also shows the final run corresponding to mintime  
1020 of  $\mathcal{N}_{P'_1, P'_2}$  takes exactly  $3 * B_1 + 1$  units of time to reach  $n_{sep}$  from  $n_0$  (i.e. all  
1021 processes have reached  $n_{sep}$ ) if  $\text{mintime}(\mathcal{N}_{P'_1, P'_2}) = 3 * B_1 + 2 * B_2 + 2$ .

1022 Now if we assume the  $P_2$  does not satisfy Non-Bin-Packing, then the mintime to  
1023 reach  $n_f$  from  $n_{sep}$  is  $2 * B_2$ . And we already know that mintime to reach  $n_{sep}$



1024 from  $n_0$  is  $3 * B_1 + 1$ . So  $\text{mintime}(\mathcal{N}_{P'_1, P'_2}) = 2 * B_2 + 3 * B_1 + 1$ . Hence this leaves  
 1025 us with the only case when  $P_1$  satisfies Bin-Packing and  $P_2$  satisfies Non-Bin-  
 1026 Packing for which we already know that the min time taken is  $3 * B_1 + 2 * B_2 + 2$   
 1027 from the reverse implication.  $\square$

1028 An open question is whether the minimal execution time can be computed  
 1029 in PTIME if the negotiation is both sound and deterministic. The reduction to  
 1030 bin packing does not work with deterministic (and sound) negotiations.

## 1031 Appendix C: $k$ -Layered Negotiations

1032 In the previous sections, we have considered sound negotiations, and determinis-  
 1033 tic negotiations. For both classes, computing the minimal execution time cannot  
 1034 be done in PTIME (unless NP=PTIME), even if constants are given in unary.  
 1035 In this section, we consider  $k$ -layeredness (see Section 2), a syntactic property  
 1036 that can be efficiently verified (it suffices to compute the depth of each node,  
 1037 which can be done in polynomial time).

### 1038 8.1 Algorithmic properties

1039 Let  $k$  be a fixed integer. We first show that Reachability, Soundness and max-  
 1040 imum execution time can be checked in PTIME for  $k$ -layered negotiations (the  
 1041 two first results were stated in Section 2). Let  $N_i$  be the set of nodes at layer  
 1042  $i$ . We define for every layer  $i$  the set  $S_i$  of subsets of nodes  $X \subseteq N_i$  which can  
 1043 be jointly enabled and such that for every process  $p$ , there is exactly one node  
 1044  $n(X, p)$  in  $X$  with  $p \in n(X, p)$ . Formally, we define  $S_i$  inductively. We start with  
 1045  $S_0 = \{n_0\}$ . We then define  $S_{i+1}$  from the contents of layer  $S_i$ : we have  $Y \in S_{i+1}$   
 1046 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  
 1047  $m \in X$ , such that  $n \in \mathcal{X}(n(X, p), p, r_m)$  for each  $n \in Y$  and  $p \in P_n$ .

1048 **Theorem 6.** *Let  $k \in \mathbb{N}^+$ . Checking reachability, soundness and computing the*  
 1049 *maximum execution time for a  $k$ -layered acyclic negotiation  $\mathcal{N}$  can be done in*  
 1050 *PTIME. More precisely, the worst-case time complexity is  $O(|P| \cdot |\mathcal{N}|^{k+1})$ .*

1051 *Proof (Sketch of Proof).* The algorithm has the same form for all problems. The  
 1052 basis is to compute  $S_i$  layer by layer, by following its inductive definition. The  
 1053 set  $S_i$  is of size at most  $2^k$ , as  $|N_i| < k$  by definition of  $k$ -layeredness. Knowing  $S_i$ ,  
 1054 it is easy to build  $S_{i+1}$  by induction. This takes time at most  $O(|P| |\mathcal{N}|^{k+1})$ : We  
 1055 need to consider all  $k$ -uple of outcomes for each layer. There can be  $|\mathcal{N}|^k$  such  
 1056 tuples. We need to do that for all processes ( $|P|$ ), and for all layers (at most  
 1057  $|\mathcal{N}|$ ).

1058 For reachability, we just need to check whether layer  $S_d = \{n_f\}$ , where  $d$  is  
 1059 the depth of  $n_f$ .

