Distributed and Parallel Databases, 11, 269–306, 2002

Temporal Reasoning in Workflow Systems

CLAUDIO BETTINI bettini@dsi.unimi.it
DSI, Università di Milano, Via Comelico 39, 20135 Milan, Italy

X. SEAN WANG xywang@gmu.edu
SUSHIL JAJODIA jajodia@gmu.edu
ISE Department, George Mason University, 4400 University Drive, Fairfax, VA 22030, USA

Recommended by: Ahmed Elmagarmid

Abstract. In a workflow system, autonomous agents perform various activities cooperatively to complete a common task. Successful completion of the task often depends on correct synchronization and scheduling of agents' activities. It would greatly enhance the capabilities of current workflow systems if quantitative temporal constraints on the duration of activities and their synchronization requirements can be specified and reasoned about. This paper investigates such requirements and related reasoning algorithms. In particular, the paper studies the consistency, prediction and enactment services in a workflow system, and provides corresponding algorithms. The consistency service is to ensure that the specification of the temporal constraints is possible to satisfy; the prediction service is to foretell the time frame for the involved activities; and the enactment service is to schedule the activities so that, as long as each agent starts and finishes its task within the specified time period, the overall constraints will always be satisfied. For the enactment service, the paper identifies two practically interesting families of enactment schedules for autonomous agents, namely "free schedules" and "restricted due-time schedules". In a free schedule, an agent may use any amount of time to finish the task as long as it is between the minimum and maximum time declared by the agent when the workflow is designed. A restricted due-time schedule is a more restrictive one in which the maximum amount of time that an agent may use is limited to a smaller number than the declared maximum. The paper presents efficient algorithms to find free and restricted due-time schedules. The paper also provides algorithms for the above services when multiple time granularities are involved in the temporal constraint specification.

Keywords: workflow, temporal reasoning, schedule, time granularity

1. Introduction

A workflow is a complete or partial automation of a business process, in which participants (humans or machines) involve in a set of activities according to certain procedural rules and constraints. The successful completion of the process often depends on the correct synchronization and scheduling of the activities. However, little attention has been given to modelling advanced temporal features for workflow systems. Commercial workflow systems (as reviewed, e.g., in [2]) are usually rather limited in their ability to specify temporal conditions for each individual activity or for the global plan and do not provide temporal reasoning.

We claim that the ability to specify and reason on quantitative temporal constraints about the duration and distance among activities would greatly enhance the capabilities of current

workflow systems. These constraints can regulate, e.g., durations of activities, not only imposing a deadline, but a time window within which the activity must be performed. Furthermore, they allow the designer to specify more precise synchronization constraints by imposing a specific range of values for the distance between the completion (or enactment) time of one activity and the completion (or enactment) time of another one. Reasoning about these constraints in a workflow system includes providing information to the participants regarding when and for how long they may engage in an activity due to the overall temporal constraints.[1] The modelling and reasoning tools provided by this paper are intended to address so called *production workflows* with particular emphasis on processes involving loosely coupled, largely distributed information sources where the agents cooperating in a workflow process are essentially independent from each other. A simple integration in such a workflow system of a known formalism to express the above constraints and to reason about them is not satisfactory. Indeed, the heterogeneous nature of activities, and the relative autonomy of workflow participants, give rise to new reasoning requirements. The purpose of this paper is to investigate these requirements and provide solutions.

A workflow management system provides a formalism and tools to specify workflow activities in terms of name, preconditions, actions, rules for exception handling, and possibly other features. In this paper, we concentrate on the temporal part of the specification. We propose to include in the workflow specification temporal constraints on individual activities, where a duration constraint or a specific deadline is given, as well as quantitative constraints on the distances between activities.

As a simple example, consider an online vendor workflow, including the following tasks that must be performed upon the receipt of an order by a customer: (a) order processing, (b) shipping, and (c) payment collection. These activities have certain conditions concerning their timing that may impose temporal distances (possibly involving different time granularities). For instance, the order processing must occur within one *business day* after the order is entered (and the whole workflow process is enacted), the order must be transmitted to the shipping sites within ten *hours* after the end of order processing, and the payment for the merchandise must be made within a time window starting and ending one month before and after delivery, respectively. The payment collection activity has a duration range (i.e., minimum and maximum time) specified in terms of *business days*, e.g., the activity can take as little as one business day and as much as 5 business days. (These requirements are included in the graphical representation of figure 2 later in the paper.)

The following questions can be important regarding the temporal constraints discussed above: (1) Are the temporal requirements among involved activities inconsistent (i.e., no matter what happens, it is impossible to satisfy all the constraints)? If the constraints are not possible to satisfy, then the workflow specification needs to be revised. If the constraints are consistent, (2) can we compute temporal bounds within which an agent will be asked to perform a task? This is to predict the possible time frame when an agent will need to carry out its task. This is important since with such information, the agent may be able to optimize its workload. Indeed, we assume the agents of a workflow system are autonomous and may be involved in different workflows. The ability to optimize its own workload is essential. Note that during the execution of the workflow, the temporal bounds may be refined and this computation may be repeated as needed.

The third question relates to the amount of time that an agent can use to perform an assigned task. Due to the desired relative autonomy of the agent, the agent should be given a time range such that no matter how much time the agent uses, as long as it is within the given time range, the overall temporal constraints are not violated. Generally, in the workflow system, an autonomous agent will declare a time range (the minimum and the maximum amount of time) it needs to finish a particular task. It is desirable to allow the agents to take any amount of time within the declared time range to finish the work. However, the time each agent actually takes to finish a task may have some impact on the overall temporal constraints because there may be constraints relating the ending times of activities. We then ask (3) whether there exists a schedule for the activities' enactment such that, no matter when each agent finishes the task within the declared time range (i.e., using any amount of time between the declared minimum and maximum), the overall temporal constraints are not violated. Such a schedule seems to be the most natural one when autonomous agents participate in the workflow. Sometimes the answer to this question is negative. For example, the workflow may require two activities to finish at exactly the same time while both agents declare that they need at least one time unit and at most two time units. In this case, the two agents cannot simply finish their respective tasks independently from each other within their declared time ranges, since this may lead to the two activities finishing at different times, violating the given constraint. Similar synchronization requirements are found, for example, in health care workflows: a transplantation surgery activity may require the concurrent presence of the organ to be implanted and blood for the patient. These products are the result of other activities and may be required to arrive within the same hour at the hospital, in order to avoid their functional degradation.

When the answer to (3) is negative, we then ask: (3′) once an agent starts an activity, is there a latest instant before which it has to finish (i.e., is there a due-time) in order to still satisfy the overall temporal constraints no matter what the other agents might choose to do within their own assigned (by the scheduler) time range? Note in this case, the amount of time allowed to the agent may be shorter than the maximum amount of time it requested. The answer to this question is valuable as it can serve for two purposes: The first is to provide the guideline for the agents to perform the tasks so that the overall constraints are guaranteed not to be violated. In the case when the agents can arrange their own workload, the agents may be able to arrange in a way so that they do not really use the declared maximum amount of time, and hence maintain the overall validity of the workflow. The second purpose is to alert the management of the workflow to check for a possible constraint violation and to invoke exception handling.

Another aspect that we consider in this paper is the presence of multiple time granularities. Due to the heterogeneity of participants in a workflow system, we assume that constraints involving different time granularities may co-exist as shown in our example above. Indeed, temporal constraint requirements on activity durations, and relative distances of activities may be specified by different agents and possibly using different time granularities. We study the above three questions and provide corresponding algorithms in the framework of multiple time granularities.

We organize the remainder of the paper as follows. In Section 2, we define temporal constraints in workflow systems and discuss in more details the above three questions involving

these constraints. In Section 3, we concentrate on question (3) and provide efficient algorithms. In Section 4, we extend our results to the cases of multiple granularities. We relate our work to other researches in Section 5, and conclude with Section 6. For presentation clarity, we relegate the proofs of all the claims made in the paper to an appendix.

2. Temporal reasoning in workflow systems

A typical workflow management system provides a formalism and tools to specify workflow activities. The techniques proposed for workflow temporal support in this paper are intended to be applicable with minor changes to a large class of formalisms for workflow process and activities description. However, for illustrative purposes we make some assumptions on the description language. In particular, we assume that the relation between activities can be specified by sequential routing as well as through the use of the operators OR-split, AND-split, OR-join, and AND-join, as presented in the consensus workflow glossary [19]. An OR-split identifies a point within the workflow where a single thread of control makes a decision upon which branch to take when encountered with multiple alternative workflow branches, while an AND-split is a point where a single thread of control splits into two or more parallel activities. OR-joins and AND-joins identify points where various branches reconverge to a single path. Note that an OR-join, as opposed to the AND-join, does not require synchronization, since the ancestors activities are alternatives and not executed in parallel.

In figure 1 we show an example of a workflow graphical description. In this example an arrow between two activities denotes sequential routing, while arrows going through a circle denote OR-split operators (C in the circle denotes the associated condition). AND-splits are implicit when more than one arrow originates from the same activity. All joins are implicit when more than one arrow lead to the same activity, where an OR-join is intended when the arrows originate from alternative activities, and an AND-join when from parallel activities. The dashed arrows mean that details on other activities in that part of the workflow are omitted.

We propose to enhance a workflow management system with a temporal support module offering two essential features: (a) a formalism to be used during the design of the workflow to specify quantitative temporal constraints both in the activity and process descriptions, and (b) a reasoning tool on these constraints for the benefit of the workflow management and of the workflow enactment service.

Temporal constraints can be part of an activity's description as a duration constraint usually provided by the agent in charge of that activity, or as a deadline constraint relative to absolute time, or to some other activity. We also provide a way to impose constraints on the instants at which the activity should start and/or end. For example, we could impose that an activity of issuing pay-checks can only be started on the last Friday of each month. In addition, temporal constraints can be part of the workflow process description in terms of quantitative constraints on the distance among activities.

In our temporal support module, the set of temporal constraints associated with a workflow process description can be represented as a directed graph whose nodes represent variables and an edge (or arc) from X_i to X_j labeled with an interval $[m, n]$ indicates that a constraint is specified on the distance between the values of these variables. Namely, that distance

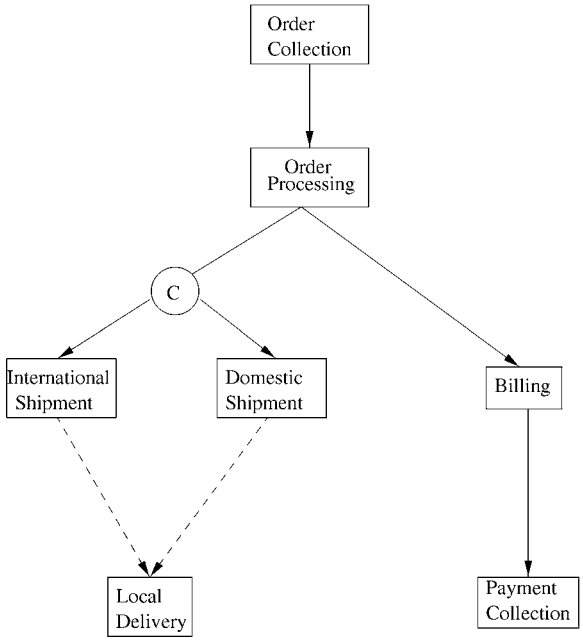

Figure 1. A workflow process description.

must not be less than m nor more than n. The constants m and n are either integers or one of the special symbols $-\infty$ and ∞, respectively. When the interval is followed by a time granularity, it indicates that the bounds are given in terms of that granularity.[2] This special type of constraint is called a Temporal Constraint with Granularity, or TCG for short.

In figure 2 we show the graph representing the temporal constraints imposed on the activities of the workflow process in figure 1. Each node in the graph is labeled by the initials of the activity's name followed by 'b' for begin or 'e' for end. Specifically, `OC` stands for the {O}rder {C}ollection activity, `OP` for {O}rder {P}rocessing, `B` for {B}illing, `PC` for {P}ayment {C}ollection, `IS` and `DS` for {I}nternational and {D}omestic {S}hipping, and `LD` for {L}ocal {D}elivery.

The temporal constraints in figure 2 restricts the durations of activities and distances between activities. For example, the order processing activity is forced to precede the billing activity by imposing $[1, +\infty]$ to the arc from *OPe* to *Bb*. Since this is a common constraint, we use the simpler symbol '$<$' instead. As an example of a more complex constraint, we can specify that the temporal distance between the completion of Payment Collection and of Local Delivery should not exceed one month by assigning $[-1, 1]$`month` to the arc from *LDe* to *PCe*. Note here *LDe* and *PCe* are not forced to be in any particular order: Either can happen before the other as long as they are not more than one month apart. The distance is 1 month from Local Delivery to Payment Collection if Payment Collection completes in the next month with respect to Local Delivery, 0 months if they occur within the same month, and -1 month if Payment Collection completes in the previous month. More than one TCG can

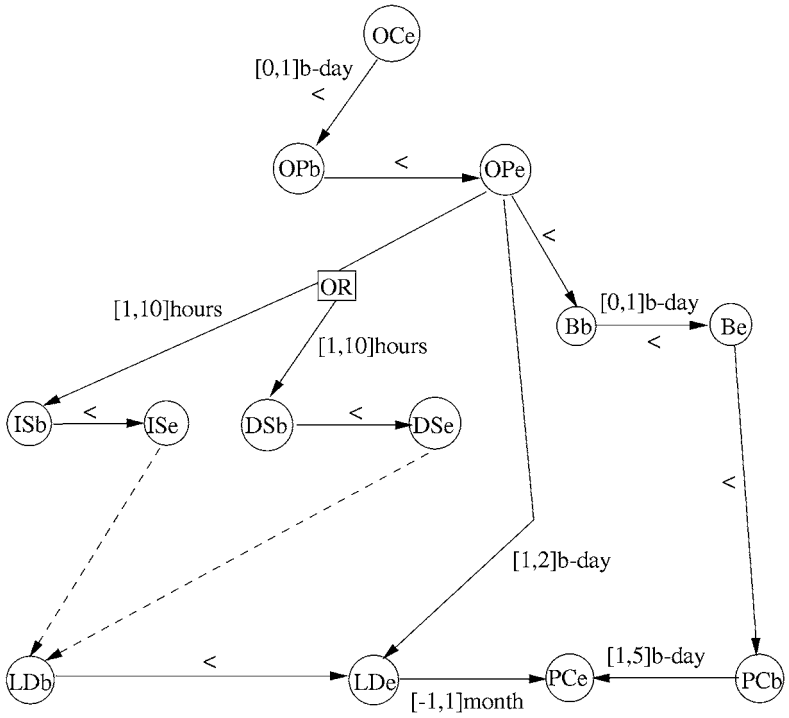

Figure 2. Temporal constraints in a workflow.

be associated with an arc, with conjunction of the TCGs being the semantics. For example, the constraints from the beginning of Billing (*Bb*) to the the end of Billing (*Be*) consists of two TCGs with different granularities, where the first is [0, 1] `b-day` imposing that the end of the Billing activity occurs in the same or next business-day as the beginning (i.e., 0–1 business days), and the second is '$<$' (i.e., $[1, \infty]$) restricting that the duration must be positive, i.e., *Be* must happen at least one time unit after *Bb*. Note that this second constraint is necessary since the constraint [0, 1] `b-day` allows that two variables are mapped to instants within the same business-day without imposing which instants should be the first.

