Multimedia Systems 7: 424–437 (1999)

Multimedia Systems

Interactive authoring of multimedia documents in a constraint-based authoring system

Junehwa Song[1], **G. Ramalingam**[1], **Raymond Miller**[2], **Byoung-Kee Yi**[2]

[1] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA; e-mail: JUNESONG, RAMA@US.IBM.COM
[2] University of Maryland, College Park, MD 20742, USA; e-mail: MILLER, KEE@CS.UMD.EDU

Abstract. As multimedia applications spread widely, it is crucial for programming and design support systems to handle "time" in multimedia documents effectively and flexibly. This paper presents a set of interactive system support tools for designing and maintaining the temporal behavior of multimedia documents. The tool set provides mechanisms for anomaly detection, temporal query processing, and interactive scheduling. It is based on a fast incremental constraint solver we have developed, which can be adapted by any constraint-based system. The incremental constraint solver provides immediate feedback to the user, supporting a highly interactive design process. Combined with existing optimal layout generation mechanisms proposed in the literature, our tools effectively utilize the flexibility provided by constraint-based systems.

Key words: Multimedia documents – Multimedia authoring – Interactive systems – Temporal constraint systems – Incremental algorithms

1 Introduction

A multimedia document involves both temporal and spatial integration of media items of different types [1, 6, 8, 14, 15, 20, 23, 24]. An interactive multimedia document also needs to support user interactions. Authoring a multimedia document where time, space, and user interactions interplay is an involved design process. The way the document is designed will greatly influence the effectiveness and quality of the presentation. To author a creative document, authors will need effective design support tools. A good design support tool will assist the authors to focus on the design goals, facilitating the iteration of the design process. Sufficient system support for an effective dialog between the designer and the system is critical for this purpose. The dialog must be direct and immediate. Also, a fine-grained control over the process is necessary for the designer to fully articulate his/her vision.

Correspondence to: J. Song

In this paper, we describe a set of design support tools for effectively designing and maintaining "time" in multimedia documents. Constraint-based management of time in multimedia documents proposed in the literature has a clear advantage over the traditional timeline approach due to its flexibility. In a constraint-based system, users describe their design choices declaratively through relationships among media objects [12, 13]. Details of satisfying and maintaining the requirements can be supported by the system's automatic layout generation mechanisms. Buchanan and Zellweger [4] reported a mechanism for automatically obtaining an optimal temporal layout using a linear programming technique. Hamakawa and Rekimoto [10] utilized the notion of temporal glue and box, along with the constraints to specify relationships between events. In [14], we described a minimal-cost fair-scheduling algorithm for the elastic time model. The focus of earlier research efforts has been on the development of automatic layout generation mechanisms.

Our aim is to equip a constraint-based authoring system with interactive control. Despite various advantages a constraint-based authoring system offers, its practical utility has been limited, because it has failed to provide a mechanism for users to easily manipulate the underlying structures. Consider the traditional timeline approach, where the representation is fixed and the operations are manual. In the timeline approach, users can have a direct view and control of the structure of the document. But in a constraint-based system, although flexibility exists, the understanding of the document's state and the fine-grained control can easily be lost. For a constraint-based system to be of practical use in the creative authoring process, more elaborate system support tools are required which enable effective communication between the designer and the document system.

A constraint-based authoring system has to satisfy three basic requirements. Firstly, the system should provide ways to help the users understand the internal state of documents. In a constraint-based system, while the author can give a high-level design description, the system's internal representation tends to be complicated, making it difficult to abstract and visualize the internal system state. This results in the separation of the user's view and system's view, and adversely affects the design process. Secondly, the system

should provide means to help users to easily handle anomalies in the design. As the users rely on the system's automatic layout mechanism, and the specification of the document becomes more complex, users become careless, thus making the design process more error-prone. As parts of the design become erroneous, an automatic layout mechanism may not be able to support the users. Worse yet, the separation of the user's view from the system's view makes it more difficult for users themselves to locate and correct erroneous parts once they are introduced. Thirdly, the system should provide ways for the designers to effectively reflect their changing design choices into the final layout and refine the design at each step. While the automatic layout mechanism simplifies the selection of the final layout, users may lose fine-grained control over that final layout. The value of an automatic layout mechanism will be maximized when it is combined with effective ways to explore alternatives provided by the design.

The importance of the above-listed requirements will increase as technologies evolve, and the demands on the system design grow in sophistication and complexity. To meet the requirements, we developed a set of interactive system support tools: anomaly handling, temporal query processing, and interactive scheduling. These system support tools are based on an incremental constraint solver, which allows immediate communication between the designers and systems.

We give our presentation in the framework of the Isis authoring system and elastic time model, described in [14]. We believe that the solutions and tools we provide are sufficiently general to be shared by any type of constraint-based authoring system. In our system, users directly manipulate the timeboxes and explicitly control the temporal behavior of the document. The system interprets the user actions and maintains the document state incrementally, and feedback from the system occurs immediately based on the fast incremental constraint solver we have developed. Upon each user interaction, the system immediately checks its validity. If valid, the system immediately gives a feasible schedule, and if it is not valid, the causes of the anomaly are reported to the users. Therefore, users do not need to worry about the correctness of the document, and are able to concentrate on their design work. In addition, users are able to query a valid range of events and the document state through temporal query processing. This overcomes the difficulty of effective abstraction of the document's internal state, and helps the users understand the current state of the document allowing them to explore a possible design space through many stages of the authoring process.

Providing an efficient incremental constraint solver which can handle nonmonotonic interactions (additions, deletions, and modifications) is not a trivial problem [9, 11, 12]. This is especially true for general linear constraints. CLP(R) [12] includes an incremental version of the Simplex algorithm. DeltaBlue [9] is an incremental constraint solver for an acyclic constraint hierarchy. Our approach is to concentrate on a restricted class of linear constraints, called difference constraints. This type of constraint has been commonly used to express temporal behavior [4, 5, 13, 14, 16, 25]. We also handle the optimal scheduling problem separately from other required system support mechanisms. By focusing on difference constraints, we have developed a very efficient incremental constraint solver, which works asymptotically faster than well-known batch-mode algorithms [5, 16]. We believe that our work is the first to present an incremental algorithm for solving difference constraints.

This paper is organized as follows. In Sect. 2, we provide a brief overview of the elastic time model and define a set of user interaction primitives. In Sect. 3, a mechanism for translating the user interaction primitives into a temporal constraint system is shown. Section 4 presents the incremental constraint solver. Section 5 describes the system support facilities for interactive authoring: anomaly handling, temporal query processing, and the interactive scheduling mechanism. We also provide an example that illustrates how the system aids the authoring process, along with a few snapshots of our system in the section. Other related issues are discussed in Sect. 6 and Sect. 7 concludes the paper. A more complete description of our model of multimedia documents, including the mechanisms for treating the spatial layout and asynchronous user events, can be found in [13, 19].

2 Elastic time model

Producing a multimedia document is a highly iterative design process. Authors will want to carefully reflect their design choices through repeated composition and validation cycles. Even with a finalized document, the authors might want to reflect slightly different design choices as their presentation environments dictate.

Our elastic time model provides a framework for dealing with the temporal behavior of the document flexibly. Using the metaphor of a spring system, the elastic time model allows users to associate with each media object a minimum and a maximum length and a length at rest (i.e., the most desirable play length). Users can connect the (elastic) objects by defining temporal relationships among them. In doing so, users can avoid the tedious positioning and repositioning of media objects in time in composing or modifying a document. Furthermore, once constructed, a document is associated with a range of possible layouts, so that the users can choose an appropriate one according to their design choices and presentation environments.

2.1 Temporal specification

We base our specification language on Allen's temporal algebra [3], and take time intervals as primitives. Let a multimedia object m be associated with a triple of lengths, which we call spring constants,

$$(\alpha_m, \lambda_m, \omega_m), \quad \text{where } \alpha_m \leq \lambda_m \leq \omega_m,$$

such that m may be presented over a time interval I_m whose length falls in the range bounded by a minimum α_m and a maximum ω_m, and λ_m is the most desirable, or optimal length.

The following two primitive relations can hold between two objects:

– $co\text{-}start(m_1, m_2)$: time intervals I_{m_1} and I_{m_2} share the same beginnings,

– *co-end*(m_1, m_2): time intervals I_{m_1} and I_{m_2} share the same ends.

Note that m_i can be a "dummy" object such that it has nothing to show or play, but it may be associated with a range of intervals of nonnegative lengths. A dummy object can be used to cause a time delay. With this, relationships such as $meet$ or $overlap$ can also be expressed using the two primitives.

For the ease of use, we also include following two more relations, in addition to the above two primitive relations:

– $meet(m_1, m_2)$: the end of I_{m_1} is shared by I_{m_2} as its beginning,[1]
– *co-occur*(m_1, m_2) : time intervals I_{m_1} and I_{m_2} share the same beginnings and endings.[2]

2.2 The time-box representation

We represent multimedia objects visually using a simple building block metaphor. Multimedia objects are treated as electronic building blocks, or timeboxes, such that the lengths of the timeboxes are proportional to the (optimal) lengths of the corresponding objects. See Fig. 1 for a graphical illustration of the four temporal relations. Two short video objects, "sneeze" and "cough", are represented by two elastic time-boxes. They can start together (*co-start*: indicated by connecting their left ends with a bracket); end together (*co-end*: indicated by connecting their right ends); start together and end together (*co-occur*: indicated by connecting both ends), or one can follow the other (*meet*). In the figure, as "sneeze" is shorter than "cough" and is stretchable, the *co-occur* resulted in stretching the "sneeze" to the same length as the "cough".