1060 For soundness, let us denote by  $\text{Next}(X, (r_n)_{n \in X})$  the set of nodes that  
 1061 are successors of nodes in  $X$  after outcomes  $(r_n)_{n \in X}$ . We need to check that  
 1062 for all layer  $i$ , for all set  $X \in S_i$  and all tuple of outcomes  $(r_n)_{n \in X}$ , there

1063 is a  $Y \subseteq \text{Next}(X, (r_n)_{n \in X})$  such that every process  $p$  is in exactly one node  
 1064  $n(Y, p)$  of  $Y$ . All nodes of  $\text{Next}(X, (r_n)_{n \in X})$  are at depth  $i + 1$ , and thus there  
 1065 are at most  $k$  nodes in  $\text{Next}(X, (r_n)_{n \in X})$ . There are thus at most  $2^k$  subset  
 1066  $Y \subseteq \text{Next}(X, (r_n)_{n \in X})$  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  $f_0(\{n_0\}, n_0) = 0$ . The maximal execution time of  $\mathcal{N}$  is  $f(\{n_f\}, n_f)$ .  
 That is, for all nodes (at most  $|\mathcal{N}|$ ), we have to consider every  $k$ -uple of out-  
 comes, and there are at most  $|\mathcal{N}|^k$  of them, and every process to compute the  
 max, and the complexity is still in  $O(|P| \cdot |\mathcal{N}|^{k+1})$ .  $\square$

1067 We can bound the complexity precisely by  $O(d(\mathcal{N}) \cdot C(\mathcal{N}) \cdot \|R\|^{k^*})$ , with:  
 1068 –  $d(\mathcal{N}) \leq |\mathcal{N}|$  the depth of  $n_f$ , that is the number of layers of  $\mathcal{N}$ , and  $\|R\|$  is  
 1069 the maximum number of outcomes of a node,  
 1070 –  $C(\mathcal{N}) = \max_i |S_i| \leq 2^k$ , which we will call the *number of contexts of  $\mathcal{N}$* , and  
 1071 which is often much smaller than  $2^k$ .  
 1072 –  $k^* = \max_{X \in \cup_i S_i} |X| \leq k$ . We say that  $\mathcal{N}$  is  *$k^*$ -thread bounded*, meaning  
 1073 that there cannot be more than  $k^*$  nodes in the same context  $X$  of any layer.  
 1074 Usually,  $k^*$  is strictly smaller than  $k = \max_i |N_i|$ , as  $N_i = \bigcup_{X \in S_i} X$ .

1075 Consider again the Brexit example Figure 1. We have  $(k + 1) = 7$ , while  
 1076 we have the depth  $d(\mathcal{N}) = 6$ , the negotiation is  $k^* = 3$ -thread bounded ( $k^*$  is  
 1077 bounded by the number of processes), and the number of contexts is at most  
 1078  $C(\mathcal{N}) = 4$  (EU chooses to enforce backstop or not, and Pa chooses to go to court  
 1079 or not).

## 1080 8.2 Minimal Execution Time

1081 As with sound negotiations, computing minimal time is much harder than com-  
 1082 puting the maximal time for  $k$ -layered negotiations:

1083 **Theorem 7.** *Let  $k \geq 6$ . The  $\text{Min} \leq T$  problem is NP-Complete for  $k$ -layered*  
 1084 *acyclic negotiations, even if the negotiation is sound and very weakly non-deterministic.*

1085 *Proof.* One can guess in polynomial time a final run of size  $\leq |\mathcal{N}|$ . If the exe-  
 1086 cution time of this final run is smaller than  $T$  then we have found a final run  
 1087 witnessing  $\text{Min}(\mathcal{N}) \leq T$ . Hence the problem is in NP.

1088 Let us now show that the problem is NP-hard. We proceed by reduction from  
 1089 the knapsack decision problem. Let us consider a set of items  $U = \{u_1, \dots, u_n\}$   
 1090 of respective values  $v_1, \dots, v_n$  and weight  $w_1, \dots, w_n$  and a knapsack of maximal  
 1091 capacity  $W$ . The knapsack problem asks, given a value  $V$  whether there exists a  
 1092 subset of items  $U' \subseteq U$  such that  $\sum_{u_i \in U'} v_i \geq V$  and such that  $\sum_{u_i \in U'} w_i \leq W$ .