The reasoning tool in our temporal support module offers the following services within the workflow management system: (i) Check the consistency of complex temporal requirements among several activities, (ii) monitor workflow activities and estimate their starting and ending time, (iii) provide useful information to the enactment service for activity scheduling. We now elaborate on each service.

2.1. Consistency

Consistency checking is clearly an important task. Enacting a workflow process that has an inconsistent specification will only lead to a waste of resources. The goal here is to discover inconsistency from the specification.

In order to efficiently process the set of constraints, a workflow constraint graph as in figure 2 needs to be decomposed according to the OR operators in the graph, so that each resulting constraints subgraph represents one possible execution thread of the workflow. In the following we call these subgraphs *constraint networks*. Formally, a constraint network is a quadruple (W, A, Γ, Dom), where W is a set of nodes, and A a set of directed arc between nodes of W, Γ assigns a set of TCGs to each arc, and *Dom* assigns an interval on the positive integers to each node.[3] When the granularities associated with all the TCGs in a constraint network are the same, the network has a well-known characterization in the constraint satisfaction literature as STP (Simple Temporal Problem) [10]. A constraint network is *consistent* if it has a solution, i.e., there is an assignment of values to the nodes of the network from the associated domains given by *Dom* such that all the constraints in Γ are satisfied. In terms of workflow, a constraint network is consistent means there is a possible execution for the corresponding thread of the workflow.

A workflow constraint graph is said to be *inconsistency-free* if all the constraint networks (one for each execution thread) are consistent. That is, the workflow may actually be carried through (from the temporal constraint perspective) regardless which execution thread is chosen. If the constraint network corresponding to a particular execution thread is not consistent, then the execution thread is impossible to carry through without raising exceptions, and workflow constraint graph should be modified, either by eliminating that execution thread or by relaxing the temporal constraints.

As an example, consider the workflow constraint graph in figure 2. This graph is decomposed into two constraint networks since there is only one OR-split operator. The first constraint network takes the left branch (i.e., performing *ISb* and *ISe*) and the second the right branch (i.e., performing *DSb* and *DSe*). The other activities and related temporal constraints will appear in both constraint networks. This simple workflow constraint graph is inconsistency-free since both networks are consistent.

Consistency checking for STPs can be done efficiently[4] by a variety of algorithms. One example is the Floyd-Warshall *all-pairs shortest path* algorithm see [10, 17] applied to a distance graph[5] that is equivalent to the constraint network. In the sequel, such an algorithm will be called generically the *consistency algorithm*. We will discuss in Section 4 the extension to constraint networks with multiple granularities.

2.2. *Prediction*

A basic service that the temporal support can offer during the workflow execution is to estimate the earliest starting time and the latest ending time (i.e., due-time) of activities. For a specific activity that is not enacted, its earliest starting time and latest ending time may be changing with the execution status of the workflow. The earliest starting time may be moved forward and the latest ending time may be moved backward. For example, assume there are three sequential activities between the end of international shipping and the beginning of the local delivery. These activities are Duty Assessment (DA), Duty Balance-checking (DB) and Duty Collection (DC). Assume further that each activity needs at least 2 hours and at most 4 hours to finish, and the time lapse allowed between *DA* and *DB* is 1 to 2 hours, and *DC* has to start immediately after *DB* finishes. When the workflow process is enacted

at 1 am in the morning (i.e., *DA* starts at 1 am), we know that the earliest possible starting time for *DC* is at 6 am. Note that this 6 am is only an estimate since this starting time may be pushed forward. Indeed, when *DA* is finished at 4 am, we can push further the earliest time for *DC* to 7 am. Although they are only estimates, such earliest possible starting and latest possible ending times are useful for dynamically allocating resources. Indeed, if we know that *DC* won't be enacted before 6 am, we may allocate the resources needed for *DC* to other activities before 6 am. Also, if an activity involves a human agent, knowing such information will help (so she does not have to get up before 6 am!).

At any moment during the execution of the workflow, the temporal support module can easily derive estimates for the earliest possible starting time and the latest possible due-time for each of the yet-to-be-enacted activities. The derivation is based on the consistency algorithm for networks with a single granularity (multiple granularity case is considered in Section 4). The consistency algorithm, in addition to checking consistency, also refines the given domains and constraints, returning a *minimal network*. A minimal network has the property that each value in each of the resulting variable domain participates in at least one solution. Note that the domains of the resulting network are still intervals. Since each solution corresponds to one possible execution of the workflow, the minimal value of the resulting domain for each variable is the earliest starting time for that activity in that thread. It is now obvious that the earliest possible starting time for all possible threads can be derived easily. A similar procedure applies to obtain the latest possible due-time if it exists.

In order to take into account the already started or finished activities, before running the consistency algorithm, we need to instantiate the variables of the corresponding nodes with the time values of the real starting and finishing times observed from the execution itself, i.e., set $Dom(w)$ to consist of a single value which is the time when w happened. With the progress of the workflow the estimates will increase their accuracy, since the number of networks to be considered will decrease (as OR-split choices will decrease) and the number of variables to be instantiated with single values will increase.

Depending on the workflow application domain, the above service can be integrated in the workflow management system in two ways: (a) the system automatically generates *early warnings* directed to the agents responsible of each activity; (b) the service generates timing estimates *on demand*, allowing the participants to submit queries about when specific activities are likely to occur. For example, the workflow administrator could ask the system when a package is supposed to be handled by local delivery, based on the current status of the workflow.

The above service can also be easily extended to support a form of hypothetical reasoning, estimating the schedule of some activities based on assumptions on the schedule of other activities. For example, even if the workflow has not been enacted yet, the workflow administrator could ask if, assume a package reaches the international shipment by Thursday night, what is the latest time for its final delivery? Technically, we can provide this service by making a copy of the current set of constraint networks and instantiating the variables according to the assumptions. Then, similarly to what explained above, the consistency algorithm is run on each network and the expected values are derived from the resulting variable domains.

2.3. Support for the enactment service

Perhaps the most important service provided by a temporal support module is the generation of a temporal schedule to be used by an activity enactment agent in the workflow system. Every time the enactment agent dispatches an activity, it needs to provide a set of constraints on the beginning and ending times to the agent in charge of that activity, such that if each agent satisfies the constraints, the whole workflow is still possible to be successfully carried out. Enactment schedules will be considered in details in the next section. Here, we only consider two examples.

Consider the graph in figure 2 and the corresponding workflow. Assume that *OPe* occurs (i.e., the order processing is completed) on June 1, 2001. By the constraints of *OPe* to *LDe* and *LDe* to *PCe*, *PCe* (i.e., completion of payment collection) has to occur before the end of July. Since *PC* may take 5 business-days to finish, we have to finish Billing (i.e., *Be*) on or before July 24 (5 business-days before the end of July). Hence, a due-date for *Be* must be set to July 24, 2001. Obviously, this kind of due-date information is important for the workflow management system and for the workflow participants.

Earliest starting times assigned to activities should also be considered carefully, as shown in the next example which is a refinement of a portion of the online vendor workflow process, from the end of order processing (*OPe*) to beginning of local delivery (*LDb*).

Example 1. Upon completion of the order processing, the online vendor asks directly the suppliers of the requested products to ship them to one of its warehouses located in the area of the customer for their final delivery. Obviously, the workflow requires some sort of synchronization to ensure that all the products will be delivered to the customer within a certain time to reduce the needs of warehouse space. In figure 3 we consider the example of two activities A and B corresponding to the shipments to be made by two suppliers. Each supplier has provided minimum and maximum bounds for the handling and shipping of its products (constraints [3, 7] and [1, 3] respectively). The two activities must start after

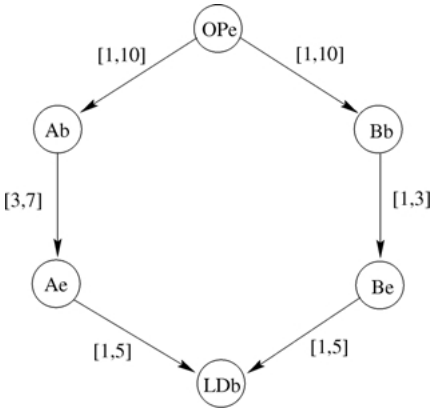

Figure 3. The TCG-graph for the online vendor example.

order processing and not later than 10 time units from it (constraints [1, 10]). Also, the final delivery must begin after all products are available and none of the products must wait more than 5 time units at the warehouse (constraints [1, 5]).

In this case, the enactment service cannot assign A and B with the same earliest possible starting times. Indeed, assume we are using granularity *hours* and suppose the order processing completed at 8 am, and A and B are assigned to start at 9 am. According to duration constraints, activity A could take 7 hours to complete, while activity B could take 1 hour only. Hence, the products shipped by B will have to stand at the warehouse more than the 5 hours allowed by the constraint. To solve this problem, we can delay the starting time for B to 11 am.

3. The generation of temporal schedules

In this section, we consider the enactment schedules in details. We will formally define different classes of schedules and give algorithms to derive these schedules.

Given the problem description in terms of a set of activities to be enacted, the minimum and maximum durations declared by each agent in charge of an activity, and other temporal constraints involving different activities, we want to provide to the enactment service with a *schedule* for these activities. For the sake of simplicity, in the rest of this paper we restrict our technical investigation to the case of networks with all constraints in terms of a single granularity. Section 4 will consider extensions to the multiple granularity case.

We now formally define the notion of a *schedule*.

Definition. A *schedule* for a set of activities $A_1, \ldots, A_k$ whose constraints are represented in a network N, is a set of triples of intervals on positive integers ($[Ebegin_i, Lbegin_i]$, $[m_i, n_i]$, $[Eend_i, Lend_i]$), one for each activity, such that: if for each activity A_i, a pair (x_i, y_i), with x_i in $[Ebegin_i, Lbegin_i]$, y_i in $[Eend_i, Lend_i]$ and (x_i, y_i) satisfies the constraint $[m_i, n_i]$, is used to instantiate the corresponding pair of beginning and ending nodes for A_i in N, then this assignment can be extended to a solution of N.

The first and last intervals in a schedule identify the allowed beginning and ending instants, respectively (L and E stand for Latest and Earliest, respectively), while the second interval identifies the minimum and maximum duration, called the *duration constraint*. Hence, if all the involved activities actually begin and end within the given bounds and do not violate the duration constraints, the schedule guarantees that all the constraints in the network are still satisfiable. The motivation for this definition is that the agents responsible for the activities should be allowed to act independently from other activities. Each agent needs only adhere to a "local" constraint and the global constraint should be satisfiable if each agent satisfies the local constraint attached to it.

As an example, assume *OPe* in figure 3 happens at time 8. Then the following is a schedule for activities A and B: ⟨([9,9],[3,4],[12,13]),([9,9],[1,2],[10,11])⟩. Indeed, since A finishes (i.e., Ae) either at 12 or at 13, and B finishes (i.e., Be) either at 10 or 11, in all the possible cases, it's possible to assign *LDb* a value to satisfy all the constraints in the network.

We say that a schedule is *well-formed* if there are no "redundant" values in each triple. Formally, a triple has no redundant values if for each value in the beginning/ending interval there exists one in the other interval such that the pair satisfies the duration constraint. Moreover, each value in the duration constraint range should be used by one of these pairs. For example, ([9, 10], [3, 4], [12, 13]) is well-formed since (i) each value in the beginning domain can find a value in the ending domain to satisfy the constraint (e.g., 9 finds 12 and 10 finds 13), (ii) each value in the ending domain can find a value in the beginning domain to satisfy the constraint (e.g., 12 finds 9 and 13 finds 9), and (iii) each duration value (i.e., 3 and 4) can find the beginning and ending pairs to have the exact durations (e.g., 9 with 12 and 9 with 13, respectively).

In this section, we assume that each schedule we mention is well-formed unless explicitly stated otherwise.

Note that in the constraint networks and schedules, constraints may involve $+\infty$ or $-\infty$ and domains may be infinite. For simplicity of presentation, we don't explicitly treat ∞ in our exposition. However, it's not difficult to slightly extend the arithmetics to take them into account.

Many different schedules can exist for a given workflow and different criteria can be adopted to identify most interesting ones. When agents for activities operate independently, schedules which do not modify the original duration constraints that these agents declared in the workflow specification, seem to be preferable to those which restrict these duration constraints. If the restriction is unavoidable, a schedule which restricts the maximum duration (hence imposing a deadline) is preferable to one forcing a greater minimal duration. For this reason we formally characterize different types of schedules. We assume for each activity A_i, the duration constraint from $A_i b$ to $A_i e$ in the network is $[min\, D_i, max\, D_i]$.

- A *free schedule* is a schedule such that for each activity, the associated constraint only imposes a time window for the beginning of the activity, while the duration constraint is the original one in the network, and the ending interval is implicit. Formally, a schedule $\langle([Ebegin_1, Lbegin_1], [m_1, n_1], [Eend_1, Lend_1]), \ldots, ([Ebegin_k, Lbegin_k], [m_k, n_k], [Eend_k, Lend_k])\rangle$ for the activities $A_1, \ldots, A_k$ within a constraint network N, is called *free* if (i) the duration constraint in the schedule is the same as that in N (i.e., $m_i = min\, D_i$ and $n_i = max\, D_i$) for each activity A_i, and (ii) $Eend_i = Ebegin_i + m_i$ and $Lend_i = Lbegin_i + n_i$.