2.3 User interaction primitives for composition

Users directly manipulate the timeboxes to specify the temporal behavior of documents: they can drag and drop timeboxes, resize, connect, separate, or remove them, using the system provided graphical user interfaces. We summarize below the set of user interaction primitives we use for designing multimedia documents interactively. This set provides sufficient capability to handle the spring constants of media objects and temporal relationships among them in a document.

– $+timeBox(t_i)$: to add a time-Box t_i.
– $-timeBox(t_i)$: to remove t_i,
– $+relation(r, t_i, t_j)$: to add r between t_i and t_j, $r \in \{meet, co\text{-}start, co\text{-}end, co\text{-}occur\}$.
– $-relation(r, t_i, t_j)$: to remove r between t_i and t_j.
– $minTime(t_i, l)$: to set the minimum length of t_i to a length l.
– $maxTime(t_i, l)$: to set the maximum length of t_i to l.
– $optTime(t_i, l)$: to set the optimal length of t_i to l.

[1] $meet(m_1, m_2)$ can be expressed by *co-end*(m_1, d), followed by *co-start*(d, m_2), where d is a dummy object with zero length.

[2] *co-occur*(m_1, m_2) is equivalent to *co-start*(m_1, m_2), followed by *co-end*(m_1, m_2).

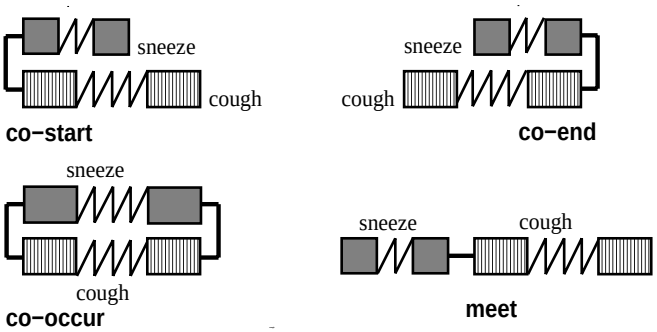


Fig. 1. Four temporal relations for building elastic stories

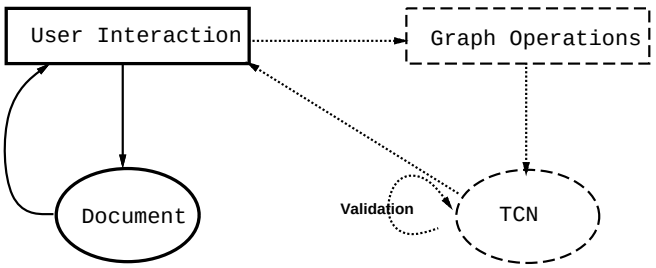


Fig. 2. Interaction loop for construction of documents: *solid arrows* show the outlook of interactions and *dotted arrows* show the internal flow of operations in the system

Note that, in the case of $meet$ relationship, the order of parameters in $+relation(r, t_i, t_j)$ and $-relation(r, t_i, t_j)$ needs to be considered. For example, $+relation(meet, t_i, t_j)$ means adding $meet(t_i, t_j)$, which is different from $meet(t_j, t_i)$.

3 Construction of the temporal constraint network

The authoring process consists of a sequence of user actions for constructing multimedia documents. In interactive authoring systems, each user interaction should be processed immediately with appropriate feedback.

Figure 2 shows the interaction loop between the user and the document during the authoring process. In our system, the state of a document is maintained in a *temporal constraint system*, which is represented by a directed, weighted graph called *temporal constraint network*, which we call *TCN* hereafter. The system maintains the TCN incrementally. It translates each user interaction into a sequence of low-level graph operations, which we will describe shortly. Each low-level operation is applied to the TCN one by one, immediately updating the state of the document, using the incremental algorithm presented in Sect. 4. The user then proceeds with the next interaction. Should there be an inconsistency, it is quickly analyzed and reported to the user along with possible ways of resolving the problem. In this section, we describe how user interaction primitives are mapped to the graph operations.

Let $S = (M, R)$ denote a multimedia document, where M is a set of media objects and R is a set of temporal relationships between media objects in M. Let $m_i.s$ and $m_i.e$ be two variables which denote the start time and end time of an object $m_i \in M$, respectively, and $T = \bigcup_{m_i \in M} \{m_i.s, m_i.e\}$.

We can represent the temporal property of the document S by a set of constraints $C(S)$ over the variable set T. This set of constraints $C(S)$ includes the following constraints. For each $m_i \in M$, there is a constraint,

$$0 \leq \alpha_{m_i} \leq m_i.e - m_i.s \leq \omega_{m_i},$$

where $\alpha_{m_i}, \omega_{m_i}$ are a minimum and a maximum length of m_i, respectively. Each temporal relationship $r(m_i, m_j) \in R$ also induces constraints. For instance,

$$meet(m_1, m_2) \Longrightarrow m_1.e = m_2.s \Longleftrightarrow m_1.e - m_2.s \leq 0 \quad \text{and} \quad m_2.s - m_1.e \leq 0.$$

We call such a set of constraints $C(S)$ a "constraint set of a document S". Observe that every constraint in $C(S)$ can be expressed in the form $x - y \leq c$. This type of linear constraint, involving the difference between two variables, is called a *difference constraint*.

The constraint set of a document S, i.e., $C(S)$, is internally represented by a directed, weighted graph, called TCN, as previously explained, which is denoted by

$$TCN(S) = \langle V, E, w \rangle,$$

where V is a set of nodes, E is a set of directed arcs, and w is a mapping from E to real numbers which defines the weights of the directed arcs in E, each of which is defined as follows:

$$V = \{n(x_i) \mid x_i \in T\} \bigcup \{source\},$$
$$E = \{(n(x_i), n(x_j)) \mid x_j - x_i \leq c \in C(S)\} \bigcup \{(source, n(x_i)) \mid x_i \in T\},$$
$$w(n(x_i), n(x_j)) = c, \text{ if } x_j - x_i \leq c \in C(S),$$
$$w(source, n(x_j)) = 0.$$

In the above definition, we include a node $n(x_i)$ for each variable x_i in T. For each constraint $x_j - x_i \leq c$ in $C(S)$, we include a directed arc $(n(x_i), n(x_j))$ in E with a weight c, i.e., $w(n(x_i), n(x_j)) = c$. We also include a special node $source$ in V and, for each node $n(x_i)$, $x_i \in T$, we include an arc $(source, n(x_i))$ in E, with $w(source, n(x_i)) = 0$.[3] In the rest of paper, we abbreviate $n(x_i)$ to x_i, for simplicity of notation. In other words, we use the same symbols for nodes in V and the corresponding variables in T.

A TCN, $TCN(S) = \langle V, E, w \rangle$, is associated with a set of graph operations:

- $+node(x_i)$: to add a node x_i to V and the special arc $(source, x_i)$ to E with $w(source, x_i) = 0$.
- $-node(x_i)$: to remove a node x_i from V and the special arc $(source, x_i)$ from E.
- $+arc(x_i, x_j, c)$: to add an arc (x_i, x_j) to E with the weight c, i.e., $w(x_i, x_j) = c$.
- $-arc(x_i, x_j)$: to remove an arc (x_i, x_j) from E[4]
- $weight(x_i, x_j, c)$: to modify the weight of the arc $(x_i, x_j) \in E$ to c, i.e., $w(x_i, x_j) = c$.

Each user interaction primitive for document composition is translated to a sequence of graph operations as summarized below. Note that initially a TCN consists only of a special node $source$.

- $+timeBox(t_i) \Longrightarrow +node(t_i.s)$; $+node(t_i.e)$; $+arc(t_i.s, t_i.e, \omega_{t_i})$; $+arc(t_i.e, t_i.s, -\alpha_{t_i})$.
- $+relation(meet, t_i, t_j) \Longrightarrow +arc(t_i.e, t_j.s, 0)$; $+arc(t_j.s, t_i.e, 0)$.
- $+relation(co\text{-}start, t_i, t_j) \Longrightarrow +arc(t_i.s, t_j.s, 0)$; $+arc(t_j.s, t_i.s, 0)$.
- $+relation(co\text{-}end, t_i, t_j) \Longrightarrow +arc(t_i.e, t_j.e, 0)$; $+arc(t_j.e, t_i.e, 0)$.
- $+relation(co\text{-}occur, t_i, t_j) \Longrightarrow +relation(co\text{-}start, t_i, t_j)$; $+relation(co\text{-}end, t_i, t_j)$.
- $minTime(t_i, l) \Longrightarrow weight(t_i.e, t_i.s, -l)$.
- $maxTime(t_i, l) \Longrightarrow weight(t_i.s, t_i.e, l)$.