1093 We build a negotiation with  $2n$  processes  $P = \{p_1, \dots, p_{2n}\}$ . Intuitively,  $p_i, i \leq$   
1094  $n$  will serve to encode the value as timing, while  $p_i, i > n$  will serve to encode  
1095 the weight as timing. We set the set of nodes  $N = \{n_0, n_f\} \cup \{C_i \mid i \in 1..n\} \cup$   
1096  $\{n_{L,0,i}, n_{L,1,i}, n_{R,0,i}, n_{R,1,i} \mid i \in 1..n\}$ . Intuitively, node  $n_{L,1,i}$  (resp  $n_{R,1,i}$ ) will  
1097 be used to remember that item  $i$  is placed in the knapsack and that its value  
1098 (resp. weight) needs to be added. For all  $i$ , node  $n_{L,1,i}$  (resp.  $n_{R,1,i}$ ) has a unique  
1099 possible outcome,  $b_i$  (resp.  $c_i$ ). Nodes of the form  $n_{L,0,i}$  remember that item  $i$   
1100 has not been placed in the knapsack, and they have outcome  $a_i$ . Nodes of the  
1101 form  $n_{R,0,i}$  remember that item  $i$  has not been placed in the knapsack, and they  
1102 all have outcome 0. This outcome does not change the execution time, matching  
1103 the fact that the current weight and value of the knapsack is not increased.

1104 Last, nodes of the form  $C_i$  will just remember the items that have already  
1105 been considered. These nodes have two outputs, yes and no, telling whether the  
1106 item  $i$  should be placed in the knapsack or not, consistently for weight and value  
1107 processes.

1108 We set  $P_{n_0} = P_{n_f} = P$ , and for other nodes  $n_{L,0,i}$ ,  $P_{n_{L,0,i}} = P_{n_{L,1,i}} =$   
1109  $\{p_1 \dots p_i\}$  and  $P_{n_{R,0,i}} = P_{n_{R,1,i}} = \{p_{n+1} \dots p_{n+i}\}$ . Last  $P_{C_i} = \{p_{i+1} \dots p_n \cdot p_{n+i} \dots p_{2n}\}$ .

1110 We define the transition relation as follows:  $\mathcal{X}(n_0, \text{yes}, p_1) = \{n_{L,1,i}\}$ , and  
1111  $\mathcal{X}(n_0, \text{no}, p_1) = \{n_{L,0,1}\}$ , such that process  $p_1$  remembers that the item is picked/notpicked.  
1112 In the same way,  $\mathcal{X}(n_0, \text{no}, p_{n+1}) = \{n_{R,0,1}\}$  and  $\mathcal{X}(n_0, \text{yes}, p_{n+1}) = \{n_{R,1,1}\}$  for  
1113 process  $p_{i+1}$ . Hence both process  $p_1, p_{n+1}$  will have the same information about  
1114 whether the first item is picked or not. Finally, for every  $k \in 2..n$ , we define  
1115  $\mathcal{X}(n_0, \text{no}, p_k) = \mathcal{X}(n_0, \text{no}, p_{k+n}) = \mathcal{X}(n_0, \text{yes}, p_k) = \mathcal{X}(n_0, \text{no}, p_{k+n}) = \{C_1\}$ .

1116 Other layers are similar: for  $i \in 1..n$ , we have  $\mathcal{X}(C_i, \text{no}, p_i) = \{n_{L,0,i+1}\}$   
1117  $\mathcal{X}(C_i, \text{yes}, p_i) = \{n_{L,1,i+1}\}$ , Similarly, for every  $i \in 1..n$ ,  $\mathcal{X}(C_i, \text{no}, p_{i+n}) =$   
1118  $\{n_{R,0,i+1}\}$ , and  $\mathcal{X}(C_i, \text{yes}, p_{i+n}) = \{n_{R,1,i+1}\}$ . We set  $\mathcal{X}(C_i, \text{no}, p_j) = \mathcal{X}(C_i, \text{yes}, p_j) =$   
1119  $\{C_{i+1}\}$  for every  $j \in [i+1, n-1] \cup [n+i+1, 2n]$ .

1120 The most interesting set of transitions are to interface  $n_{L,0,i}, n_{L,1,i}, n_{R,0,i}, n_{R,1,i}$   
1121 with the next layer, in a non deterministic way because they dont know whether  
1122 the next item will be picked or not:  $\mathcal{X}(n_{L,0,i}, a_i, p_j) = \mathcal{X}(n_{L,1,i}, b_i, p_j) = \{n_{L,0,i+1}, n_{L,1,i+1}\}$   
1123 for  $j \in 1..i$  and,  $\mathcal{X}(n_{R,0,i}, 0, p_j) = \mathcal{X}(n_{R,1,i}, c_i, p_j) = \{n_{R,0,i+1}, n_{R,1,i+1}\}$  for  
1124  $j \in n+1..n+i$ .