In this case the agent of the activity is just told the set of time instants at which it can begin and it simply has to meet the duration constraints as declared in the workflow specification. Referring to Example 1 and assuming the time for *OPe* is 8, we find one of the free schedules is the one assigning 9 and 11 to the beginning of `A` and `B`, respectively. The schedule can be represented as ⟨`([9,9],[3,7],[12,16]),([11,11],[1,3],[12, 14])`⟩. This is clearly a schedule by definition. It is also clear that this is a free one since the duration constraints [3, 7] and [1, 3] are as declared in the constraint network, and condition (ii) is easily verified to be true.

The reason we call this type of schedule *free* is that the agents are actually free to use any amount of the time they declared as long as they start the activities in the assigned interval.

- A *restricted due-time* schedule differs from the free schedule because, in addition to the constraint on the starting time, it may impose as the maximum duration a value smaller than the original one, while preserve the minimum duration bound. Formally, condition (ii) for a free schedule is still required, while condition (i) is changed to (i′): the duration constraint $[m_i, n_i]$ satisfies the conditions $m_i = min\, D_i$ and $n_i \leq max\, D_i$.

Referring to Example 1 and assuming *OPe* occurs at time 8, we see one of the restricted due-time schedules is the one assigning 9 to both the beginning of A and B, but restricting the duration of activity A to 5, i.e., ⟨([9,9],[3,5],[12,14]),([9,9],[1,3],[11,13])⟩.

The motivation for this type of schedule is that the agents will take at least the time declared as the minimum amount in the original network, but may not be able to use the declared maximum amount of time.

- A *bounded* schedule is a schedule only requiring condition (ii) for a free schedule. Hence, it may impose both a greater minimum and a smaller maximum duration for the activities than the declared ones.

In this type of schedule, an agent may be forced to take more time than the declared minimum, but less than the declared maximum amount. An example of bounded-schedule is ⟨([9,9],[3,6], [12,15]),([9,10],[2,3],[11,13])⟩. This is not a free nor a restricted due-time schedule, since the minimum duration for B is changed from 1 to 2.

Note that a general schedule properly includes bounded schedules and it is such that all three intervals must be explicitly specified in the constraint for at least one activity. Consider the same example as above. Then ⟨([9,9],[3,6],[12,15]), ([9,11],[1,3],[11,13])⟩ is a bounded schedule for activities A and B. In this type of schedule the duration that the agent may use for an activity depends on when it starts the work. In the above bounded schedule, if B starts the activity at 11, it can only use at most 2 units of time (it cannot use 3 units).

Proposition 1. *If the constraint network is consistent, there always exists at least one bounded schedule. There is no guarantee, however, that there exists a free or restricted due-time schedule.*

Note that if the network is inconsistent, there does not exist any schedule of any kind for any activities represented in the network.

Among all the possible schedules some are subsumed by others. Here we define the notion of a *maximal schedule*.

Definition. Let $\preceq$ be a partial order over schedules on the same activities defined as follows: $S' \preceq S$ if, for each activity A_i, the constraint on A_i imposed by S' is tighter[6] than that imposed by S. A schedule S is said to be *maximal* if there is no schedule S' on the same activities such that $S' \neq S$ and $S \preceq S'$.

This notion naturally extends to the different types of schedules presented above (free, restricted due-time, and bounded schedules). Examples 3 and 4 in the sequel will show two maximal free schedules.

3.1. Finding free schedules

In this paper, we particularly focus on the derivation of free schedules. We propose a general algorithm for free schedule generation (called the FSG algorithm from now on) which consists of three main steps:

1. decompose the TCG graph according to OR-split operators and derive the minimal network for each resulting constraint network using the consistency algorithm;
2. for each network characterize the set of free schedules for the given set of activities to be scheduled;
3. derive a particular free schedule from the result of Step 2 above according to some criteria.

Deriving the minimal network in Step 1 is performed by the generic consistency algorithm, which guarantees the resulting constraints being as tight as possible while maintaining the same network solution set. If the original duration constraints for the activities to be scheduled are shrunk during Step 1, no free schedule can exist and the algorithm terminates. Indeed, this will mean that one of the minimum (or maximum) duration declared by an agent cannot be used as part of any solution, i.e., if the activity really uses the minimum (or maximum) amount of time as declared, then the whole network is not satisfiable, and some constraints sooner or later will be violated.

When the FSG algorithm is invoked for a schedule of the activities `A` and `B` in Example 1, Step 1 derives the implicit constraints and domains depicted in figure 4.[7] In this case, neither duration was shrunk, i.e., each maintained the original values. This means that the use of

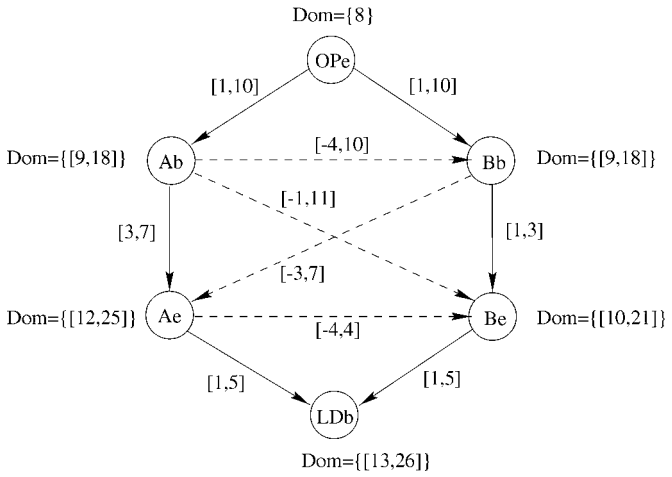

Figure 4. The TCG-graph after constraint propagation.

the minimum or maximum durations are not entirely ruled out by the network, and we can still hope to derive free schedules.

Step 2 is nontrivial and it is supported by some theoretical results.

Theorem 1. *Let $A_1, \ldots, A_k$ be activities to be enacted, and N a minimal network of constraints on these and possibly other activities. The set of all free schedules for $A_1, \ldots, A_k$ in N having a single value as the beginning interval is the set of schedules of the form $([x_i, x_i], [min\, D_i, max\, D_i], [x_i + min\, D_i, x_i + max\, D_i])$ for each activity A_i, where $min\, D_i$ and $max\, D_i$ are the duration bounds for A_i in N, and the values $x_1, \ldots, x_k$ satisfy the conditions*:

- *for each activity A_j, we need $Min(Dom(A_j)) \leq x_j \leq Max(Dom(A_j))$*
- *for each pair of activities $\langle A_i, A_j \rangle$ with $i < j$, we need $x_i + k_{ij} \leq x_j \leq x_i + K_{ij}$, where the constants k_{ij} and K_{ij} are derived from the following*:

$$k_{ij} = Max((max\, D_i - min\, D_j + Min(TCG(A_i e, A_j e))), Min(TCG(A_i b, A_j b)), (Min(TCG(A_i b, A_j e)) - min\, D_j), (max\, D_i - Max(TCG(A_j b, A_i e)))), \quad \textit{and}$$

$$K_{ij} = Min((min\, D_i - max\, D_j + Max(TCG(A_i e, A_j e))), Max(TCG(A_i b, A_j b)), (Max(TCG(A_i b, A_j e)) - max\, D_j), (min\, D_i - Min(TCG(A_j b, A_i e)))).$$

In the above, $TCG(node_1, node_2)$ represents the TCG assigned to the arc from $node_1$ to $node_2$ in N, and $Min(TCG(node_1, node_2))$ and $Max(TCG(node_1, node_2))$ represents the minimum and maximum values of the TCG, respectively.

Note that the theorem characterizes all the single-beginning-point free schedules in N, the minimized network. As mentioned earlier, if the value $min\, D_i$ or $max\, D_i$ in N was changed by the generic consistency algorithm, there does not exist any free schedule. The theorem, however, is still useful since if the system of inequations has any solution, these solutions will give restricted due-time or bounded schedules, depending on whether the generic consistency algorithm has shrunk only maximum duration bounds or some of the minimum ones.

The first condition in the theorem ensures that starting and ending values are included in their corresponding domains. The second ensures that the constraints of the network will be satisfied independently from the specific amount of time taken by each activity within its duration constraint. Note that all values used in the calculations of the constant expressions are given by the minimal network computed in Step 1.

The above result provides an efficient way to find a free schedule. Indeed, the inequations given by Theorem 1 can be expressed in a new constraint network with k nodes, with domains bound by the first inequations and constraints given by the second ones. We apply the consistency algorithm to this network obtaining a minimal network representing the schedule solution space. If we take the minimal value from each domain, the resulting set of values is guaranteed to be a solution [10], and, in terms of our problem, this solution provides the earliest free schedule. Technically, "earliest" here means that that solution identifies the point closest to the origin in the k-dimensional space representing all solutions. Intuitively, it is a schedule with the earliest enactment times.

Example 2. Consider Example 1 and in particular the minimal network as shown in figure 4. Applying Theorem 1 we obtain, from the first condition, $x_A \in [9, 18]$, $x_B \in [9, 18]$,

INPUT: A consistent constraint network (W, A, Γ, Dom) representing the solution space for free schedules.

OUTPUT: An earliest maximal free schedule.

METHOD:
- Let $N' = (W, A, \Gamma', Dom')$ be a copy of the input network
- For $i = 1$ to n Do
  For each node X_j with $i \neq j$ Do
    $Dom'(X_j) := Min(Dom(X_j))$
  $Dom'(X_i) := Dom(X_i)$
  propagate N'
  $newmax_i := max(Dom'(X_i))$
  Restore original constraints in N' (i.e., $\Gamma' := \Gamma$)
  EndDo
- Return $[min(Dom(X_i)), newmax_i]$ for each $i = 1 \ldots n$.

Figure 5. Deriving a maximal free schedule.

and, from the second condition, $2 \leq x_B - x_A \leq 4$. These constraints define a new network having only two nodes for A and B with the corresponding domains and a single arc from A to B labeled by the constraint [2, 4]. When the propagation algorithm is applied, the resulting minimal network is identical except that the domain of A has been shrunk to [9, 16], and that of B to [11, 18]. Indeed, the values 17, 18 for A and 9, 10 for B cannot be part of any solution for this network. According to the procedure illustrated above, the earliest single-valued free schedule is obtained taking as starting instant x_A and x_B, the values 9 and 11, respectively, as they are the minimal values of the minimal domains in the network representing the solution space. It is easily checked that it is a free schedule. Note that this is exactly the example given when free schedules are defined earlier.

3.2. Maximal free schedules and optimization

The schedule we have identified is not necessarily maximal. The procedure described in figure 5 exploits the constraint network representing the free schedules in order to extend the earliest schedule to a maximal one.

Theorem 2. *The procedure illustrated in figure* 5 *terminates returning a unique earliest maximal free schedule.*

The earliest maximal free schedule is the maximal free schedule containing the earliest single-beginning-point free schedule. The basic idea of the algorithm in figure 5 is that any (high-dimensional) rectangular region in the solution space given in Theorem 1 yields a free schedule. The algorithm simply finds the largest rectangular region that includes the earliest point in the solution space.

This procedure implements Step 3 of the FSG algorithm in case of a simple criteria asking for the earliest maximal free schedule.

Example 3. Consider the application of the above procedure to our running example. The input network is the one we derived in Example 2 representing all possible free schedules

with a singleton starting interval. In the first iteration, we consider $Dom'(X_B) = \{11\}$ and $Dom'(X_A) = [9, 18]$. The propagation, considering these domains and the constraint among them, restricts the domain for X_A to the single value 9 which is also the value of $newmax_A$. Note that the constraint among the nodes has been tightened by propagation into [2, 2] and it must be restored to the original value [2, 4] before proceeding. In the second iteration, we consider $Dom'(X_B) = [11, 18]$ and $Dom'(X_A) = \{9\}$. The propagation shrinks $Dom'(X_B)$ to [11, 13]. Hence, [9, 9] and [11, 13] are the intervals defining the maximal earliest schedule for activities A and B, respectively. Therefore, the maximal earliest free schedule is ⟨([9,9],[3,7],[11,16]),([11,13],[1,3],[12,16])⟩.

Theorem 2 also says that the schedule derived by the above procedure is unique. This is not the case in general for maximal free schedules, which may be numerous.

Proposition 2. *The number of maximal free schedules can be equal to the range of the values allowed for a node in the constraint network* (*i.e.*, *beginning or ending of an activity*).

3.2.1. Optimization. The criteria we used to choose among the maximal free schedules is the "earliest" enactment time, which can be reasonable for this application in many situations. However, in some cases we could be interested in deriving an *optimal* free schedule according to different criteria; for example, we may be interested in the maximal free schedule with the largest intervals as starting time windows. This is the choice for a schedule which leaves the maximal freedom on the enactment time, even if this may imply that it is not the earliest possible schedule. In these cases we must include in the system of inequations given by Theorem 1 an objective function reflecting the optimization criteria. Then, in order to find the solutions, we must rely on standard linear programming techniques, which are, in general, less efficient than the one we provide above. For example, to adopt the above criteria, the system of inequations should be solved aiming at maximizing $\prod_{i=1}^{k}(Lstart_i - Estart_i)$.

Example 4. The maximal free schedule which is not the earliest, but it is optimal according to the criteria of largest starting intervals, is the one having [9,10] and [12,13] as starting inte rvals for A and B respectively. Its complete description is ⟨([9,10],[3,7],[12,17]), ([12,13], [1,3],[13,16])⟩

3.3. Finding other types of schedules

In some cases, a free schedule does not exist; intuitively, this is the case when duration bounds provided by agents have quite a wide range and/or when synchronization among activities is quite strict. In these cases, the most reasonable schedules become restricted due-time schedules which impose a due-time to the agents in charge of the activities, by restricting the maximum duration. We have seen that, when Step 1 of the FSG algorithm returns a minimal network which restricts some of the durations bounds, the solutions to the inequations in Theorem 1 identify restricted due-time or bounded schedules. Note that any of the techniques illustrated for free schedules applies to restricted due-time schedules and

to the bounded schedules resulting from restriction of the duration bounds. Indeed, these schedules can be considered as "free" ones with respect to the new duration bounds.