Note that $-relation(r, t_i, t_j)$ is handled by removing all the arcs which have been added by the corresponding $+relation(r, t_i, t_j)$. The following provides an example for the $-relation$ operation:

$$-relation(meet, t_i, t_j) \Longrightarrow -arc(t_i.e, t_j.s); \; -arc(t_j.s, t_i.e).$$

For $-timeBox(t_i)$, all the arcs involving $t_i.s$ and $t_i.e$ as well as the two nodes are to be removed. As for the optimal length of each media object, it is not included in TCN, but separately handled by the minimum-cost fair scheduler. Therefore, $+optTime(t_i, l)$ is not translated to graph operations. [5]

Consider the small multimedia document shown in Fig. 3. A user has constructed this document with three media objects, $Running\ Dog$ (an animated drawing), $House$ (static drawing), and $Waltz$ (sound) and a $delay$ (dummy object) and four temporal relationships among them. The document shows a house in the background, and after a while, a dog appears in front of the house and runs and is accompanied by the waltz. Assume that the user continues by adding a new relation, $+relation(co\text{-}end, R, W)$, so that the $Running\ Dog$ stops running when $Waltz$ finishes.

The current document can be represented by $S^\circ = (M^\circ, R^\circ)$, where $M^\circ = \{R, H, W, D\}$ and $R^\circ = \{co\text{-}start(D, H), meet(D, R), co\text{-}start(W, R), co\text{-}end(H, W)\}$. The set of constraints for this document is

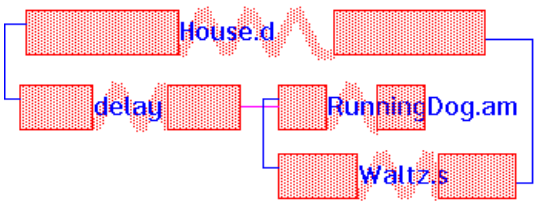
$$C(S^\circ) = \{D.s - D.e \leq -10, D.e - D.s \leq 15, \ldots, H.s - H.e \leq -25, H.e - H.s \leq 35\} \bigcup \{D.s - H.s \leq 0, H.s - D.s \leq 0, \ldots, W.e - H.e \leq 0, H.e - W.e \leq 0\}.$$

The TCN of the current document, $TCN(S^\circ)$ is graphically represented in Fig. 4. Upon user action $+relation(co\text{-}end, R, W)$, the system generates two graph operations

[3] This special node $source$ and special arcs $(source, n(x_i))$ are required by the constraint solver described in the next section. Intuitively, they ensure that every node in the TCN is reachable from $source$.

[4] We assume that there is only one arc between any given ordered pair of nodes to simplify the presentation of the incremental constraint solver in Sect. 4. The algorithms can be easily extended to handle the more general case in which multiple arcs are allowed between the same pair of nodes.

[5] Given a schedule of a document, the cost of a media object in the schedule is defined by the difference between the scheduled duration and its optimal duration. The minimum-cost fair schedule is the schedule which minimizes the sum of the costs as well as distributes the total cost evenly over the set of media objects in the document. Details can be found in [14].



a Timebox diagram

media object	type	min	max
Delay (D)	dummy	10	15
Waltz (W)	sound	13	15
Running Dog (R)	animation	5	8
House (H)	drawing	25	35

b Minimum and maximum play duration

Fig. 3a,b. Authoring with three media objects and a dummy object: media objects, *Running Dog (R), House (H), Waltz (W),* and a dummy object *Delay (D)* are related by $co\text{-}start(D, H)$, $meet(D, R)$, $co\text{-}start(W, R)$, and $co\text{-}end(W, H)$. Each object is represented by it's initial, i.e., D, W, R, H respectively, when this example is referred in the paper

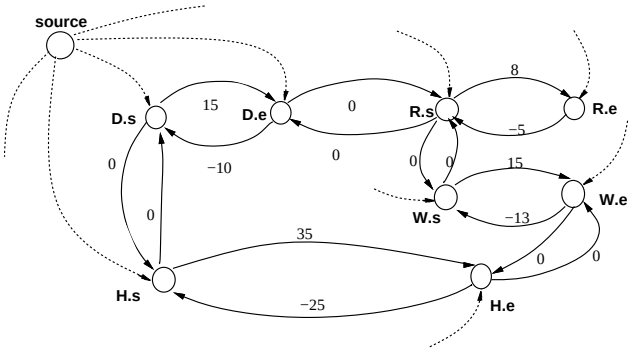


Fig. 4. Temporal constraint network for the document in Fig. 3. *Dotted arrows* represent the arc from source to each node with weight 0. Arcs with negative weights show minimum durations and those with positive weights show maximum durations

$+arc(W.e, R.e, 0)$; $+arc(R.e, W.e, 0)$,

and applies them to the current TCN. Assuming that the two graph operations do not cause any inconsistency, the TCN will be updated by including two additional arcs, $(R.e, W.e)$ and $(W.e, R.e)$, with $w(R.e, W.e) = 0$ and $w(W.e, R.e) = 0$, respectively.

4 Incremental constraint solver

Let $TCN(S) = \langle V, E, w\rangle$ be a TCN for a document S, and let $C(S)$ be the underlying system of difference constraints. An arc $(u, v) \in E$ with $w(u, v) = c$ represents a difference constraint:

$$v - u \leq c \in C(S).$$

Let $(x_1, x_2, ..., x_n)$ be a vector composed of the variables in T, i.e., $x_i \in T$, $1 \leq i \leq n$. The system $C(S)$ is said to be feasible, or satisfiable, if there exists an assignment of values to variables in T which simultaneously satisfies all the constraints in $C(S)$. We represent such an assignment by a vector $\vec{S}$, i.e., $\vec{S} = (a_1, a_2, ..., a_n)$, such that assignments, $x_i = a_i$, $1 \leq i \leq n$, makes $C(S)$ feasible, and call $\vec{S}$ a solution or a schedule. Given a $\vec{S}$, we represent an assignment to a specific variable x_j, $1 \leq j \leq n$ by $\vec{S}(x_j)$, i.e., $\vec{S}(x_j) = a_j$ when $\vec{S} = (a_1, a_2, ..., a_n)$.

The feasibility of $C(S)$ is equivalent to the absence of a negative cycle in $TCN(S)$. In addition, when the TCN does not include any negative cycle, the shortest distance from *source* to each node x_i $dist(source, x_i)$ comprises a solution for $C(S)$, i.e., $(dist(source, x_1), dist(source, x_2), \ldots, dist(source, x_n))$ is a solution vector for $C(S)$ [16, 18]. Therefore, in our algorithm, we test for feasibility by checking the existence of a negative cycle in $TCN(S)$. In general, a set of constraints, $C = \{c_1, c_2, \ldots\}$, which is infeasible, is called a minimal infeasible set, if removal of any $c_i \in C$ makes C feasible. That is, a minimal infeasible subset is a minimal unit of source of infeasibility. In our case, each minimal infeasible subset of $C(S)$ corresponds to a negative (simple) cycle in $TCN(S)$.

In this section, we provide an informal description of the algorithm which we developed to incrementally test the feasibility of the constraint system and provide a solution if the system is feasible. A more formal presentation of the algorithm with further improvements can be found in [17]. The algorithm processes the addition of a new arc and change of an arc weight in $O(n \log n + e)$ time, where n is the number of nodes in V and e is the number of arcs in E. It takes a constant time for other graph operations. This is faster than reevaluating a TCN by a batch-mode algorithm such as the one in [5], which takes O(n^3) time.

The idea behind the algorithm is as follows. Graph operations such as addition or deletion of a node, and deletion of an arc from TCN (or from any directed graph) cannot introduce any negative cycle as long as the current TCN (directed graph) does not include one. Therefore, the set of constraints remains feasible. Careful consideration is required only when adding a new arc to TCN or changing the weight of an arc. In these cases, we test the feasibility (absence of negative cycles) by applying Dijkstra's single-source shortest path algorithm [2, 21]. However, the problem is that Dijkstra's algorithm works only when there are no negative weighted arcs. To overcome this problem, we transform the weights of the arcs to nonnegative values using the currently available solution.

The algorithm assumes that the current TCN does not include any negative cycles and finds out if a change to the TCN will introduce one. In each step, it also maintains a solution vector $\vec{S}$ for the corresponding $C(S)$. We show in the following the implications of each graph operation on the feasibility of $C(S)$ and how the algorithm works in each case.

$+node(x_i)$: an addition of a node neither introduces a negative cycle, nor further constrains the system $C(S)$. It is an isolated node connected only to $source$. The algorithm simply adds the node to V, and an arc from $source$

to x_i with $w(source, x_i)$ = 0, and extends $\vec{S}$ by setting $\vec{S}(x_i)$ = 0.

$-node(x_i)$: a deletion of a node is processed only after all its incoming and outgoing arcs have been deleted, as will be discussed below. When all of its arcs are deleted, the node x_i is an isolated node. The node is simply removed from V, and from the solution vector $\vec{S}$.