1125 Last, all processes synchronize on  $n_f$  by setting  $\mathcal{X}(n_{L,0,n}, 0, p_j) = \mathcal{X}(n_{L,1,n}, b_n, p_j) =$   
1126  $\mathcal{X}(n_{R,0,n}, 0, p_j) = \mathcal{X}(n_{R,1,n}, c_n, p_j) = \{n_f\}$

1127 We now have to set timing constraints for outcomes. Outcomes 0, yes and  
1128 no are associated with  $[0, 0]$ . Outcome  $c_i$  is associated with  $[w_i, w_i]$ , the weight  
1129 of  $u_i$ . Last, outcome  $b_i$  is associated with a more complex function, such that  
1130  $\sum_i b_i \leq W$  iff  $\sum_i v_i \geq V$ . For that, we set  $[\frac{(v_{max}-v_i)W}{n \cdot v_{max}-V}, \frac{v_{max}W}{n \cdot v_{max}-v_i}]$  for outcome  
1131  $b_i$ , where  $v_{max}$  is the largest value of an item, and  $V$  is the total value we want to  
1132 reach at least. Also, we set  $[\frac{(v_{max})W}{n \cdot v_{max}-V}, \frac{v_{max}W}{n \cdot v_{max}-v_i}]$  for outcome  $a_i$ . We set  $T = W$ ,  
1133 the maximal weight of the knapsack.

1134 Now, consider a final run  $\rho$  in  $\mathcal{N}$ . The only choice is about *yes, no* from  
1135  $C_i$ . Let  $I$  be the set of indices such that *yes* is the outcome from all  $C_i$  in  
1136 this path. We obtain  $\delta(\rho) = \max(\sum_{i \notin I} a_i + \sum_{i \in I} b_i, \sum_{i \in I} c_i)$ . We have  $\delta(\rho) \leq$   
1137  $T = W$  iff  $\sum_{i \in I} w_i \leq W$ , that is the sum of the weight are lower than  $W$ , and

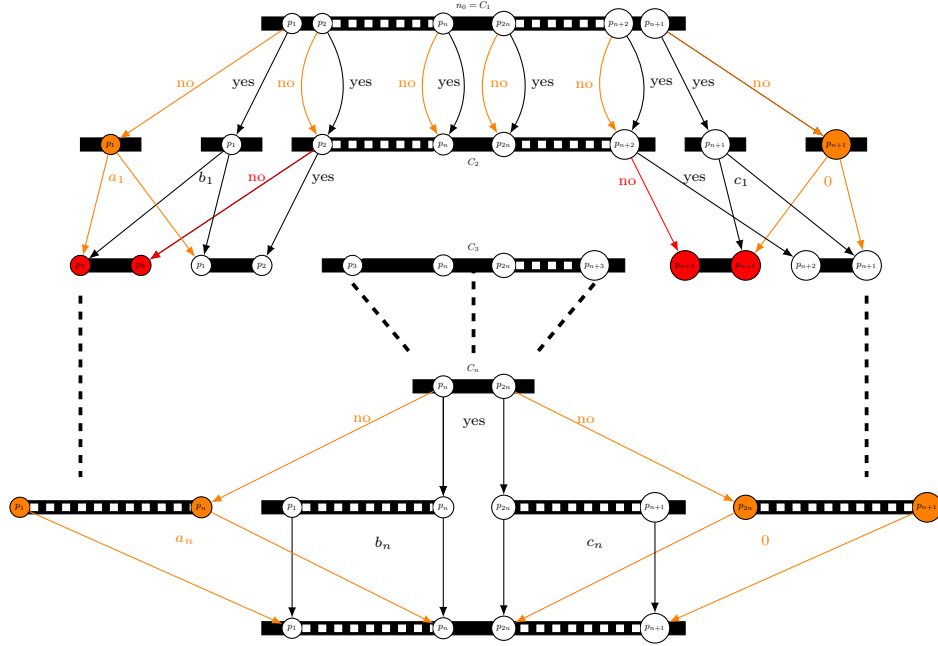


Fig. 5. The negotiation encoding Knapsack

1138  $\sum_{i \notin I} \frac{(v_{max})W}{n \cdot v_{max} - V} + \sum_{i \in I} \frac{(v_{max} - v_i)W}{n \cdot v_{max} - V} \leq W$ . That is,  $n \cdot v_{max} - \sum_{i \in I} v_i \leq n \cdot v_{max} - V$ ,  
 1139 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  
 1140 a set of items of weight less than  $W$  and of value more than  $V$ .

1141 So, given a knapsack of size  $n$ , a value  $V$  and a weight limit  $W$  one can build  
 1142 a negotiation  $\mathcal{N}_V^{Knapsack}$  with  $O(3n + 2)$  nodes. We can encode all weights with  
 1143  $\log(v_{max} \cdot W) + \log(n \cdot v_{max})$  bits. One can notice that  $\mathcal{N}_V^{Knapsack}$  is 5-layered and  
 1144 sound.