However, even if the minimization in Step 1 does not introduce any restriction on the durations, the system of inequations given by Theorem 1 may still not have any solution. This does not exclude the existence of restricted due-time and bounded schedules. The following proposition provides a necessary condition for restricted due-time schedules.

Proposition 3. *Given a minimal network N and two activities A and B to be enacted, any restricted due-time schedule for A and B in N will have values for* $\max D'_A$ *and* $\max D'_B$ *satisfying the following*:

- $\min D_A \leq \max D'_A \leq (p + s)$ *with*

  $p = Min(Max(TCG(Ab, Bb)), (\min D_A - Min(TCG(Bb, Ae))))$, *and*
  $s = Min((\min D_B - Min(TCG(Ae, Be)), Max(TCG(Bb, Ae)))$
- $\min D_B \leq \max D'_B \leq (r - q)$ *with*

  $r = Min(\min D_A + Max(TCG(Ae, Be)), Max(TCG(Ab, Be)))$, *and*
  $q = Max(Min(TCG(Ab, Bb)), (Min(TCG(Ab, Be)) - \min D_B))$
- $\max D'_A + \max D'_B \leq (r + s)$ *with r and s as above.*

Consult Theorem 1 for the symbols used here. Note that p, q, r and s are constant values derived from N. The conditions in the proposition may be a practical tool to search for restricted due-time schedules.

Example 5. Consider a slight modification to the constraint network of figure 3 where a stricter synchronization is required by constraints $[1, 3]$ from the ending of A and B to the beginning of activity *LDb*. The minimal network is shown in figure 6. Duration

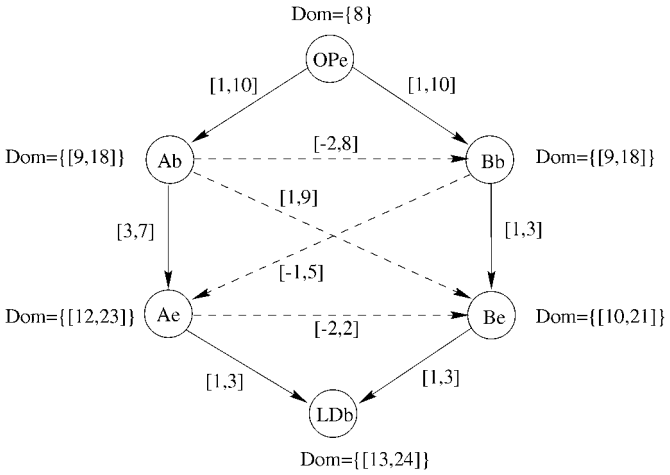

Figure 6. The minimal network for Example 5.

bounds are not restricted by minimization. Hence, we first try to apply Theorem 1 to look for free schedules. It is easily seen that for this network the inequations has no solutions since $k_{AB} = 4$ and $K_{AB} = 2$. By Proposition 3 we may find a restricted due-time schedule restricting the maximal durations of A and B as long as the following is satisfied: $3 \leq max\, D'_A \leq 7$, $1 \leq max\, D'_B \leq 5$ and $max\, D'_A + max\, D'_B \leq 8$. The first two actually do not give any hint in this case since the bounds do not restrict the original ones, but the third one says, for example, that restricting $max\, D_B$ to 2 while leaving $max\, D_A$ untouched does not lead to a restricted due-time schedule. However, if we in addition restrict $max\, D_A$ to 6 we satisfy that condition, and by recomputing k_{AB} and K_{AB} we discover that this time the system has a solution. The same procedure to derive the earliest (maximal) free schedule can be applied, resulting in the following restricted due-time schedule: ⟨([9,9],[3,6],[12,15]),([12,12],[1,2],[13,14])⟩.

A complete characterization of restricted due-time schedules, can be obtained if the constants $max\, D_A$ and $max\, D_B$ are substituted by variables $max\, D'_A$ and $max\, D'_B$, respectively, in the original inequations in Theorem 1, bounding the values of these new variables within the duration bounds given in N. Different optimization criteria can then be applied to restrict the duration bounds.

Corollary 1. *Let A and B be two activities to be enacted, and N be a minimal network of constraints on these and possibly other activities' starting and ending points. Then, the set of all restricted due-time schedules with a single value as the beginning domain is formed by the schedules of the form*

$$\langle([x_A, x_A], [min\, D_A, max\, D'_A], [x_A + min\, D_A, x_A + max\, D'_A]),$$
$$([x_B, x_B], [min\, D_B, max\, D'_B], [x_B + min\, D_B, x_B + max\, D'_B])\rangle$$

with any value for x_A, x_B, $max\, D'_A$, $max\, D'_B$ satisfying the following system of inequations:

- $Min(Dom(Ab)) \leq x_A \leq Max(Dom(Ab))$, *and* $Min(Dom(Bb)) \leq x_B \leq Max(Dom(Bb))$
- $x_A + k \leq x_B \leq x_A + K$, *where the constants k and K are derived from the constraint values*:

$$k = Max((max\, D'_A - min\, D_B + Min(TCG(Ae, Be))), Min(TCG(Ab, Bb)),$$
$$(Min(TCG(Ab, Be)) - min\, D_B), (max\, D'_A - Max(TCG(Bb, Ae)))), \quad and$$
$$K = Min((min\, D_A - max\, D'_B + Max(TCG(Ae, Be))), Max(TCG(Ab, Bb)),$$
$$(Max(TCG(Ab, Be)) - max\, D'_B), (min\, D_A - Min(TCG(Bb, Ae))))$$

- $min\, D_A \leq max\, D'_A \leq max\, D_A$ *and* $min\, D_B \leq max\, D'_B \leq max\, D_B$

Again, consult Theorem 1 *for the symbols used here.*

The result can be easily extended to the case of k activities to be scheduled.

Bounded schedules are less interesting for the enactment of activities since they impose somehow unnatural constraints to the agents participating in the workflow forcing the agents to take more than declared minimum amount of time. Nevertheless, Proposition 1 provides an efficient way to find the earliest bounded schedule for a given network.

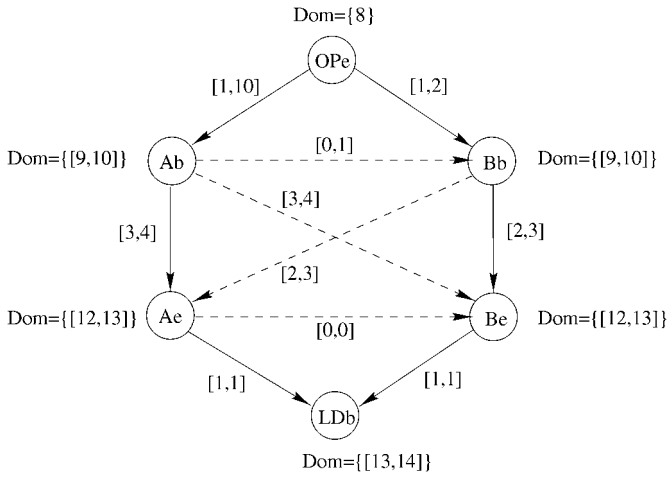

Figure 7. The minimal network for Example 6.

Similar techniques can be used to characterize and derive this type of schedule, but we do not elaborate on that in this paper. We simply provide an example of such schedule in the context of our running example.

Example 6. Consider a slight modification to the constraint network of figure 3 where exact synchronization is required by constraints [1, 1] from the ending of A and B to the beginning of activity LDb. Also, suppose that activity B must be enacted within 2 time units instead of the previous maximal bound 10. The minimal network is shown in figure 7. Step 1 of the algorithm restricts the minimum duration for activity B to be greater than 2. Hence, we know that no free, nor restricted due-time schedule exists in this case. Then, by Proposition 1 we can immediately identify the bounded schedule ⟨([9,9],[3,3],[12,12]),([9,9],[3,3],[12,12])⟩. If we are interested in different ones we may try to apply Theorem 1 to check if there exists a free schedule on these new duration bounds. In this case there is none since $k_{AB} = 2$ and $K_{AB} = 0$. Adopting a technique similar to the one for restricted due-time schedule, we can discover other bounded schedules as, for example, ⟨([9,9],[4,4],[13,13]), ([10,10],[3,3],[13,13])⟩, and ⟨([9,10],[3,4],[13,13]),([10,10],[3,3], [13,13])⟩.

4. Supporting multiple time granularities

In this section, we consider the problem of extending the techniques presented earlier in the paper to the case of constraints in terms of different time granularities. We first formally define granularities and introduce the technical machinery needed to perform temporal reasoning in this context.

4.1. *Temporal constraints with multiple granularities*

The notion of time granularity, as formally defined in [4], is the following.

Definition. A *granularity* is a mapping G from the integers (the *index set*) to subsets of the time domain such that:

(1) if $i < j$ and $G(i)$ and $G(j)$ are non-empty, then each element of $G(i)$ is less than all elements of $G(j)$, and
(2) if $i < k < j$ and $G(i)$ and $G(j)$ are non-empty, then $G(k)$ is non-empty.

The first condition states that granules in a granularity do not overlap and that their index order is the same as their time domain order. The second condition states that the subset of the index set that maps to non-empty subsets of the time domain is contiguous. While the time domain can be discrete, dense, or continuous, a granularity defines a countable set of granules; each granule is identified by an integer.

For example, we can define a special granularity `days starting from 1/1/2001` as follows: `day`(1) is the set of absolute time corresponding to the day 1/1/2001, `day`(2) is the set corresponding to 1/2/2001, and so on. The definition also allows *non-standard* granularities with non-contiguous granules (e.g., `b-day` representing business days) or even with granules consisting of non-convex sets (e.g., `business-month`, where every business month granule is the union of all business days in a month).

There is an important relationship among granularities: A granularity G *groups into* a granularity H, denoted $G \trianglelefteq H$, if for each index j there exists a (possibly infinite) subset S of the integers such that $H(j) = \bigcup_{i \in S} G(i)$. Intuitively, $G \trianglelefteq H$ means that each granule of H is a union of some granules of G. For example, `day` $\trianglelefteq$ `week` since a week is composed of 7 days and `day` $\trianglelefteq$ `b-day` since each business day is a day. Obviously, the set of all granularities is a partial order with respect to $\trianglelefteq$, and in [7] we proved that, for each pair of granularities G and H, there exists a unique greatest lower bound, denoted $glb_{\trianglelefteq}(G, H)$, with respect to $\trianglelefteq$.

In this paper, we assume that for each workflow, there exists a *base* time granularity which has an infinite number of non-empty granules, and which *groups periodically into* any other used in the workflow.[8] Formally, a granularity G *groups periodically into* a granularity H if

1. $G \trianglelefteq H$, and
2. there exist $R, P \in \mathbb{Z}^+$, where R is less than the number of non-empty granules of H, such that for all $i \in \mathbb{Z}$, if $H(i) = \bigcup_{r=0}^{k} G(j_r)$ and $H(i+N) \neq \emptyset$ then $H(i+R) = \bigcup_{r=0}^{k} G(j_r + P)$.

This definition may appear complicated but it is actually quite simple. Since G groups into H, any non-empty granule $H(i)$ is the union of some granules of G; for instance, assume it is the union of the granules $G(a_1), G(a_2), \ldots, G(a_k)$. The periodicity property (condition 2 in the above definition) ensures that the Rth granule *after* $H(i)$, i.e., $H(i+R)$, if non-empty, is the union of $G(a_1 + P), G(a_2 + P), \ldots, G(a_k + P)$. This results in a periodic "pattern" of the composition of n granules of H in terms of granules of G. The

pattern repeats along the time domain by "shifting" each granule of H by P granules of G. P is also called the "period" of H. Many common granularities are in this kind of relationship, for example, `month` and `day` both group periodically into `year`.

In general, this relationship guarantees that granularity H can be finitely described (in terms of granules of G) by providing the following information: (i) the finite sets $S_0, \ldots, S_{R-1}$ of indexes of G each one describing the composition of one of the R repeating non-empty granules of H; (ii) the value of P; (iii) the indexes of first and last non-empty granules in H, if they are not infinite. The description of arbitrary granules of H can be obtained by the following formula:

$$H(j) = \bigcup_{i \in S_{j \bmod R}} G(P * \lfloor j/R \rfloor + i)$$

This formula applies in general, provided that $S_0, \ldots, S_{R-1}$ are the sets of indexes of G describing $H(0), \ldots, H(R-1)$, respectively, and no granule of H is empty, but it can be easily adapted to the case with finite index for first and last non-empty granules.

These granularities can be represented by a user-friendly symbolic formalism, as, for example, the one proposed in [3]. However, the algorithms and results in this paper do not rely on a specific representation but on the conceptual model illustrated above.

To introduce the notion of a temporal constraint with granularity, we need to define a conversion operation among granularities [7]: For each integer i and granularity G, if there exists i' such that $\texttt{base}(i) \subseteq G(i')$, then let $\lceil i \rceil^G = i'$; otherwise, $\lceil i \rceil^G$ is undefined. Note that i' above is unique (if it exists) by the monotonicity of G. Intuitively, $i' = \lceil i \rceil^G$ means that the i'-th granule of G contains i-th granule of the base granularity, e.g., if `base=second`, $\lceil i \rceil^{month}$ gives the month that includes the i-th second.

Definition. Let m, n be integers or symbols in $\{-\infty, +\infty\}$ with $m \leq n$ and G a granularity. Then $[m, n]\,G$, called a *temporal constraint with granularity* or *TCG*, is the binary relation on positive integers defined as follows: For positive integers i_1 and i_2, (i_1, i_2) *satisfies* $[m, n]\,G$ if and only if (1) $\lceil i_1 \rceil^G$ and $\lceil i_2 \rceil^G$ are both defined, and (2) $m \leq (\lceil i_2 \rceil^G - \lceil i_1 \rceil^G) \leq n$.