$-arc(x_i, x_j)$: a deletion of an arc $(x_i, x_j) \in E$ does not introduce a negative cycle. The arc is simply removed from E. The solution vector $\vec{S}$ remains feasible, since the system $C(S)$ has become less constrained due to the deletion of the arc, or equivalently a constraint, $x_j - x_i \leq w(x_i, x_j)$.

$+arc(x_i, x_j, c)$: an addition of an arc potentially introduces a negative cycle. The procedure $AddArc$ in Sect. 4.1 handles this operation.

$weight(x_i, x_j, c)$: let $w(x_i, x_j) = c_1$ before the $weight$ operation. If $c_1 \leq c$, i.e., if the weight of an arc is increased, the system becomes no more constrained, and the $\vec{S}$ is still a solution. The algorithm will simply change the weight of the arc. If $c_1 > c$, i.e., if the weight of the arc is decreased, it is treated as a deletion of the arc followed by an insertion of a new arc with the corresponding weight.

As we have seen so far, the most difficult part in the algorithm is to handle the addition of a new arc to TCN.

4.1 Adding a new arc

Given a TCN(S) without any negative cycle and an operation $+arc(u, v, c)$, we need to check if the new arc, (u, v) with weight c introduces any negative cycle. This can be done by computing the shortest distance from node v to u and comparing it with c. To get the shortest distance, we use Dijkstra's single-source shortest path algorithm which runs in time $O(nlogn+e)$ [2]. The procedure $AddArc$ that handles $+arc(u, v, c)$ appears below.

Dijkstra's algorithm works only for directed graphs with nonnegative weights. But our TCN may contain negative-weight arcs as shown in Fig. 4. This problem can be solved by a weight transformation technique [7]. We explain how we adopt the technique to our problem in the Sect. 4.1.1.

Procedure $AddArc(TCN(S),\ (u, v),\ c,\ \vec{S})$
Parameters :
$TCN(S) = \langle V, E, w\rangle$
(u, v) : arc to be inserted to E
c : the weight of the arc (u, v)
$\vec{S}$: the feasible solution vector
begin
(1) for each $(x, y) \in E$,
$w_{transf}(x, y) = -\vec{S}(y) + w(x, y) + \vec{S}(x)$;
(2) compute $dist_{transf}(v, x)$, for each $x \in V$, using Dijkstra's algorithm and the new weight w_{transf};
(3) for each $x \in V$,
$dist(v, x) = \vec{S}(x) + dist_{transf}(v, x) - \vec{S}(v)$;
(4) if $c + dist(v, u) < 0$, then return (*not feasible*);
(5) else begin
(5.1) compute $dist_{transf}(source, x)$, for each $x \in V$, using Dijkstra's algorithm and the new weight w_{transf};
(5.2) for each $x \in V$, $dist(source, x) = \vec{S}(x) + dist_{transf}(source, x) - \vec{S}(source)$;
(5.3) for each $x \in V$, $\vec{S}_{updated}(x) = min(dist(source, x), dist(source, u) + c + dist(v, x))$;
(5.4) add (u, v) to E ;
(5.5) $w(u, v) = c$;
end ;
end

In the procedure, we first transform the weight of each arc, $(x, y) \in E$, by using the current solution vector $\vec{S}$ in Step 1. Thus transformed weight, $w_{transf}(x, y)$, is always nonnegative as shown in Sect. 4.1.1 and makes it possible to use Dijkstra's algorithm in later steps. In Step 2, we calculate $dist_{transf}(v, x)$, the shortest distance from v to each node $x \in V$ with the transformed weight, w_{transf}. From $dist_{transf}(v, x)$, the shortest distance $dist(v, x)$ with the original weight w is recovered in Step 3. In Step 4, we test if $dist(v, u) + c < 0$. If so, the new arc (u, v) forms a negative cycle and we reject the addition. Otherwise, we update the solution vector in Step 5.

This is done by computing the shortest distance from the $source$ node to each node x in the updated TCN (Step 5.3)[6]. It is the minimum of the shortest distance from $source$ to x in the old TCN (which is computed in Step 5.1 and Step 5.2) and the shortest distance from $source$ to node x through the newly added arc (u, v). This shortest distance in the new TCN forms a new solution.

Though the solution computed by the procedure $AddArc$ is the shortest distance in the TCN, subsequent deletion or modification operations to the TCN may change the shortest paths, but not the solution. That is why we need to compute $dist(source, x)$ in Step 5.1 and in Step 5.2, and cannot use the current solution $\vec{S}$ for it.

4.1.1 Weight transformation

As mentioned earlier, Dijkstra's algorithm works only on graphs with nonnegative weight arcs. We first summarize the arc weight transformation idea proposed in [7] for using Dijkstra's algorithm for graphs with negative arcs.

Let G = $\langle V, E, w\rangle$ be a directed, weighted graph. If we construct a new graph, G' = $\langle V, E, w_{transf}\rangle$, with

$$w_{transf}(x, y) = f(y) + w(x, y) - f(x),\ (x, y) \in E,$$

where f is any function from a node to a real value, then the shortest paths in graph G are also shortest path in G', and vice versa. Further, let P be a path from node x to node y, and $length_G(P)$ be the length of P in G. Then,

$$length_{G'}(P) = f(y) + length_G(P) - f(x).$$

Coming back to our problem, if we can find some function f that satisfies

$$f(y) + w(x, y) - f(x) \geq 0$$

[6] We can not directly use Dijkstra's algorithm to calculate the shortest distance on the new TCN. The problem here is that when the weight transformation is applied to the new arc (u, v), by $w_{transf}(u, v) = -\vec{S}(v) + w(u, v) + \vec{S}(u)$, $w_{transf}(u, v)$ is not guaranteed to be nonnegative.

for every arc $(x, y) \in E$, then Dijkstra's algorithm can be used to compute the shortest paths under the transformed weight, and the lengths of the shortest paths under the corresponding original weight can easily be recovered.

We define $f(x)$ to be $-\vec{S}(x)$, where $\vec{S}$ is the current feasible solution. That is, we define the new weights by

$$w_{transf}(x, y) = -\vec{S}(y) + w(x, y) + \vec{S}(x).$$

Since $\vec{S}$ is the feasible solution vector,

$$\vec{S}(y) - \vec{S}(x) \leq w(x, y), \quad \text{for each } (x, y) \in E.$$

Thus,

$$w_{transf}(x, y) = -\vec{S}(y) + w(x, y) + \vec{S}(x) \geq 0.$$

Hence, the new weight w_{transf} is nonnegative.

5 System support for interactive authoring

Although the elastic time model provides a foundation for temporal flexibility in the document, it has to be appropriately supported by the system to be useful in the authoring process. The main features of our system support facilities include: anomaly handling, temporal query processing, and interactive scheduling. We describe each of the features below and provide a simple example to demonstrate how they work cooperatively and support the authoring process. Along with the example, we provide a few snapshots of our system and explain related interfaces of the system. Our system support tools are based on the incremental constraint solver described in Sect. 4.

5.1 Anomaly handling

In general, one of the problems with declarative specifications is that it can be difficult to identify the causes of infeasibility. Infeasibility is not always generated by a single media object or a single temporal relationship. Rather, it is formed by multiple media objects through multiple temporal relationships imposed on them.

However, our approach of interactive authoring with immediate validation inherently alleviates the possibly complex problem of determining the causes of infeasibility. As the user interacts with the system via user interaction primitives, the system checks if the new state of the document is feasible. If the system detects infeasibility, the interaction is temporarily rejected and the system notifies the user that infeasibility has occurred. As such, a document under construction is always feasible. If a new interaction results in an infeasible system, we can safely assume that the new interaction is related to the cause of infeasibility.

Visualization of the incompatible components

To recover from the infeasibility, the user will have to change parts of the document, changing relationships, durations of timeboxes, adding or removing delay objects, and so on, keeping the design choices in mind. Knowing that the current user interaction has caused the infeasibility, user may take a simple solution to change the current interaction. Alternatively, the user may consider other options. The system highlights the parts of the document, which may, when changed, remove the cause of infeasibility, to give visual feedback to users.

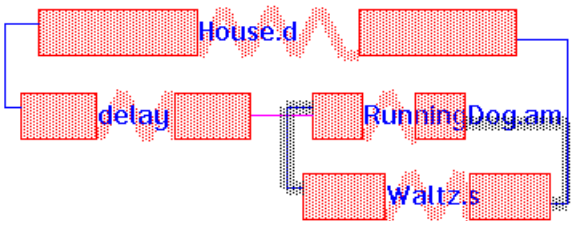


Fig. 5. Visualization of the incompatible components: media objects $Running\ Dog$ and $Waltz$ and the relationship $co\text{-}start(W, R)$ is not compatible with a new relationship $co\text{-}end(R, W)$

Let us consider the $TCN(S)$ to see how this can be done. As explained earlier, the minimal infeasible subsets of $C(S)$ are identified as the negative cycles in $TCN(S)$. Once the negative cycles are identified, they are mapped back to the corresponding media objects and temporal relationships which form them.