1145 However, it is not (weakly) non-deterministic because of nodes  $n_{L,0,i}, n_{L,1,i}, n_{R,0,i}, n_{R,1,i}$ .  
 1146 It is easy to add two processes  $V$  (resp.  $W$ ), present in all nodes  $n_{L,0,i}, n_{L,1,i}$   
 1147 (resp.  $n_{R,0,i}, n_{R,1,i}$ ), and make process  $P_i$  (resp.  $P_{n+i}$ ) leave these nodes, deter-  
 1148 ministically leading to a new node  $garbage_{i+1}$  at layer  $i + 1$ . Then the negotiation  
 1149 is very weakly deterministic, and 6-layered.  $\square$

1150 Following the same lines as for the proofs of Propositions 2 and 4, a conse-  
 1151 quence of Theorem 7 is that the  $Min = T$  problem is in DP for  $k$ -layered acyclic  
 1152 negotiations.

1153 It is well known that Knapsack is weakly NP-hard, that is it NP-hard only  
 1154 when weights/values are given in binary. This means that Thm. 7 shows that  
 1155 minimum execution time  $\leq T$  is NP-hard only when  $T$  is given in binary. We  
 1156 can actually show that for  $k$ -layered negotiations, the  $mintime(\mathcal{N}) \leq T$  problem  
 1157 can be decided in PTIME if  $T$  is given in unary (i.e. if  $T$  is not too large):

1158 **Theorem 8.** *Let  $k \in \mathbb{N}$ . Given a  $k$ -layered negotiation  $\mathcal{N}$  and  $T$  written in*  
 1159 *unary, one can decide in PTIME whether the minimum execution time of  $\mathcal{N}$  is*  
 1160  *$\leq T$ . The worst-case time complexity is  $O(|\mathcal{N}| \cdot |P| \cdot (T \cdot |\mathcal{N}|)^k$ ).*

1161 *Proof.* We will remember for each layer  $i$  a set  $\mathcal{T}_i$  of functions  $\tau$  from nodes  $N_i$   
 1162 of layer  $i$  to a value in  $\{1, \dots, T, \perp\}$ . Basically, we have  $\tau \in \mathcal{T}_i$  if there exists a  
 1163 path  $\rho$  reaching  $X = \{n \in N_i \mid f(n) \neq \perp\}$ , and this path reaches node  $n \in X$   
 1164 after  $\tau(n)$  time units. As for  $S_i$ , for all  $p$ , we should have a unique node  $n(\tau, p)$   
 1165 such that  $p \in n(f, p)$  and  $\tau(n(\tau, p)) \neq \perp$ . Again, it is easy to initialize  $\mathcal{T}_0 = \{\tau_0\}$ ,  
 1166 with  $\tau_0(n_0) = 0$ , and  $\tau_0(n) = \perp$  for all  $n \neq n_0$ .

1167 Inductively, we build  $\mathcal{T}_{i+1}$  in the following way:  $\tau_{i+1} \in \mathcal{T}_{i+1}$  iff there exists a  
 1168  $\tau_i \in \mathcal{T}_i$  and  $r_p \in R_{n(\tau_i, p)}$  for all  $p \in P$  such that for all  $n$  with  $\tau_{i+1}(n) \neq \perp$ , we  
 1169 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  $\mathcal{N}$  is  $\min_{\tau \in \mathcal{T}_n} \tau(n_\tau)$ , for  $n$  the  
 depth of  $n_f$ . There are at most  $T^k$  functions  $\tau$  in any  $\mathcal{T}_i$ , and there are at most  
 $|\mathcal{N}|$  layers to consider, giving the complexity.  $\square$

1170 As with Thm. 6, we can more accurately state the complexity as  $O(d(\mathcal{N}) \cdot$   
 1171  $C(\mathcal{N}) \cdot \|R\|^{k^*} \cdot T^{k^* - 1})$ . The  $k^* - 1$  is because we only need to remember minimal  
 1172 functions  $\tau \in \mathcal{T}_i$ : if  $\tau'(n) \geq \tau(n)$  for all  $n$ , then we do not need to keep  $\tau'$  in  $\mathcal{T}_i$ .  
 1173 In particular, for the knapsack encoding in the proof of Thm. 7, we have  $k^* = 3$ ,  
 1174  $\|R\| = 2$  and  $C(\mathcal{N}) = 4$ .

1175 Notice that if  $k$  is part of the input, then the problem is strongly NP-hard,  
 1176 even if  $T$  is given in unary, as e.g. encoding bin packing with  $k$  bins result to a  
 1177  $k + 1$ -layered negotiations.