Intuitively, for time instants i_1 and i_2 (in terms of base), i_1 and i_2 satisfy $[m, n]\,G$ if the difference of the integers i_1' and i_2' is between m and n (inclusive), where $G(i_1')$ and $G(i_2')$ are the granules of G (if exist) that cover, respectively, the i_1-th and i_2-th seconds. That is, the instants i_1 and i_2 are first translated in terms of G, and then the difference is taken. If the difference is at least m and at most n, then the pair of instants is said to satisfy the constraint. For example, the pair of seconds (i_1, i_2) satisfies the TCG $[1, 1]$ `b-day` if i_2 is contained in the next business day with respect to i_1.

Note that, as we have shown in [7], it is not always possible to convert a set of TCGs in terms of multiple granularities into a set of TCGs in terms of a single granularity without changing the meanings of the TCGs.

When the constraint network representing a workflow execution thread contains constraints in terms of different granularities, it is called a TCG network and it is formally defined as follows.

Definition. A *TCG network* over variables $t_1, \ldots, t_k$ is a directed graph (W, A, Γ, Dom), where $W = \{t_1, \ldots, t_k\}$, $A \subseteq W \times W$, Γ is a mapping from A to the TCGs, and Dom is a mapping from W to the periodical subsets of the positive integers. In the above, a set of positive integers S is said to be *periodical* if there exists a granularity G such that the base granularity groups periodically into G and $S = \{i | \lceil i \rceil^G \text{ is defined}\}$.

The notions of solution, consistency, and minimal network are analogous to the single granularity case.

However, the consistency algorithm, adopted for the single granularity case, cannot be straightforwardly extended to deal with general TCGs. The intuition is that distances expressed in terms of granularities denote different absolute values depending on the elements on which the distance is measured. For example, a distance of 1 business-day measured between a Thursday and a Friday is equivalent to 1 day, while it is 3 days between a Friday and the next Monday. Since there is no simple way to reduce a TCG network into one of the constraint networks we used in the previous section without losing information, we have recently proposed a new consistency algorithm specialized for TCG networks [6]. The algorithm is essentially an extension of the arc-consistency algorithm (AC-3) proposed in [14] for networks of constraints with finite domains. The extension allows to deal with possibly infinite (but periodic) domains and with constraints in terms of multiple periodic granularities. We refer to this algorithm as the *TCG consistency* algorithm. In [6] we proved its soundness and completeness, i.e., the input network is consistent if and only if all the domains of the network given as output by the algorithm are non-empty. Moreover, similarly to the single granularity case, the algorithm also provides a solution when the network is consistent:

Proposition 4 [6]. *For a given TCG network N, assume $Dom'(i)$ is the domain in the output network for each variable X_i. Let min_i be the minimum value in $Dom'(i)$ for each i. Then, the assignment of min_i to X_i for each variable X_i is a solution of the network.*

We also proved that the algorithm runs in time polynomial in the number of nodes in the network when a fixed set of granularities is assumed, despite being significantly less efficient than the corresponding algorithm for single granularity networks.

4.2. Reasoning with granularity constraints in the workflow

When multiple granularities are considered, the TCG consistency algorithm provides a tool to check consistency of the overall workflow description. It can be applied in an analogous way with respect to what illustrated in Section 2 for the single granularity case. Similarly, that algorithm is needed to support the prediction temporal service. Hence, each agent can be supplied with a minimum and maximum[9] time at which it could be asked to start and/or end its activity, and the workflow administrator can analyze workflow execution and perform conditional reasoning as described in Section 2.

4.2.1. Deriving schedules in TCG networks. The formal notion of schedule naturally extends to the multiple granularity context: We still have a triple for each activity, but

beginning and ending intervals denote sets of instants in the base granularity, while the duration can be given in terms of a granularity. Despite in general we allow multiple TCGs on each arc, for simplicity here we assume that at most one TCG in terms of a granularity different from the base granularity is given for activity durations. If a TCG in terms of the base granularity is not explicitly given, we assume the existence of the TCG $[1, +\infty]$ in order to enforce activities to have non-zero duration. Hence, a schedule in the case of a single granularity is a special case of a schedule with multiple granularities.

Differently from the previous section, we do not assume schedules to be well-formed. The reason is that some granularities may contain "gaps" in their constraints, while we still use intervals as the begin and ending times in a schedule. For example, (I_b, [0, 1]`business-day`, I_e) may be part of a schedule, where I_b and I_e are two intervals in terms of the base granularity. Since the TCG in business-day is not applicable to instants during weekends, the instants in I_b or I_e that fall into weekends cannot be used as the (real) beginning or ending time for the corresponding activity.

A *free schedule*, analogously to the single granularity case, is such that (i) for each A_i the only constraint in the schedule is the TCG $[min\, D_i, max\, D_i]G_i$ as specified in the original network, and (ii) $Eend_i = Min\{y \mid \lceil y \rceil^{G_i} - \lceil Ebegin_i \rceil^{G_i} = min\, D_i \text{ and } y > Ebegin_i\}$, and $Lend_i = Max\{y \mid \lceil y \rceil^{G_i} - \lceil Lbegin_i \rceil^{G_i} = max\, D_i\}$. Implicitly, we assumed that $\lceil Ebegin_i \rceil^{G_i}$ and $\lceil Lbegin_i \rceil^{G_i}$ are both defined.

A *restricted due-time* schedule is such that for each activity's starting instant, there must be a possible ending instant within any distance allowed by the original constraint, which must be smaller than a certain upper bound, fixed as the due-time restriction. Formally, if $[min\, D_i, max\, D_i]G_i$ is the original TCG in N for A_i, then the duration constraint of a restricted due-time schedule has the form $\{[m_i, n_i], [min\, D_i, newmax\, D_i]G_i\}$ with $newmax\, D_i \leq max\, D_i$ for each A_i, and the following holds: (i) $x + k$ is in $[Eend_i, Lend_i]$ for each $x \in [Ebegin_i, Lbegin_i]$ and each k such that (a) $\lceil x \rceil^{G_i}$ and $\lceil x + k \rceil^{G_i}$ are defined, and (b) $y_1 \leq x + k \leq y_2$, where $y_1 = Min\{y \mid \lceil y \rceil^{G_i} - \lceil x \rceil^{G_i} \geq min\, D_i \text{ and } y \geq x + m_i\}$, and $y_2 = Max\{y \mid \lceil y \rceil^{G_i} - \lceil x \rceil^{G_i} \leq newmax\, D_i \text{ and } y \leq x + n_i\}$. Moreover, (ii) $Eend_i = Max(Ebegin_i + m_i, Min\{y \mid \lceil y \rceil^{G_i} - \lceil Ebegin_i \rceil^{G_i} = min\, D_i\})$ and $Lend_i = Min(Lbegin_i + n_i, Max\{y \mid \lceil y \rceil^{G_i} - \lceil Lbegin_i \rceil^{G_i} = newmax\, D_i\})$. Implicitly, we assumed $\lceil Ebegin_i + m_i \rceil^{G_i}$ and $\lceil Lbegin_i \rceil^{G_i}$ are both defined.[10]

A *bounded* schedule has a duration constraint of the form $\{[m_i, n_i], [m'_i, n'_i]H\}$ where H is an arbitrary granularity allowed in the workflow system, and it satisfies the following: $Eend_i = Max(Ebegin_i + m_i, Min\{y \mid \lceil y \rceil^{H} - \lceil Ebegin_i \rceil^{H} = m'_i\}, Min\{y \mid \lceil y \rceil^{G_i} - \lceil Ebegin_i \rceil^{G_i} = min\, D_i\})$ and $Lend_i = Min(Lbegin_i + n_i, Max\{y \mid \lceil y \rceil^{H} - \lceil Lbegin_i \rceil^{H} = n'_i\}, Max\{y \mid \lceil y \rceil^{G_i} - \lceil Lbegin_i \rceil^{G_i} = max\, D_i\})$. This definition implies that durations allowed by bounded schedules may not only be greater (smaller, resp.) than the minimum (maximum, resp.) original ones, but also exclude, due to the granularity H, some intermediate values.[11]

The notion of *maximality* naturally extends to schedules for networks with TCGs in multiple granularities.

By Proposition 4, we can easily derive a solution by running the TCG consistency algorithm. Then, Proposition 1 extends to the case of multiple granularities, identifying a bounded schedule for each set of activities in a TCG network, if at least one schedule (of any kind) exists for the activities in that network. This is also the earliest schedule.

Moreover, if the TCG constraints resulting from the run of the consistency algorithm have restricted any of the maximal duration bounds for the activities to be scheduled, we know that no free schedule exists, while if any of the minimal bounds have been restricted, no restricted due-time nor free schedule exists.

In general, deriving free and restricted due-time schedules is considerably more difficult in the presence of multiple granularities, due to the lack of the minimality and decomposability property in TCG networks [7], in contrast to the single granularity case. For this reason, we cannot provide a simple characterization of the solution space for restricted due-time or free schedule as we did for the single granularity case. The number of equations to consider is essentially equal to the number of constraints in the network and, due to the presence of multiple granularities, they become non-linear equations. In particular, the task of finding a free schedule seems to require, in the worst case, time exponential in the number of activities to be scheduled.

However, we provide efficient techniques to derive the earliest restricted due-time schedules. The derivation algorithm is shown in figure 8. Intuitively, the algorithm finds a solution with the minimum duration allowed by the duration constraints in the network. Note that when the duration has a TCG in terms of a non-base granularity, care needs to be taken to derive the minimum duration. For example, if the duration constraint says $[1, 4]$`b-day`, then, an activity that takes the minimum duration is an activity that finishes at the first instant of the next business day. This is the reason behind step 1.2 in figure 8.

Instead of implementing the domain restriction of Step 1.2 for activity A_i, it is sufficient to add to the network an extra node X_i and a TCG $[0, \infty]G_i''$ from X_i to Ae_i, where G_i'' is the granularity whose granules correspond to the required set of values, i.e., the first instants of each granule of G_i.[12] When this is done, the consistency algorithm will automatically perform the domain restriction. The sets $S_0 \ldots S_{R-1}$ belong to the definition of G_i (see

INPUT: A consistent TCG network N for activities A_i with $i = 1 \ldots n$ to be scheduled, whose beginning and ending instants are represented by nodes Ab_i and Ae_i respectively, with $[minD_i, maxD_i]G_i$ being the related duration TCG.

OUTPUT: The earliest restricted due-time schedule if one exists.

METHOD: Let N' be a copy of the input network N

1. `For` $i = 1$ `to` n `Do`
   1.1 Restrict the TCG for A_i to its minimal value:
   $TCG(Ab_i, Ae_i) := [minD_i, minD_i]G_i$
   1.2 `If` $minD_i = 0$ `Then` $TCGs(Ab_i, Ae_i) = \{[0,0]G_i,\ [1,1]\}$
   `Else`
   Restrict $Dom(Ae_i)$ to the set of values corresponding to the first instant of each granule of G_i:
   $Dom(Ae_i) := Dom(Ae_i) \cap \bigcup_{j \in Z} base(P_i * \lfloor j/R_i \rfloor + Min(S_{j \bmod R_i}))$
2. apply consistency algorithm to N'
   `If` consistent `Then Return`
   $[Min(Dom'(Ab_i))], [Min(Dom'(Ae_i) - Min(Dom'(Ab_i))], [Min(Dom'(Ae_i))]$
   for each $i = 1 \ldots n$ (This is a restricted due-time schedule)
   `Else Return` False (no restricted due-time schedule)

Figure 8. Due-time schedule derivation algorithm.

INPUT: A TCG network N, and the schedule for activities $A_1, \ldots, A_n$ in N resulting from the Due-time Schedule Derivation algorithm.

OUTPUT: The earliest restricted due-time schedule with maximal durations for $A_1, \ldots, A_n$.

METHOD: Let $([x_i], m_i, [x_i + m_i])$ for each A_i be the input restricted due-time schedule.

1. `For` $i = 1$ `to` n `Do`
   Let N^i be a copy of N with $Dom^i(Ab_j) = \{x_j\}$ and $TCG(Ab_j, Ae_j) = [m_j, n_j]$ with $n_j = m_j$ for each $j = 1 \ldots n$.
   `Repeat`
      $newn_i := n_i$
      $n_i := n_i + 1$
      `While` $\lceil x_i + n_i \rceil^{G_i}$ undefined `Do`
         $n_i := n_i + 1$
      apply consistency algorithm to N^i
   `Until` inconsistent `Or` $x_i + n_i > Max\{y \mid \lceil y \rceil^{G_i} - \lceil x_i \rceil^{G_i} = maxD_i\}$
2. $newmaxD_i := \lceil x_i + newn_i \rceil^{G_i} - \lceil x_i \rceil^{G_i}$ for each $i = 1 \ldots n$
   `Return` $[x_i], \{[m_i, newn_i], [minD_i, newmaxD_i]G_i\}, [x_i + m_i, x_i + newn_i]$ for each $i = 1 \ldots n$.

Figure 9. Maximal durations due-time algorithm.

Subsection 4.1). Note that the algorithm can be optimized by considering the fact that if an activity does not need synchronization (i.e., no constraints with other activities in the original graph), or if G_i is a subgranularity[13] of the base granularity, then no restriction to the ending domain is needed.

Theorem 3. *The algorithm in figure* 8 *is correct.*

Among the earliest restricted due-time schedules we would like to have the least restricted, i.e., providing the latest possible due-time. In figure 9 we propose an algorithm performing this task. The idea is to extend the due-time of one activity at a time. Exploiting the properties of schedules, this process for one activity can be done independently with other activities. Hence, the process is still efficient (i.e., no combination of activities needs to be considered). The algorithm simply exploits this fact. Indeed, to extend one activity, the algorithm starts with a copy of the network from the restricted due-time schedule derivation algorithm (in which each beginning/ending domain consists of one value and each duration only has one value). The algorithm then repeatedly extends the due-time for that particular activity as long as it is possible. Then, it tries to extend the due time of the next activity with respect to the original input network.

Theorem 4. *The algorithm in figure* 9 *is correct.*

A natural question concerns the possibility to extend the schedule we have derived in order to obtain a *maximal* schedule. The algorithm in figure 10 shows how this can be done.