Consider the simple example in Figs. 3 and 4. Suppose again that the user wants to impose a new relationship $+relation(co\text{-}end, R, W)$ to have the $Running\ Dog$ stops running when $Waltz$ finishes. The system detects infeasibility as $+arc(R.e, W.e, 0)$ is applied to $TCN(S)$. The system identifies the negative cycle, $(W.e, W.s, R.s, R.e, W.e)$ and indicates that media objects R and W and their relationships $co\text{-}start(W, R)$, $co\text{-}end(R, W)$ are not compatible as in Fig. 5.[7] The user may choose to remove the relationship $co\text{-}start(W, R)$, which will remove the incompatibility. However, the document is still infeasible and the system identifies another negative cycle, $(W.e, H.e, H.s, D.s, D.e, R.s.R.e, W.e)$ and indicates that the entire document consisting of media objects W, H, D, R, and the three relationships, $co\text{-}start(D, H), meet(D, R), co\text{-}end(W, H)$, are incompatible with the new relationship $co\text{-}end(R, W)$.

System-suggested corrective actions

Users also may ask the system to suggest possible recovery actions. Upon such an user request, our system further analyzes the TCN, the current user action, and the graph operation which caused the infeasibility, and generates some possible recovery actions. Some examples are shown in Table 1.

To see how the suggestions are constructed, let us consider a set $C(S)$ and a $TCN(S)$ for a document S. Assume that $x - y \leq c$ is the constraint that creates infeasibility when added to $C(S)$, i.e., the arc (y, x) with $w(y, x) = c$ is to be added to $TCN(S)$, which creates one or more negative cycles. If we change $TCN(S)$ so that $dist(y, x) + dist(x, y) \geq 0$ (for example, by changing $w(y, x)$

[7] We say that a set of media objects and their relationships are not compatible (or incompatible) if it generates infeasibility in the underlying constraint system.

Table 1. Example: possible corrective actions, where $length(P)$ is the length of shortest path P with which a negative cycle has been detected in the procedure $AddArc$, and $delay$ is a timebox which represents a dummy, or delay object

User interaction	Graph operation	Possible corrective actions
$+relation(meet, t_i, t_j)$	$+arc(t_i.e, t_j.s, 0)$	$+timeBox(delay)$; $+relation(meet, t_i, delay)$; $+relation(meet, delay, t_j)$; $maxTime(delay, l)$, s.t. $l \geq -length(P)$
		$maxTime(t_i, \omega_{t_i} - l)$, s.t. $l \leq length(P)$
	$+arc(t_j.s, t_i.e, 0)$	$minTime(t_j, \alpha_{t_i} + l)$, s.t. $l \leq qlength(P)$
$+relation(co\text{-}start, t_i, t_j)$	$+arc(t_i.s, t_j.s, 0)$	$+timeBox(delay)$; $+relation(meet, delay, t_j)$; $maxTime(delay, l)$, s.t. $l \geq -length(P)$
		$maxTime(t_j, \omega_{t_j} + l)$, s.t. $l \geq -length(P)$
$+relation(co\text{-}end, t_i, t_j)$	$+arc(t_i.e, t_j.e, 0)$	$+timeBox(delay)$; $+relation(meet, t_i, delay)$; $maxTime(delay, l)$, s.t. $l \geq -length(P)$
		$maxTime(t_i, \omega_{t_i} + l)$, s.t. $l \geq -length(P)$

to c', such that $c' + dist(x, y) \geq 0$), all the negative cycles will become nonnegative. The system suggests sequences of user interaction primitives that achieve this effect.

Let us go back to the example in Figs. 3 and 4 and consider Fig. 5 for the incompatibility detected upon the interaction $+relation(co\text{-}end, R, W)$. When the user requests a recovery suggestion, the system can suggest that the user "increase the maximum duration of R ($Running\ Dog$) by at least 5" as shown in the last row of Table 1. Following the suggestion results in increasing $dist(W.e, R.e)$ by at least 5, thus making $dist(W.e, R.e) + dist(R.e, W.e) \geq 0$, and removes the incompatibility.

Note that it is hard for mechanically generated system messages such as those in Table 1 to match the user's design preference. One may consider that identifying the existence or cause of anomaly may be the best which a system can do. However, providing such suggestions which a user can optionally take will be useful as the documents become more complex and will make them feasible when followed.

5.2 Temporal query processing

As we have mentioned earlier, one of the advantages in authoring documents with the elastic time model is that the users do not have to go through the tedious computation of the time position of the media objects. However, users may still want to know certain temporal properties of the document at different stages of the authoring process.

As a document becomes flexible (with elastic time), inferring even most simple temporal properties becomes nontrivial. When each media object can shrink or stretch according to the elastic time model, the temporal distance between two events[8] in a document can vary within a range. Likewise, the total presentation time of the document also varies.

Our system guides the authoring decision by answering questions such as:

- What is the range of temporal distances between the events e_i and e_j in the current document?
- What is the range of durations of the current document?

The first query is represented by $Projection(e_i, e_j)$ and the second by $GlobalDuration$. $Projection(e_i, e_j)$ computes the minimum and the maximum distance between two temporal events e_i and e_j. The minimum distance is $-dist(e_j, e_i)$, and the maximum distance is $dist(e_i, e_j)$ in the corresponding TCN [5]. Both can be computed by Dijkstra's algorithm using the weight transformation technique explained in Sect. 4.1.1.

$GlobalDuration$ returns the temporal range of the document. Given a schedule (or a solution) $\vec{S}$ for a document S, we denote the (presentation) length of the document by

$$Duration(\vec{S}) = \max_i\{\vec{S}(i)\} - \min_i\{\vec{S}(i)\}.$$

Then, the $GlobalDuration$ can be defined by

$$GlobalDuration = [MinDuration, MaxDuration],$$

where $MinDuration = \min_{\vec{S}}\{Duration(\vec{S})\}$ and $MaxDuration = \max_{\vec{S}}\{Duration(\vec{S})\}$, where the minimum and maximum is computed over all feasible schedules $\vec{S}$.

A difficulty in computing the $GlobalDuration$ is that there can be infinitely many feasible schedules. However, a presentation duration is the temporal distance between the first event and the last event in a schedule. Thus, it is sufficient to consider only the distances between the (possible) first events and the (possible) last events. We define the sets F and L as follows:

$$F = \{x_i \mid \text{if } (x_i, x_j) \in E, \text{ then } w(x_i, x_j) \geq 0, x_i, x_j \in V\},$$
$$L = \{x_j \mid \text{if } (x_i, x_j) \in E, \text{ then } w(x_i, x_j) \geq 0, x_i, x_j \in V\}.$$

F is a set of nodes which are potential first events, and similarly L is a set of nodes which are potential last events. Then, we can show that

$$MinDuration = -\min_{x_i \in F}\{\min_{x_j \in L}\{dist(x_j, x_i)\}\},$$
$$MaxDuration = \max_{x_i \in F}\{\max_{x_j \in L}\{dist(x_i, x_j)\}\}.$$

That is, we can compute the $GlobalDuration$ by computing the shortest distances between the nodes in F and L. This can be done by using the Dijkstra's algorithm repeatedly or using an all-pairs-shortest-paths algorithm such as Floyd-Warshall's [5] just once.

[8] An event corresponds to a node in TCN. We use the terms $event$ and $node$ interchangeably.

Table 2. Example

a Media objects and spring constants.

name	type	min	opt	max
delay1	dummy	8	10	14
delay2	dummy	8	10	14
Narration1	sound	10	12	16
Narration2	sound	10	12	16
Medicine1	digital image	19	25	28
Medicine2	digital image	19	25	28
Music	sound	50	50	50

b Temporal relationships

$meet$	(*delay1*, *Narration1*)
$meet$	(*delay2*, *Narration2*)
$co\text{-}start$	(*delay1*, *Medicine1*)
$co\text{-}start$	(*delay2*, *Medicine2*)
$co\text{-}end$	(*Narration1*, *Medicine1*)
$co\text{-}end$	(*Narration2*, *Medicine2*)
$meet$	(*Medicine1*, *Medicine2*)
$co\text{-}start$	(*Medicine1*, *Music*)
$co\text{-}end$	(*Medicine2*, *Music*)

5.3 Interactive scheduling

If a document does not include any anomaly, the incremental constraint solver generates a feasible schedule immediately. This schedule does not consider the optimal durations of media objects and can be used when the user does not care about it. However, it can be powerfully used to interactively reflect the designer's choice to the final schedule, when combined with the other system support tools.

In many constraint-based authoring systems, the automatic schedulers mechanically optimize the schedule by globally negotiating the possible flexibilities allowed for each media item. This means that users lose direct control over the final schedule. Users may not be totally satisfied with the globally negotiated result. Since multimedia documents are designed for human perception, a small difference in the final schedule can result in a serious difference in the presentation quality. From our experience, users want to have fine-grained control over the various parts of the documents.

In our system, users can effectively reflect their design choices in the final schedule. Users repeatedly choose the exact durations (or tightly control the flexibility) of the desired media (or delay) objects. Upon each selection, they can immediately see the effect of their choice as the new schedule (which preserves the interactive selection previously made) is given immediately. This interactive scheduling occurs very smoothly, as the feasibility of the selection is checked right away. After interactive selection of layout of parts of the document, they may use the minimum-cost fair scheduler [14] to optimize the other parts of the document or just use the current nonoptimal schedule. This refinement may be continued further by alternatively using the optimizing scheduler and interactive selection.