In Step 1 the algorithm considers each activity A_i independently, and then checks (Step 1.1.1) if the next instant for starting activity A_i still leads to a restricted due-time schedule when all other activities starting and ending instants are fixed accordingly to the earliest restricted due-time schedule derived by the algorithm in figure 8. If this is the case,

INPUT: A TCG network N, and the schedule for activities $A_1, \ldots, A_n$ in N resulting from the Maximal Durations Due-time Algorithm.

OUTPUT: The maximal earliest restricted due-time schedule with maximal durations for $A_1, \ldots, A_n$.

METHOD: Let $([x_i], Dur_i, [x_i + m_i, x_i + n_i])$ for each A_i be the input restricted due-time schedule, with $Dur_i = \{[m_i, n_i], [minD_i, newmaxD_i]G_i\}$.

1. `For` $i = 1$ `to` n `Do`
   Let N^i be a copy of N with $Dom^i(Ab_j) = \{x_j\}$ for each $j = 1 \ldots n$, and $Dom^i(Ae_j) = \{x_j + m_j\}$ for each $j \neq i$.
   1.1.`Repeat`
      $Dom^i(Ab_i) := \{Min(Dom^i(Ab_i)) + 1\}$
      1.1.1. apply Due-time Schedule Derivation Algorithm to N^i
      1.1.2. `If` consistent `Then`
         apply Maximal Durations Due-time Algorithm
   `Until` N^i is inconsistent `Or` for some A_j new duration constraints $\neq Dur_j$.
2.`Return` $[x_i, Min(Dom^i(Ab_i)) - 1], Dur_i, [x_i + m_i, Min(Dom^i(Ab_i)) - 1 + n_i]$ for each $i = 1 \ldots n$.

Figure 10. Maximizing a restricted due-time schedule in a TCG network.

analogously to what was done for the original starting value x_i, Step 1.1.2 tries to extend the durations for all activities accordingly to this new restricted due-time schedule. Clearly, the resulting maximal duration ranges must be the same as those derived for the original schedule in order to extend the set of starting instants for A_i with the new value. The loop in Step 1.1 is repeated until this condition is violated, or until the next starting value does not lead at all to a restricted due-time schedule. Then, the next activity to be scheduled is considered. In the end (Step 2) a schedule is returned, with the starting interval for each activity including the maximum that could be derived for that activity, the original largest duration ranges, and the implicit ending domains. Note that all TCG networks considered in the algorithm have a single instant in each node domain corresponding to the start of an activity (*Min*( ) is only used in the algorithm to access this value). The algorithm is efficient since it extends the beginning interval for each activity independently from other extension steps, i.e., the algorithm does not need to consider the effect of the extension of other activities to the one it is working on.

Theorem 5. *The algorithm in figure* 10 *is correct.*

4.2.2. Extension to free schedules. Note that the algorithm in figure 9 may also succeed in extending the due-time up to the original maximal duration (*max D*) for each activity. Hence, an easy check on the algorithm's output could reveal that the output schedule is actually a free one. It is easily seen that this would be the earliest free schedule. However, a free schedule may still exist even if none is detected by that algorithm. Unfortunately, a complete algorithm for free schedules may have to involve a combinatorial search among the candidate starting instants.[14] As in many constraint satisfaction problems, consistency algorithms are good preprocessing techniques to limit the combinatorial search [10]. In particular, the search space can be greatly reduced considering candidate starting instants from the domains of the corresponding nodes, obtained by the intersection of those resulting from separate

executions of the consistency algorithm with minimum and maximum durations needed by free schedules. In practice, we expect that the synchronization constraints between activities force these resulting domains to be very restricted. Moreover, a number of heuristics may be applied to optimize the search.

4.3. Efficiency considerations

All the algorithms illustrated for single granularity networks are polynomial ($O(n^3)$) in the number of nodes (n) in the network, hence in the number of activities. Considering multiple granularities, from [6] the TCG consistency algorithm takes time polynomial in the number of variables in the network and in the size of the maximum finite domain. However, the efficiency of that algorithm can be greatly improved. A first optimization can be obtained by representing any periodical set and granularity in terms of $glb_{\trianglelefteq}(\ )$ of the granularities appearing in the network (including domains) instead of the `base` granularity. For example, if the only granularities appearing in the network are `week` and `day` (assume all positive integers as domains), then any solution (and operation performed by the algorithm) can be in terms of `day` instead of seconds, greatly reducing the time needed to check consistency.

It is easily seen, that the algorithms to generate bounded and due-time schedules for TCG networks are also polynomial.

5. Related research

A large amount of work has been done on workflow systems across the fields of CSCW (Computer Supported Cooperative Work) and advanced transaction models. Functionality and limitations of workflow systems have been analyzed in several papers (see, for example, [1, 2]). An effort to provide a "consensus" workflow reference model [13] has been made by The Workflow Management Coalition, gathering ideas from researchers in the field. Most terms and notions on workflow systems used in this paper are from this model. Up to now, however, little attention has been given to modelling advanced temporal features for workflow systems, with some exceptions. Casati et al. [8] describes the WIDE system, developed in the context of an European project on workflow systems. In that paper, the definition of temporal information is considered of great importance for these systems, particularly for handling temporal exceptions. Primitives are introduced to model time intervals, durations and periodic conditions, but there is no investigation about reasoning with temporal constraints in this domain. Reasoning with temporal constraint networks in the context of workflows has been proposed in [12] to track individual tasks subject to constraints with other tasks in the workflow. However, that paper does not address the technical problems of supporting time frame prediction and scheduling of tasks.

Two recent papers follow research directions closer to ours. The authors of [15] propose a framework for time modeling in production workflows. They consider the same constructs for workflow specification as we do and similar temporal constraints. Duration constraints in the form [*min*, *max*] for each involved activity are integral part of a workflow specification. They provide an efficient algorithm to check that any other type of constraint that may be needed (deadline and inter-task constraints) is "implied" by the duration constraints and the

workflow structure. This is similar to checking in our framework that a free schedule retains its "free" property upon the addition of certain constraints. The main differences with our work are the following: (a) we consider a more general set of constraints as part of the workflow specification (e.g., constraints among different activities, indeterminate bounded order, like $[-1, 1]$ `day`, and others) which make the notion of "freeness" (which they call "consistency") more involved. The price we pay for the additional expressiveness is in terms of computational cost, since the path consistency algorithm cannot be easily generalized to deal with the OR split and joins as they do with the shortest-path algorithm; (b) We don't simply check the workflow specification consistency, but we give rules in order to achieve freeness assigning specific starting times during workflow enactment, and, when this is not possible, we suggest how to impose restrictions on activity durations in order to make it possible.

The second paper, [11], is also concerned with temporal constraints reasoning and management. However, the temporal constraints in [11] form a subclass of those considered in this paper, due to the "well structured" requirement in [11]. The deadline reasoning in both build-time and run-time of [11] can be seen as a special case of this paper. In particular, the reasoning about the allowed "buffer time" for parallel activities is a subcase of deriving free schedules. Furthermore, the fixed-date constraints (e.g., some event must happen on a Friday), which are not dealt with in [11], are considered in this paper in a more general framework of workflow with constraints in multiple granularities (e.g., a TCG may be in terms of granularity `Friday`).

Commercial workflow systems (as reviewed, e.g., in [2]) are usually limited to the specification of a deadline for each activity or global plan. In some cases more elaborate temporal conditions can be specified, but no reasoning on these conditions is supported other than run-time evaluation.

The temporal constraints used in this paper can be seen as a generalized version of "gap-order" constraints as introduced in [16]. In particular we allow both a minimal and a maximum bound on the distance between two variables. The notion of temporal constraint network used in this paper is deeply investigated in [10], and extended with multiple granularities in [7]. Arc consistency algorithms have only been defined for single granularity networks [14].

The literature on the subject of temporal reasoning with multiple granularities is surprisingly scarce. In most cases, either the multiple granularities are not considered (e.g., [10]) or it is assumed that constraints in terms of multiple granularities can be equivalently translated in terms of a single granularity (e.g., [9]). However, it is not always possible to convert a temporal distance among two events in terms of one granularity into one in terms of another granularity, unless the time occurrence of the first event is already known. For example, suppose we require that two events must occur within the same day. We cannot equivalently translate this distance as being "less than 24 hours", since two events that are apart by only one hour may very well occur in two different days. Technically, we showed in [7] that a problem known to be in P-TIME for single granularities becomes NP-hard with arbitrary multiple granularities. In [6], we investigated the problem of consistency checking of temporal constraints with multiple time granularities. In Section 4 we applied some of the results and techniques to specific problems arising in a workflow system. A comprehensive

presentation of our granularity model and reasoning techniques as well as examples of their applications can be found in [5].

Regarding the scheduling problem, a rich literature on this subject exists (see [18] for a good survey). However, our algorithms exploit the particular structure of the temporal constraints to solve the problems addressed in the paper, and, for this reason they have much better computational properties than general purpose scheduling algorithms.

6. Conclusion

In this paper, we studied temporal reasoning for workflow systems. In particular, we provided algorithms for the consistency, prediction and enactment services. Especially, we investigated the enactment scheduling problem in details, formally classifying the schedules according to their usefulness in the workflow. The notion of free schedule, in particular, is a relevant contribution of this paper, together with the efficient algorithm for its computation.

Once activities are instantiated, the workflow management still needs to monitor the process. Indeed, if one of the activities is not finished after its (implicit or explicit) due-time, an exception condition occurs. The temporal reasoning performed on the TCG-graph provides a tool for an early detection of constraint violation. For this purpose, we may introduce the concept of *guarding time*. At any time in an instantiated workflow process, this time represents the instant such that if no event occurs before this instant an exception should occur, since we are guaranteed that at least one constraint has been violated. To derive this guarding time is an interesting research problem.

In addition to the three types of the schedules we studied, a fourth could be devised, for workflow models in which agents can autonomously synchronize (without involvement from the workflow management) with each other while performing their tasks. In this case, more relaxed bounds can be given to the agents, together with the synchronization constraints they must satisfy during the execution. This will be another interesting further research direction.

Appendix

A.1. Proof of Proposition 1

Since the network is consistent, there exists a solution. For n activities to be enacted, the solution will contain the values tb_i, and te_i for $i = 1 \ldots n$ corresponding to beginning and ending of activity $\mathtt{A}_i$. The schedule containing, for each activity, the triple $([tb_i, tb_i], [te_i - tb_i, te_i - tb_i], [te_i, te_i])$ is a bounded schedule. Example 6 in the paper shows a consistent network not admitting free nor restricted due-time schedules.

A.2. Proof of Theorem 1

The proof of this theorem needs some preliminary definition and results.

Definition. A pair of intervals I and I' is called *sync-free* with respect to a constraint on the distance between elements of I and elements of I', if each pair (x, y) with $x \in I$ and $y \in I'$ satisfies the constraint.

For example, let $I = [9, 10]$ and $I' = [11, 13]$. Then I and I' are sync-free with respect to the constraint $[1, 4]$, but not sync-free with respect to $[2, 4]$, nor $[1, 3]$.

Lemma 1. *A pair of intervals* $I = [min\, I, max\, I]$, $I' = [min\, I', max\, I']$ *is sync-free with respect to constraint* $[m, n]$ *from* I *to* I', *if and only if*: $(min\, I' - max\, I) \geq m$ *and* $(max\, I' - min\, I) \leq n$

Proof: Since intervals are sets of contiguous values with $min\, I \leq max\, I$ and $min\, I' \leq max\, I'$, it is sufficient to check that the maximal distance between any two elements of the intervals $(max\, I' - min\, I)$ is less than n and that the minimal distance $(min\, I' - max\, I)$ is greater than m. □

Lemma 2. *Let S be a well-formed schedule for activities* $A_1, \ldots, A_n$ *in a network* N, A_i *and* A_j *be any two of these activities*, $I(A_i)$ *the beginning or ending interval in S for* A_i, *and similarly* $I(A_j)$ *for* A_j. *Then*, *the intervals* $I(A_i)$ *and* $I(A_j)$ *are sync-free with respect to the temporal constraint between the corresponding nodes in* N.

Conversely, *given a minimal network N including the constraints among n activities*, *any set of triples* $([Ebegin_i, Lbegin_i], [m_i, n_i], [Eend_i, Lend_i])$ *for* $i = 1 \ldots n$ *with the values in the first and last intervals included in the domain of the corresponding node in* N, *and the second included in the corresponding constraint in* N, *is a schedule if the above sync-free property holds on each pair of beginning/ending intervals of different activities with respect to the corresponding constraint in* N.

Proof: By schedule definition, if a pair (x_i, y_i), for each activity A_i, with x_i from the first interval, y_i from the third, and $y_i - x_i$ from the second, is used to instantiate the corresponding pair of beginning and ending nodes in N, then this assignment can be extended to a solution of N. By well-formedness, any value in the starting and any value in the ending interval is part of one of the pairs (x_i, y_i). Since any combination of these pairs, for different activities, can be chosen, this means that the network N has a solution for any combination of values in the considered intervals. Indeed, any starting value of activity A_i must belong to a solution together with each value of activity A_j for all $i \neq j$, and, by well-formedness, the same must holds for the ending points. This also implies that any beginning value will be in a solution with any ending value for another activity. Thus, the sync-free property holds for each pair of beginning/ending intervals of different activities.

To prove the second part of the lemma we have to show that any set of triples with those properties is indeed a schedule. The proof is based on a property of minimal networks, namely decomposability, that ensures that any partial solution (i.e., an assignment to a subset of the variables which satisfies the constraints among these variables) can always be extended to a global solution. Also, by hypothesis, any value in the beginning and ending interval belongs to the (minimal) domain of the corresponding node in N. Then, by the sync-free property of these intervals, any n-tuple $\langle (x_1, y_1), \ldots, (x_n, y_n) \rangle$ where x_i, y_i

are values from starting and ending interval, respectively, of the same activity A_i in the candidate schedule, is such that all constraints among the corresponding nodes in N are satisfied. Hence, it is a partial solution for N. Then, by the decomposability property of N, there exists a global solution including this assignment. This is the property required by the schedule definition. Indeed, each pair (x_i, y_i) considered above is such that $y_i = x_i + d_i$ where d_i is a value in the network constraint corresponding to the duration of A_i. Since the network is minimal, each value in that constraint participate in one of these sums, and, since the duration interval in the schedule has, by hypothesis, the same bounds as this constraint, all pairs with any possible value from the duration interval are considered. □

We can now prove the theorem. We initially consider the case of only two activities A and B to be enacted. We show that any free schedule with a single starting value, has the properties stated in the theorem: By definition, all free schedules have the form given in the theorem. Regarding the two conditions, the first is clearly satisfied, since any value outside the corresponding domain in N would not be part of a solution, hence it cannot be part of a schedule. The second condition is derived by Lemma 1 considering the four pair of nodes involved in the scheduling of two activities:

$$\begin{aligned}
x_B - x_A &\geq Min(TCG(Ab, Bb)) \\
x_B - x_A &\leq Max(TCG(Ab, Bb)) \\
x_B + min\, D_B - x_A &\geq Min(TCG(Ab, Be)) \\
x_B + max\, D_B - x_A &\leq Max(TCG(Ab, Be)) \\
x_A + min\, D_A - x_B &\geq Min(TCG(Bb, Ae)) \\
x_A + max\, D_B - x_B &\leq Max(TCG(Bb, Ae)) \\
x_B + min\, D_B - (x_A + max\, D_A) &\geq Min(TCG(Ae, Be)) \\
x_B + max\, D_B - (x_A + min\, D_A) &\leq Max(TCG(Ae, Be))
\end{aligned}$$

A simple elaboration of these equations gives the ones in the thesis. Lemma 2 ensures that any schedule satisfies that sync-free condition.

Moreover, by the second part of Lemma 2 any set of triples with the form and properties considered in the theorem, is a free schedule. Indeed, the first condition ensures that each value in the beginning interval is in the corresponding domain in N, freeness ensures the duration interval is the same as the constraint in N, and, since the network is propagated, this implies that the ending interval is also included in the corresponding domain. Finally, the second condition in the theorem ensures the sync-free property. Considering now n activities to be scheduled, since we are dealing only with binary constraints, the set of inequations for each pair of variables is equivalent to the global condition of schedule freeness. It is correct to consider only pairs with $i < j$ since we need to consider each arc between the endpoints of two activities in the network only once (in one direction).

A.3. Proof of Theorem 2

The proof needs the following preliminary result.

Lemma 3. *Let S be a free schedule in a constraint network N. S is maximal if, for each scheduled activity, any extension of a beginning interval, leads to a non-schedule.*

Proof: The proof relies on the sync-freeness property of schedules stated in Lemma 2 above. Given a pair of sync-free intervals, excluding a minimum or maximum value from one of them, clearly still leads to a pair of sync-free intervals. Applying this fact to schedules, we derive that excluding a minimum or maximum value from one of the beginning intervals associated with an arbitrary activity, which has at least two values, still gives a schedule. Indeed, excluding such a value from one of the intervals, and reconsidering all the pairs according to the schedule definition, they still define a set of sync-free intervals, and represent a partial solution for all the nodes corresponding to the scheduled activities. Moreover, since these partial solutions are a subset of those identified before the value exclusion, these can also be extended to a global solution. Then, suppose that S is a schedule satisfying the condition in the lemma, and that, by contradiction, a different free schedule S' exists for the same activities and network such that $S \preceq S'$. Then, by definition of $\preceq$, there exists an interval in S' for one of the activities which properly contains the corresponding one in S. This interval cannot be a duration one, since we are considering free schedules, for which the duration is fixed. Moreover, for free schedules beginning and ending intervals are tightly related by the durations so that if this interval is an ending one, the beginning one for the same activity will also be properly included in the corresponding one of S'. Then, the beginning interval in S' can be shrunk to match the one in S (and, consequently, the ending ones), maintaining the properties of a free schedule accordingly to what shown above. Repeating this process for each mismatching interval, S' can be reduced to S. In the last step a schedule is reduced to S by shrinking a beginning interval by a single unit. This contradicts the fact that any extension of the beginning intervals in S leads to a non-schedule. □

We can now prove the theorem. Termination is ensured by the finite number of nodes and by the termination property of the consistency algorithm. We prove that the set of time-windows returned by the procedure defines a free schedule showing that each n-tuple obtained by taking a value from each time-window in the output is a solution to the input network. Indeed, any solution to the input network is guaranteed to be a free schedule. Then, if any combination is a free schedule, their union in the form of a time-window for each activity's starting instant is also a free schedule. For each X_i with $1 \leq i \leq n$, $newmax_i$ is derived by the algorithm in the i-th iteration of the main `For` statement, so that the n-tuple made by any value in the interval $[min(Dom(X_i)), \ldots, newmax_i]$ together with $min(Dom(X_j))$ for each X_j with $j \neq i$ is a solution (hence, a free schedule). Consider now two arbitrary nodes X_i and X_j. As we have seen, the pair $(newmax_i, min(Dom(X_j)))$ is part of a solution, as well as (symmetrically) $(min(Dom(X_i)), newmax_j)$. Then $(newmax_i, newmax_j)$ is also part of a solution; indeed, suppose the constraint from X_i to X_j is $[m, n]$, then, by the two solutions above, we have $m \leq (min(Dom(X_j)) - newmax_i) \leq n$ and $m \leq (newmax_j - min(Dom(X_i))) \leq n$. Since $newmax_i \geq min(Dom(X_i)$ we have $(newmax_j - newmax_i) \leq n$, and, analogously we derive $m \leq (newmax_j - newmax_i)$. This result is trivially extended to n variables, since we are dealing with binary constraints, and it also shows that any combination of values between $min(Dom(X_i))$ and $newmax_i$ for each i is a solution.

Since the output schedule contains the minimal value for each domain, the schedule is among the earliest schedules.

Finally, by Lemma 3, the schedule is maximal if none of its time-windows can be extended remaining a (free) schedule. Suppose that the time window for activity A_i in the output schedule can be extended to $newmax_i + 1$. By the soundness of the propagation algorithm we know that $newmax_i$ is the maximum value satisfying the network constraints when $min(Dom(X_j))$ for each $j \neq i$ is assigned as the domain of each node X_j. Since the network represents the solution space for free schedules, this implies that that extension is not a free schedule.

The schedule is also unique, since the earliest schedule is unique and its extension to a maximal one, computed by the above procedure, is independent from the order of the nodes considered in the network.

A.4. Proof of Proposition 2

Consider the class of networks with activities `A` and `B`. Let $[x, y]$ be the range of values for the beginning of `A`, `1` be the duration of both `A` and `B`, and an additional constraint imposes B ends exactly 1 time unit after A ends. There are $y - x$ maximal free schedules: take any value `t` with $x \leq t \leq y$, then it is easily checked that ⟨`([t,t],[1,1],[t+1,t+1])`, `([t+1,t+1],[1,1],[t+2,t+2])`⟩ is a maximal free schedule.

A.5. Proof of Proposition 3

A due-time schedule can be considered as a free schedule with respect to a constraint network where maximal duration bounds have been restricted. The three equations in the proposition are obtained considering the second condition in Theorem 1, which requires $k_{AB} \leq K_{AB}$ to admit a free schedule in its input network. Substituting constants $max\,D_A$ and $max\,D_B$ by variables $max\,D'_A$ and $max\,D'_B$, that requirement implies that the equations in the proposition must be satisfied.

A.6. Proof of Corollary 1

The main idea is that of modifying the original maximal duration bounds so that, considering the new bounds as if they were the original ones, a set of free schedules can be derived. Obviously these schedules will be restricted due-time ones with respect to the original network.

A.7. Proof of Theorem 3

We first show that the output is a restricted due-time schedule. If a solution of the resulting network exists, taking the minimum of each domain, we identify n pairs of instants for starting and ending the n activities such that all the constraints, including durations, are satisfied. These n pairs are actually the output of the algorithm, and since they are part of a global solution they satisfy the schedule property. To prove that the schedule

is of type *restricted due-time* conditions (i) and (ii) in the definition must be satisfied. (ii) is clearly satisfied by the form of the output. To show that (i) is satisfied, we prove the following: for each A_i, $Min(Dom'(Ae_i)) = Min\{y \mid \lceil y \rceil^{G_i} - \lceil Min(Dom'(Ab_i)) \rceil^{G_i} = min\, D_i$ and $y > Min(Dom'(Ab_i))\}$. Indeed, $y_1 = y_2 = Min(Dom'(Ae_i))$. If $min\, D_i = 0$, then the algorithm adds $TCG(Ab_i, Ae_i) = [1, 1]$ in N' by Step 1.2. Then, if N' is consistent, $Min(Dom'(Ae_i)) = Min(Dom'(Ab_i)) + 1$, and no smaller value can satisfy the TCGs with $Min(Dom'(Ab_i))$. Since one of these TCGs is $[0, 0]G_i$, condition (i) is satisfied in this case. Let us consider now $min\, D_i = k > 0$. The value $Min(Dom'(Ae_i))$, assigned by the algorithm to Ae_i, is part of a solution when $Min(Dom'(Ab_i))$ is assigned to Ab_i, hence $\lceil Min(Dom'(Ae_i)) \rceil^{G_i} - \lceil Min(Dom'(Ab_i)) \rceil^{G_i} = k$ since the duration TCG for A_i was shrunk by the algorithm to $[k, k]G_i$. Since $k > 0$ this also implies $Min(Dom'(Ae_i)) > Min(Dom'(Ab_i))$. $Min(Dom'(Ae_i))$ is also the minimum of the values with these properties, since, by the restriction of the domain of Ae_i, there exists a single candidate value for each granule of G_i, and hence, a single value in that domain falling in the kth granule of G_i after that in which $Min(Dom'(Ab_i))$ is included.

We now show completeness, i.e., if a restricted due-time schedule exists, one is found by this algorithm. This is equivalent to show that if the algorithm returns "no due-time schedule", actually, there exists none. Suppose the algorithm returns "no due-time schedule", but there exists a restricted due-time schedule S. Then, if $Ebegin_i$ is the first instant of its beginning interval, and $[Eend_i, Lend_i]$ its ending interval, for each activity A_i, using condition (i) in the definition of restricted due-time schedules, for each A_i, there exists $y_i \in [Eend_i, Lend_i]$ such that $y_i = Min\{y \mid \lceil y \rceil^{G_i} - \lceil Ebegin_i \rceil^{G_i} = min\, D_i$ and $y > Ebegin_i\}$. Suppose for each i, $min\, D_i > 0$. Then, this y_i value must be the first instant of a granule of G_i, otherwise the $Min(\ )$ condition would be violated. By the schedule definition, we also know that the set of pairs $(Ebegin_i, y_i)$ is part of a network global solution. Note that the network N', as defined by the algorithm, actually imposes, for each A_i, its duration TCG to be the value $min\, D_i$ in terms of G_i, and each value in $Dom(Ae_i)$ to be the first instant of a granule of G_i. Then, the existence of schedule S contradicts the completeness of the consistency algorithm run on N', which ensures that no global solution exists for N', i.e., there exists no set of pairs (x_i, y_i) assigned to the beginning and ending node of each A_i and satisfying the above conditions which is part of a solution of the constraint network. We now extend the result to the case in which $min\, D_i = 0$ for some of the activities. If that is the case, then condition (i) in the restricted due-time schedule definition implies that the minimum y falling in the same granule of G_i and satisfying $y > x$ is $y_i = Ebegin_i + 1$. This is reflected in N' by the conditional statement in Step 1.2 of the algorithm. Hence, again, the completeness of consistency of N' contradicts the existence of S.

We now show that the derived schedule is also the earliest. By construction, it identifies the minimal starting instants admitted in the network for the minimal duration of each activity, but still remains to be shown that no earlier instants are admitted when relaxing the duration. Consider an activity A_i which is assigned value x as starting instant by the schedule derived by the algorithm, and suppose there exists a restricted due-time schedule S whose starting interval for A_i includes $x' < x$. Then, by condition (i) of restricted due-time schedule, and by that of schedule, the pair $(x', Min\{y \mid \lceil y \rceil^{G_i} - \lceil x' \rceil^{G_i} = min\, D_i$ and $y > x'\})$ must be part of a global solution of N. Then, by what we have shown in the first part of this proof,

x' should be in the domain identified by the consistency algorithm in Step 2. However, this is a contradiction since $x' < x$ and x was taken as the minimum of that domain.

A.8. Proof of Theorem 4

The correctness of this algorithm is based on a new result that we provide for TCG networks, and which is independent from the specific application.