5.4 Example

See Fig. 6 for the main view of our system. It also shows the timebox diagram of an example document. The document is composed of multimedia objects and temporal relationships shown in Table 2. The document is to advertise two drugs by a drug manufacturer. *Medicine1* and *Medicine2* are their two digital images, which will be shown one after the other. *Narration1* and *Narration2* are voice narratives for the two drugs, each of which will start playing shortly after its corresponding image appears on the screen. There is also a background music that plays for the duration of the document.

The white space in the screen, where the timebox diagram is located, is called "canvas". A user brings timeboxes of media objects onto canvas, and composes his/her document. To add a media object, the user clicks one of the buttons on the bottom of the screen. The system then opens up another window which lists available media objects of the designated type. When an object is selected from the list by a user, the system adds a timebox for the object on canvas. Similarly, by selecting the "stories" button on the bottom left corner, a user can bring a whole document which has already been constructed to the canvas and include it as a part of a new document.

Figures 7 and 8 show how timeboxes are manipulated on "canvas" to build the example document. In Fig. 7, a user selects a digital image object, $Medicine1$. Upon this, the system pops up an "action menu". From the "action menu", the user can select one of the four temporal relationships to be added. We can see, in Fig. 8, that the user has selected the "meet" relationship. After this, the system responds with a template timebox marked by "?", which will be replaced by the target timebox which the user clicks on. In Fig. 8, the user selects $Medicine2$ to relate it with $Medicine1$ by "meet". Similarly, to delete a temporal relationship, a user selects the button marked by "del-REL" from the action menu, and the system asks which relation to delete. The button "duration" is chosen to set the minimum, optimum, and/or maximum durations of a timebox. The interface for setting durations is shown in Fig. 9. The other two buttons on the "action menu" are to delete a timebox from canvas or to inspect properties of the associated media object.

Assume that a user continues in this fashion and constructs the document in Fig. 6. Note that no matter how simple or complex the document is, the document constructed in our system is always error free, since each step of construction has been checked. In the following, we show how the interactive scheduling method is used for already constructed documents without changing the content or the structure.

There are many possible temporal layouts (schedules) for the document. The user may want to select a schedule where each media object is presented for a duration as close as possible to the optimal duration. To do this, the user can ask for a minimal-cost fair schedule (by clicking the button marked "optimal" shown in Fig. 6), which is summarized in Table 3.

However, after some experience with the presentation of the document, the user may find that there is more interest in *Medicine1*, and that it requires more than 25 s to adequately describe the medicine. Suppose the user increases the presentation duration of *Medicine1* to its maximum, 28 s

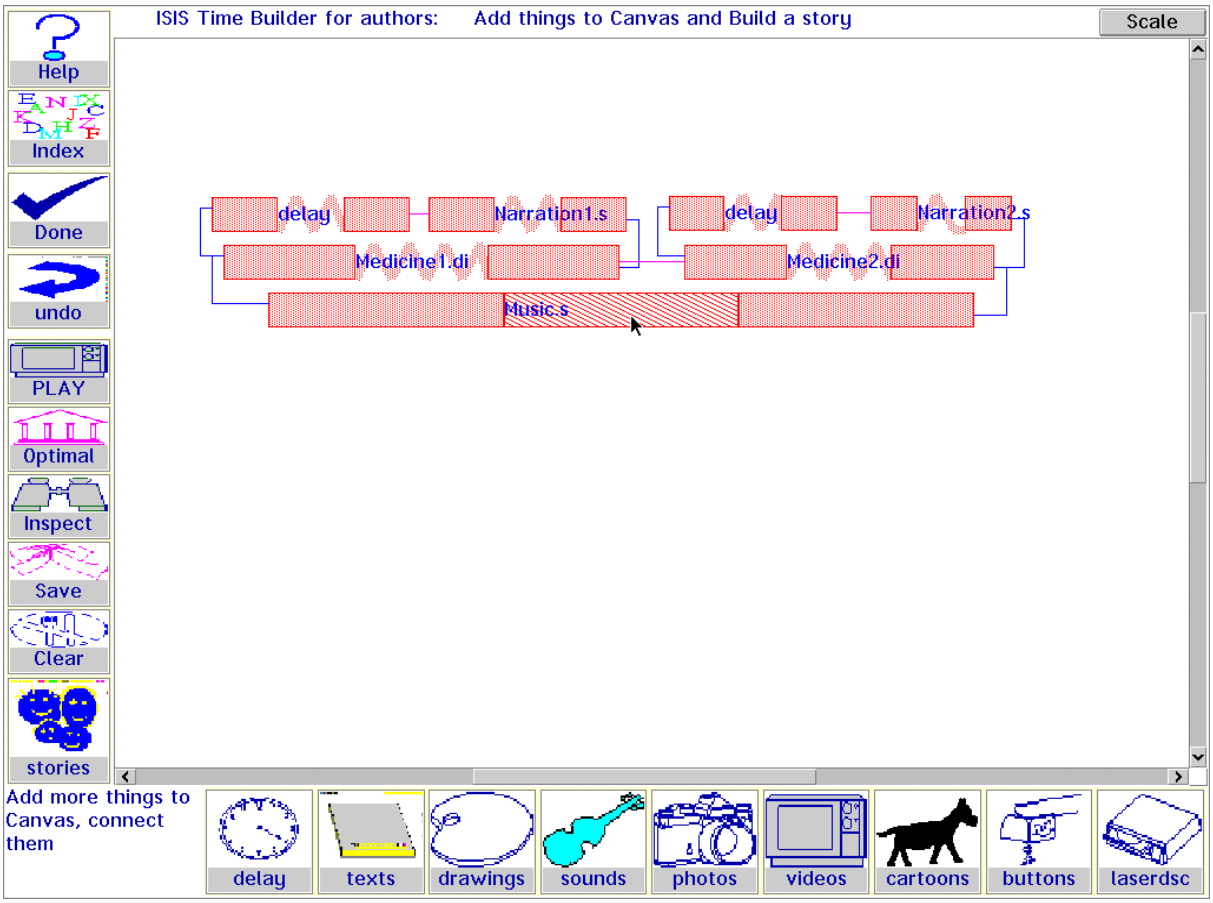


Fig. 6. Main view of system and the timebox diagram of an example document

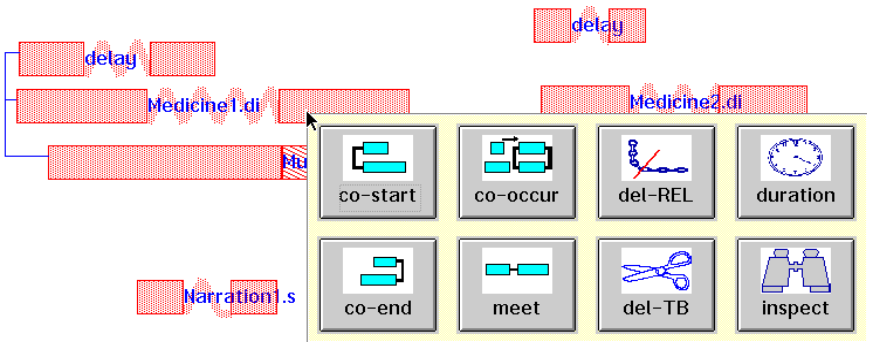


Fig. 7. Selecting a media object and an action from action-menu: a user selects digital image, $Medicine1$ and a $meet$ relation

Table 3. Minimal cost fair schedule

media object	scheduled duration
delay1	11.5
delay2	11.5
Narration1	13.5
Narration2	13.5
Medicine1	25
Medicine2	25
Music	50

(Fig. 9). The system will then compute another consistent schedule and return it to the user. At this point, the user may invoke the minimal-cost fair scheduler, which then will use the fixed duration of 28 s (as requested by the user) and optimize the remaining objects so that the total cost is minimal and fair. But suppose that the user tries to increase the duration further to 33 s. As shown in Fig. 10, the system immediately informs the user that this is not possible, and that if 33 s are needed, the objects (*Medicine1*, *delay1*, *Narration1*) or the relationships among them should be changed. The system also pops up a button in the upper right corner of canvas, which can be clicked to request the suggestions for correction.