Lemma 4. *Let N be a TCG network with nodes $X_1, \ldots, X_n$, $(X_1 = t_1, \ldots, X_k = t_k)$ and $(X_1 = s_1, \ldots, X_k = s_k)$ be two partial solutions which can be extended to a global solution. Then, the assignments $(X_1 = min(t_1, s_1), \ldots, X_k = min(t_k, s_k))$ and $(X_1 = max(t_1, s_1), \ldots, X_k = max(t_k, s_k))$ are also partial solutions that can be extended to a global solution.*

Proof: Consider the first candidate $(X_1 = min(t_1, s_1), \ldots, X_k = min(t_k, s_k))$, and let $X_i = t_i$ and $X_i = s_i$ for each $i = 1, \ldots, n$ be the two global solutions which, by hypothesis, include the assignments $(X_1 = t_1, \ldots, X_k = t_k)$ and $(X_1 = s_1, \ldots, X_k = s_k)$. We claim that $X_i = min(t_i, s_i)$, which includes the candidate assignments, is also a global solution. Indeed consider an arbitrary pair of nodes X_i and X_j and let $[m, n]G$ be the TCG from X_i to X_j. From the known global solutions we know: $m \leq \lceil s_j \rceil^G - \lceil s_i \rceil^G \leq n$ and $m \leq \lceil t_j \rceil^G - \lceil t_i \rceil^G \leq n$. Hence, if $min(s_j, t_j) = s_j$ and $min(s_i, t_i) = s_i$ the TCG is satisfied by our candidate solution; the same if $min(s_j, t_j) = t_j$ and $min(s_i, t_i) = t_i$. Consider now the case with $min(s_j, t_j) = s_j$ and $min(s_i, t_i) = t_i$. Substituting t_i to s_i in the first inequation above, we obtain $m \leq \lceil s_j \rceil^G - \lceil t_i \rceil^G$, and by substituting s_j to t_j in the second we obtain $\lceil s_j \rceil^G - \lceil t_i \rceil^G \leq n$. Hence, also in this case, the TCG is satisfied by our candidate solution. The remaining case with $min(s_j, t_j) = t_j$ and $min(s_i, t_i) = s_i$ is analogous, and this reasoning applies to all TCGs on that arc. The proof for the other candidate partial solution is analogous when considering the global solution $X_i = max(s_i, t_i)$. □

We now prove the theorem. We show that the algorithm extends the earliest restricted due-time schedule, received as input, to one having the most relaxed duration constraints, and that this extension preserves the properties of earliest and restricted due-time. Step 1 repetitively calls the consistency algorithm ensuring that for each $i = 1 \ldots n$ any assignment with $(x_i, x_i + n_i)$ together with $(x_j, x_j + m_j)$ for all $j \neq i$ for beginning and ending nodes of the activities to be scheduled, extends to a global solution, when $m_i \leq n_i < newn_i$, and $min\, D_i \leq \lceil x_i + n_i \rceil^{G_i} - \lceil x_i \rceil^{G_i} \leq newmax\, D_i$. Indeed, the consistency is checked for each value n_i satisfying those constraints, skipping any n_i for which $\lceil x_i + n_i \rceil^{G_i}$ is undefined. Then, by Lemma 4, any assignment obtained combining $(x_i, x_i + n_i)$ for each $i = 1 \ldots n$ with n_i satisfying the above conditions does extend to a global solution. Then, the algorithm's output satisfies the schedule definition. Moreover, this schedule satisfies the definition of restricted due-time: condition (i) is satisfied, since, by Theorem 3, the input schedule is the earliest restricted due-time, and this implies $x_i + m_i = Min\{y \mid \lceil y \rceil^{G_i} - \lceil x_i \rceil^{G_i} = min\, D_i \text{ and } y > x_i\} = y_1$. Moreover, by Step 2 of the algorithm, $y_2 = x_i + newn_i$, and the ending interval is exactly $[x_i + m_i, x_i + newn_i]$. For condition (ii), note that, by the algorithm, $newmax\, D_i = \lceil x_i + newn_i \rceil^{G_i} - \lceil x_i \rceil^{G_i}$, and, hence,

$Min(x_i + newn_i, Max\{y \mid \lceil y \rceil^{G_i} - \lceil x_i \rceil^{G_i} = newmax\, D_i\} = x_i + newn_i$ for each $i = 1 \ldots n$. Similarly, $Max(x_i + m_i, Min\{y \mid \lceil y \rceil^{G_i} - \lceil x_i \rceil^{G_i} = min\, D_i\}) = x_i + m_i$, since $x_i + m_i = Min\{y \mid \lceil y \rceil^{G_i} - \lceil x_i \rceil^{G_i} = min\, D_i$ and $y > x_i\}$. Then, condition (ii) is satisfied by the output schedule.

Suppose now that S is the output schedule, and that there exists another earliest restricted due-time schedule S', such that, for activity A_i, it allows a pair (x, y) for beginning and end, with duration $d = y - x$ which is not allowed in S. Hence, either $d > newn_i$ or $m_i \leq d \leq newn_i$ but $min\, D_i \leq \lceil x_i + d \rceil^{G_i} - \lceil x_i \rceil^{G_i} \leq newmax\, D_i$ is violated. It is easily seen that the only possibility for this to be violated is that $\lceil x_i + d \rceil^{G_i}$ is undefined. However, in this case we have a contradiction, since the distance would not satisfy $TCG(Ab_i, Ae_i)$ in N while S' was assumed to be a schedule. We are left with the case of $d > newn_i$. Consider d' as the smallest duration with $newn_i < d' \leq d$ such that $\lceil x_i + d' \rceil^{G_i}$ is defined. We claim that d' must also be an allowed duration in S'. Indeed, consider condition (i) in the restricted due-time schedule definition. Since S' is an earliest schedule, it has x_i in its beginning interval. Then, for this x_i and S', we have $y_1 = x_i + m_i$ and $y_2 \geq x_i + d$, and consequently $y_1 \leq x_i + d' \leq y_2$. Since $\lceil x_i + d' \rceil^{G_i}$ is defined, by (i), $x_i + d'$ must be in the ending interval of S'. Now, since S' is a schedule, and it is an earliest one, the assignment $(x_j, x_j + m_j)$ for all $j \neq i$ and $(x_i, x_i + d')$ must be part of a global solution. This leads to a contradiction, since this assignment was checked for consistency by the algorithm deriving S, and rejected. Indeed, d' was assigned to n_i by the `While` statement in Step 1, as the next value, contained in a granule of G_i, after $newn_i$. Since $newn_i$ is the maximum in the output of S, the condition in the `Until` statement became true: either the network was inconsistent for that assignment or $x_i + d' > Max\{y \mid \lceil y \rceil^{G_i} - \lceil x_i \rceil^{G_i} = max\, D_i\}$, but this last condition is excluded since, in S', $Lend > x_i + d'$, and, by condition (ii) of restricted due-time schedule, cannot be greater than that $Max(\ )$ expression.

A.9. Proof of Theorem 5

The proof requires an extension of Lemma 4.

Lemma 5. *Let N be a TCG network with nodes $X_1, \ldots X_n$, $t_i < s_i$ for $i = 1, \ldots, k$ be a set of values, and $\{(a_1, \ldots, a_k) \mid \exists i (a_i = s_i$ and $a_j = t_j$ for each $j \neq i\}$ be a set of partial solutions, each of which extends to a global solution. Then $(s_1, \ldots, s_k)$ is a partial solution that extends to a global solution for N.*

Proof: We claim that $(s_1, \ldots, s_m, t_{m+1}, \ldots, t_k)$ is a partial solution that extends to a global one for each $m > 0$. We prove this by induction on m. It is obvious if $m = 1$ since $(s_1, t_2, \ldots, t_k)$ is in the given set. The induction hypothesis says that $(s_1, \ldots, s_m, t_{m+1}, \ldots, t_k)$ is a partial solution that extends to a global one. Consider $(s_1, \ldots, s_{m+1}, t_{m+2}, \ldots, t_k)$. Clearly, $(t_1, \ldots, t_m, s_{m+1}, t_{m+2}, \ldots, t_k)$ is a partial solution in the given set. Since $s_i > t_i$ for each i, we know by Lemma 4 that $(s_1, \ldots, s_{m+1}, t_{m+2}, ..., t_k)$ is a partial solution that extends to a global one. □

We now prove the theorem. We first show that the output is indeed a schedule. Step 1 repetitively calls the Due-time Schedule Derivation algorithm and the Maximal Durations

Due-time algorithm ensuring that for each $i = 1 \ldots n$ and for each x with $x_i \leq s_i \leq Min(Dom^i(Ab_i)) - 1$ the assignment with $(s_i, s_i + k_i)$ together with $(x_j, x_j + m_j)$ for all $j \neq i$ for beginning and ending nodes of the activities to be scheduled, extends to a global solution, when the pair $(s_i, s_i + k_i)$ satisfies Dur_i. Then, by Lemma 5, any assignment obtained combining $(s_j, s_j + k_j)$ for each $j = 1 \ldots n$ with s_j and k_j satisfying the above conditions does extend to a global solution. Then, the algorithm's output satisfies the schedule definition. It is easily seen that the output schedule is a restricted due-time schedule and one with maximal durations.

Suppose now that the output schedule S is not the maximal one, i.e., there exists an earliest restricted due-time schedule $S' \neq S$ with the same duration constraints such that $S \preceq S'$. Since durations are the same, and, by being restricted due-time schedules, the ending intervals are determined by the beginning intervals and durations, it must be that for some i with $1 \leq i \leq n$ there exists x' in the beginning interval of A_i in S' such that $x' > Lbegin_i$ where $Lbegin_i$ is the maximum value in the corresponding interval in S. Since any interval is a set of contiguous values, the existence of x' implies that $Lbegin_i + 1$ is also in the beginning interval of A_i in S'. Hence, each of the assignments $(Lbegin_i + 1, Lbegin_i + 1 + k)$ and $(x_j, x_j + m_j)$ for all $j \neq i$, for any k satisfying Dur_i, will extend to a global solution. This contradicts the correctness of the algorithms applied in Step 1.1, since our algorithm performed an iteration of that step with $Dom^i(Ab_i) = \{Lbegin_i + 1\}$, and $Dom^i(Ab_j) = \{x_j\}$ and $Dom^i(Ae_j) = \{x_j + m_j\}$ for all $j \neq i$. Since $Lbegin_i$ is the value in the output of the algorithm, it means the condition in the `until` statement was satisfied for that iteration, i.e., either the assignment does not lead to a solution no matter what k is used (N^i is inconsistent), or there is some value of k among those allowed by Dur_i which does not lead to a solution.

Acknowledgments

This work was partially supported by the Army Research Office under the grant DAAG-55-98-1-0302. The work of Wang was also supported by an NSF Career Award under the grant 9875114.

Notes

1. Exception handling is necessary when constraints are violated.
2. When the interval is not followed by a time granularity, it indicates by default that the bounds are given in terms of a "base" granularity, which provides a smallest "unit" of time measurement in the workflow, e.g., hours.
3. If *Dom* is not explicitly shown for a node w in a network, it is assumed that $Dom(w)$ is $[1, +\infty]$.
4. Time $O(n^3)$ where n is the number of nodes in the network.
5. A distance graph has the same nodes as the network, but each arc in the network from node i to node j labeled with an interval [m, n] is mapped in the distance graph to an arc from i to j labeled "m" and an arc from j to i labeled "$-n$".
6. This means that each value in the beginning (ending, resp.) interval allowed by S' is also allowed by S in the beginning (ending, resp.) interval for the same activity, and that any duration allowed by S' for that activity is also allowed by S.
7. For the sake of clarity, we do not depict the implicit constraints originating from the first and last node since they won't be used in the algorithms.

8. This base time granularity is the default one when a TCG is not explicitly attached with a time granularity, as assumed in Section 2.
9. Due to technicalities in the TCG consistency algorithm, the maximum is infinite whenever the one in the output network is equal to a constant MAX. See [7].
10. Otherwise, we move $Ebegin_i$ forward (or $Lbegin_i$ backward) to make these defined.
11. This holds since the schedules are not forced to be well-formed.
12. This granularity is easily defined, since it has the same R and T of G_i and sets of values $S_0 \ldots S_{R-1}$ each one equal to the first value of the corresponding one in G_i.
13. G is a subgranularity of H if for each index i, there exists an index j such that $G(i) = H(j)$.
14. We conjecture that the problem is NP-hard even with a small fixed set of granularities such as `day` and `week`, or `day` and `business-day`.

References

1. K.R. Abbott and S.K. Sarin, "Experiences with workflow management: Issues for the next generation," in Proc. of ACM Conference on Computer-Supported Cooperative Work, 1994, pp. 113–120.
2. G. Alonso, D. Agrawal, A. El Abbadi, and C. Mohan, "Functionalities and limitations of current workflow management systems," Research Report, IBM Almaden Research Center, 1997.
3. C. Bettini and R. De Sibi, "Symbolic representation of user-defined time granularities," Annals of Mathematics and Artificial Intelligence, vol. 30, no. 1–4, 2001.
4. C. Bettini, C.E. Dyreson, W.S. Evans, R.T. Snodgrass, and X.S. Wang, "A glossary of time granularity concepts," in Temporal Databases: Research and Practice, O. Etzion, S. Jajodia, and S. Sripada (Eds.), LNCS State-of-the-Art Survey, vol. 1399, Springer: Berlin, 1998, pp. 406–413.
5. C. Bettini, S. Jajodia, and X.S. Wang, Time Granularities in Databases, Data Mining, and Temporal Reasoning. Springer: Berlin, 2000.
6. C. Bettini, X. Wang, and S. Jajodia, "Satisfiability of quantitative temporal constraints with multiple granularities," in Proc. of 3rd Intern. Conf. on Principles and Practice of Constraint Programming, 1997. Springer-Verlag, LNCS, vol. 1330,
7. C. Bettini, X. Wang, and S. Jajodia, "A general framework for time granularity and its application to temporal reasoning," Annals of Mathematics and Artificial Intelligence, vol. 22, no. 1/2, pp. 29–58, 1998.
8. F. Casati, B. Pernici, G. Pozzi, G. Sanchez, and J. Vonk, "Conceptual workflow model," in Database Support for Workflow Management: The WIDE Project, Kluwer: Boston, MA, 1999.
9. T. Dean, "Artificial intelligence: Using temporal hierarchies to efficiently maintain large temporal databases," JACM, vol. 36, no. 4, 1989.
10. R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," Artificial Intelligence, vol. 49, 1991.
11. J. Eder, E. Panagos, and M. Rabinovich, "Time constraints in workflow systems," in Proc. of 11th Intl. Conference on Advanced Information Systems Engineering, Heidelberg, Germany, 1999.
12. I.J. Haimowitz, J. Farley, G.S. Fields, and J. Stillman, "Temporal reasoning for automated workflow in health care enterprises," in Lecture Notes in Computer Science, vol. 1028, 1996, pp. 87–113.
13. D. Hollingsworth, "The workflow reference model," Workflow Management Coalition, WFMC-TC-1003, 1994.
14. A.K. Mackworth and E.C. Freuder, "The complexity of some polynomial network consistency algorithms for constraint satisfaction problems," Artificial Intelligence, vol. 25, 1985.
15. O. Marjanovic and M.E. Orlowska, "On modeling and verification of temporal constraints in production workflows," Knowledge And Information Systems, vol. 1, no. 2, pp. 157–192, 1999.
16. P.Z. Revesz, "A closed form evaluation for Datalog queries with integer (gap)-order constraints," Theoretical Computer Science, vol. 116, no. 1, 1993.
17. R.L. Rivest, C.E. Leiserson, and T.H. Cormen, Introduction to Algorithms, MIT Press: Cambridge, MA, 1991.
18. V. Suresh and D. Chaudhuri, "Dynamic scheduling: A survey of research," International Journal of Production Economics, vol. 32, no. 1, 1993.
19. Workflow Management Coalition, "Terminology & glossary," Document number WFMC-TC-1011D. June 96. `http://www.aiai.ed.ac.uk/project/wfmc/DOCS/glossary /glossary.html`