At this point, the user may want to know what temporal range is possible for *Medicine1*, using the $Projection$ query. This is done by clicking the left rectangle of the timebox for $Medicine1$ (meaning the start of $Medicine1$) with right mouse button, dragging the mouse to the right rectangle (meaning the end of $Medicine1$), and releasing the mouse

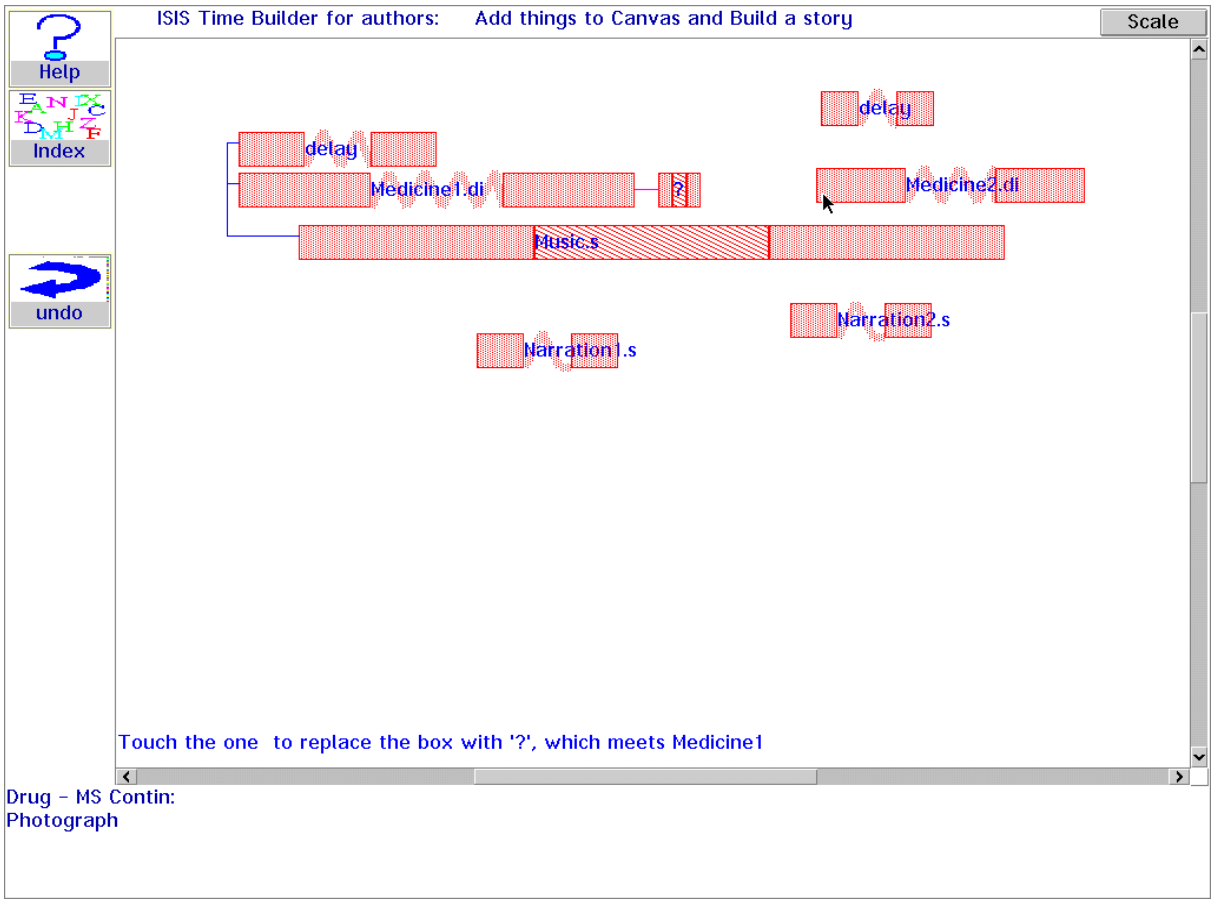


Fig. 8. Composition of the example document on "canvas": a user is imposing a "meet" relationship between Medicine1 and Medicine2

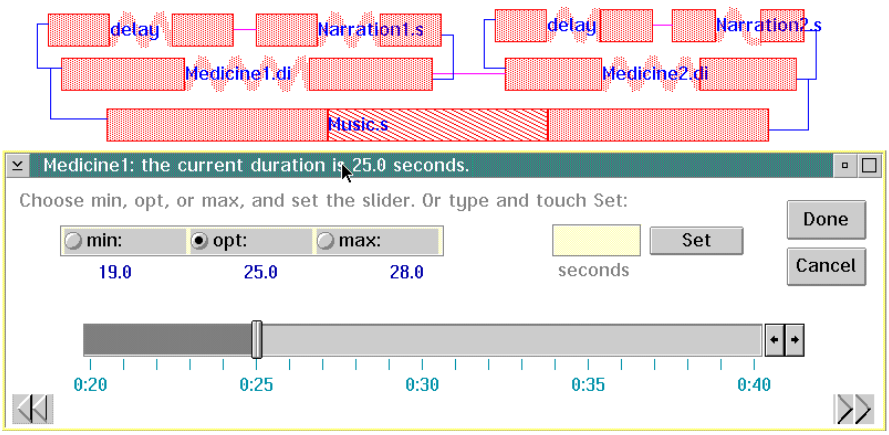


Fig. 9. Manipulation of durations of media objects

button on it.[9] As shown in Fig. 11, the system immediately informs that minimum and maximum durations between the two events are $(22, 30)$ s The users may then adjust the presentation duration of the image to 30 s, which will keep the document consistent. With this, the system computes a schedule with a fixed duration of 30 s for the digital image. The user may play back the document according to the schedule disregarding the optimal durations of the objects, or the user may use the minimum-cost fair scheduler. This interactive adjustment can be made for whatever objects the users want to control.

Our system support facilities provide users with a wide range of possibilities for combining their design choices and the system's automatic inferencing mechanisms effectively. Their value and utility will grow rapidly as the complexity of the document increases and the users' requirements grow in sophistication.

[9] To ask the duration of the whole document ($GlobalDuration$), a user clicks "Inspect" button shown in Fig. 6.

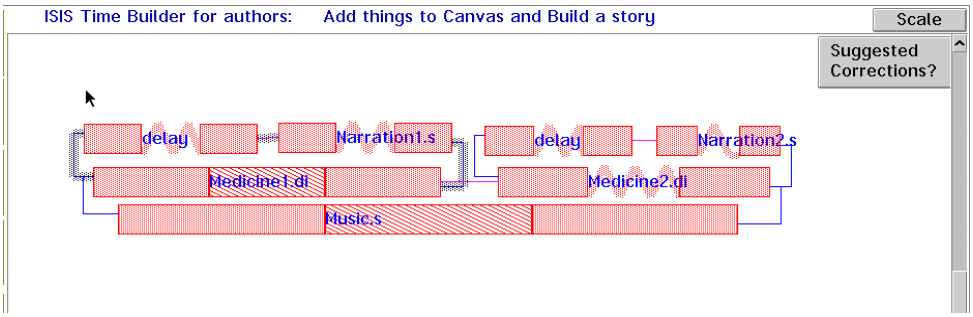


Fig. 10. Visualization of the incompatible components

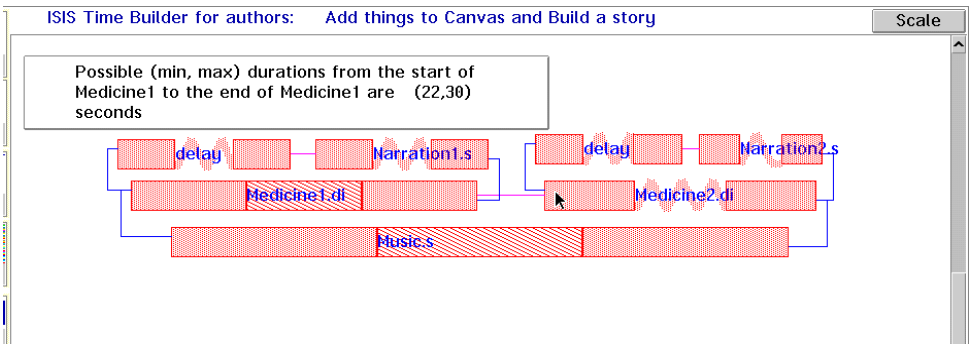


Fig. 11. Querying for the range of possible durations between two events

6 Discussion

The elastic time model and the interactive system support tools have been implemented as part of the Isis authoring environment at IBM T.J. Watson Research Center. Isis is a prototype multimedia authoring environment implemented in Smalltalk/OS2. In Sect. 5.4, we show a major component of Isis called "Time Builder". Other important components of Isis include Creator, Maker, Space Builder, Net Builder, Player, and Indexing System. Creator provides the author the capabilities for audio, video recording, capturing still images, drawing bitmaps, and text editing. Using the Maker, the author defines attributes of multimedia objects such as title, durations, and annotation for indexing. Space Builder is to design spatial property of documents. Net Builder links multimedia documents by adding hyperlinks among them, and Player provides a playback time environment. Lastly, Indexing System assists an author to create and use a structured index for archiving and retrieving multimedia information.

An earlier version of Isis provided the multimedia substrate for the Home Health-care system for Children with Leukemia [22], which IBM Research has jointly developed with New England Medical Center. It has also been used for building multimedia training systems to teach business decision-making process. From the experience with this earlier version of Isis, we observed that the underlying constraint system with automatic scheduling was very useful, especially at the early stage of the document construction. However, we also observed that users repeatedly come back to their document and elaborate it by adding or changing the contents and the structure. In this process, the document becomes larger and more complex, and users tend to lose their perception of the underlying document structure, while they want to have finer grained control. This experience motivated us to extend Isis and provide more elaborate interactive support tools described in this paper. Our internal experience with this new version so far has been successful in supporting the whole process of the document construction and iterative evolution. One thing to note is that in our experience users also tend to modularly design their document, i.e., they compose a document by piece, and later on include them as a part of one or more larger documents. Our incremental framework was very efficient in this respect, too. We believe that a more formal study, involving a human-computer interaction point of view, of the usability of the constraint-based authoring system with these interactive tools is an important research issue. We plan to conduct such a study in the future. However, that is beyond the scope of this paper,

Another important feature, which we have not addressed yet, in an interactive authoring system is "rollbacks"(or undos). Our interactive framework efficiently incorporates a rollback operation as follows. First, we can think of the reverse operation for each of the user interaction primitives described in Sect. 2.3. For instance,

$$+relation(r, t_i, t_j) \overset{reverse}{\Longleftrightarrow} -relation(r, t_i, t_j) \quad \text{and}$$
$$+timeBox(t_i) \overset{reverse}{\Longleftrightarrow} -timeBox(t_i).$$

For the other three primitives, to set minimum, optimum, or maximum duration, we need slight modifications in the representation to include one more parameter. For instance, $minTime(t_i, l)$ is changed to $minTime(t_i, l_1, l_2)$, which means that the minimum duration of t_i is changed from l_1 to l_2.[10] With this modification,

$$minTime(t_i, l_1, l_2) \overset{reverse}{\Longleftrightarrow} minTime(t_i, l_2, l_1).$$

We also keep track of the most recent user interaction, say A_1 (or most recent k interactions, $A_1, ..., A_k$, most recent first),

[10] This modification is only for the representation. In reality, the system remembers and appends the current duration.

as well as $\vec{S_1}$, the schedule (or solution vector) just before A_1 was applied (or $\vec{S_1}, ..., \vec{S_k}$). Upon rollback request, we first apply the reverse operation of A_1 to the current document (and therefore to the current TCN). This will restore the TCN to the previous state. However, note that the solution vector in the restored TCN might be different from $\vec{S_1}$, and thus we also reset the restored TCN's schedule to $\vec{S_1}$.

7 Conclusion

We have presented an incremental constraint solver and the interactive system support tools of our prototype authoring system, Isis. Our system allows users to directly control time in multimedia documents using timebox diagrams. It allows them to concentrate better on the creative aspects of their design as each action is validated by the system. The temporal query processing allows users to explore the property or the state of the document. They can generate the presentation schedule most commensurate to their creativity using the chosen combination of the minimal-cost fair-scheduling and the interactive scheduling mechanisms.

The constraint-based management of time has great potential in its flexibility. However, the advantages cannot be explored without sufficient interactive support tools. The interactive support tools we have described, interfaced with automatic scheduling mechanisms, provide the users with both the flexibility of a constraint-based system and fine-grained control at the same time.

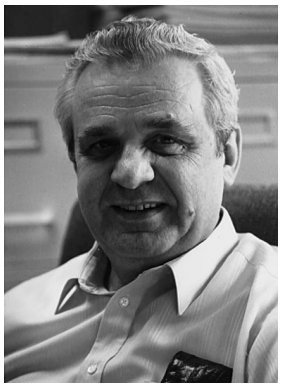
Acknowledgements. The authors thank the reviewers for helpful comments on the initial submission. The authors are also grateful to M. Kim, J. Jaffar, L. Joskowicz, J. Lassez, M. Maher for their many useful suggestions.

References

1. Ackermann P (1994) Direct Manipulation of Temporal Structures in a Multimedia Application Framework. In: Proc. of ACM Multimedia Conference, October 1994, San Francisco, Calif., ACM Press pp 51-58
2. Aho A, Hopcroft J, Ullman J (1974) The design and analysis of computer algorithms. Addison- Wesley, Reading, Mass.
3. Allen J (1983) Maintaining Knowledge about Temporal Intervals. Commun ACM 26(11): 832- 843
4. Buchanan MC, Zellweger PT (1993) Automatically Generating Consistent Schedules for Multimedia Documents. Multimedia Syst J 1(2): 55-67
5. Dechter R, Meiri I, Pearl J (1991) Temporal Constraint Networks. Artif Intell 49(1): 61-95
6. Diaz M, Senac P (1994) Time Stream Petri Nets: A Model for Timed Multimedia Information. In: Proc. 15th Conf. on Application and Theory of Petri Nets, June 1994, Zaragoza, Spain, pp 219-238
7. Edmonds J, Karp RM (1972) Theoretical improvements in algorithmic efficiency for network flow problems. J ACM 19(2): 248-264
8. Fiume E, Tsichrizis D, Dami L (1987) A Temporal Scripting Language for Object-Oriented Animation. In: Proceedings of Eurographics, August 1987, Amsterdam, The Netherlands, pp 283-294
9. Freeman-Benson BN, Maloney J, Borning A (1990) An Incremental Constraint Solver. Commun ACM 33(1): 54-63
10. Hamakawa R, Rekimoto J (1993) Object Composition and Playback Models for Handling Multimedia Data. In: Proc. of ACM Multimedia Conference, August 1993, Anaheim, Calif., ACM Press pp 273-281
11. Jaffar J, Maher MJ, Stuckey PJ, Yap RHC (1994) Beyond Finite Domains. In: Proceedings of Workshop on Principles and Practice of Constraint Programming, May 1994, Rosario, Orcas Island, Wash., pp 86-94
12. Jaffar J, Michaylov S, Stuckey PJ, Yap RHC (1992) CLP(R) language and system. ACM Trans Program Lang Syst 14(3): 339-395
13. Kim MY, Song J (1994) Hyperstories: Combining Time, Space, and Asynchrony in Multimedia Documents. IBM Research Report RC 19277. IBM T.J. Watson Research Center, Yorktown Heights, N.Y.
14. Kim MY, Song J (1995) Multimedia Documents with Elastic Time. In: Proc. ACM Multimedia Conference, November 1995, San Francisco, Calif., pp 143–154
15. MacNeil R (1991) Generating Multimedia Presentations Automatically using TYRO, the Constraint, Case-Based Designer's Apprentice. In: Proc. IEEE Workshop on Visual Languages, 1991
16. Pratt VR (1977) Two easy theories whose combination is hard. Technical Report. MIT, Cambridge, Mass.
17. Ramalingam G, Song J, Joskowicz L, Miller R (1999) Algorithmica 23: 261–275
18. Shostak R (1981) Deciding Linear Inequalities by Computing Loop Residues. J ACM 28(4): 769- 779
19. Song J, Doganata Y, Kim MY, Tantawi A (1996) Modeling Timed User Interactions in Multimedia Documents. In: IEEE International Conference on Multimedia Computing and Systems, June 1996, Hiroshima, Japan, pp 407-416
20. Song J, Dan A, Sitaram D (1997) Efficient Retrieval of Composite Multimedia Objects in the JINSIL Distributed Environment. In: Proc. of ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, June 1997, Seattle, Washington, pp 260–271
21. Tarjan RE (1983) Data Structures and Network Algorithms. Society for Industrial and Applied Mathematics, Philadelphia, Pa.
22. Tetzlaff L, Kim MY, Schloss R (1995) Home Health-Care Support System for Children with Leukemia. In: Conference on Human Factors in Computing Systems, May 1995, Denver, Colo., pp 11–12
23. Rossum G van, et al. (1993) CMIFed: A Presentation Environment for Portable Hypermedia Documents. In: ACM Multimedia Conference, August 1993, Anaheim, Calif., pp 183–188
24. Weizman L, Wittenburg K (1994) Automatic Presentation of Multimedia Documents Using Relational Grammars. In: Proc. of ACM Multimedia Conference, October 1994, San Francisco, Calif., pp 443–451
25. Wolfe W, Davidson S, Lee I (1991) RTC: Language Support for Real-Time Concurrency. In: Proceedings of IEEE Real-Time Systems Symposium, December 1991, San Antonio, Tex., pp 43-52

JUNEHWA SONG is currently a research staff member at IBM T.J. Watson Research Center. He received his doctorate degree from the University of Maryland, College Park, M.D., masters degree from the state University of New York, Stony Brook, N.Y., and bachelors degree from the Seoul National University, Seoul Korea, all in computer sciences. He has been working on various aspects of distributed multimedia systems and internet technologies. He was awarded an IBM Cooperative Graduate Fellowship from 1995–1997.

G. RAMALINGAM is currently a Research Staff Member at the T.J. Watson Research Center in Hawthorne, New York. He received his bachelor's degree in Computer Science from the Indian Institute of Technology, Madras, India, and his master's degree in Computer Science and in Mathematics and his doctorate in Computer Science from the University of Wisconsin at Madison. His research interests are in the areas of Programming Languages and Algorithms.



RAYMOND E. MILLER received his B.S. degree in mechanical engineering from the University of Wisconsin, and the B.S.E.E., M.S. and Ph.D. degrees from the University of Illinois-Urbana Champaign. He is a fellow of AAAS, ACM and IEEE. He is a Professor of Computer Science at the University of Maryland, College Park. Among his many society activities he has served on the ACM council as a Member-at-Large, on the IEEE/CS Board of Governors including Vice President for Educational Activities, on the Board of the Computing Research Association, and as President of the Computing Sciences Accreditation Board. He has written over 100 technical papers in areas of theory of computation, machine organization, parallel computation, networking and communication protocols.