Interactive Authoring of Multimedia Documents

Junehwa Song * *Michelle Y. Kim* *G. Ramalingam*
IBM T.J. Watson Research Center, Yorktown Heights, NY 10598
SONG, YOON, RAMA @ WATSON.IBM.COM

Raymond Miller *Byoung-Kee Yi*
University of Maryland, College Park, MD 20742
MILLER, KEE @ CS.UMD.EDU

Abstract

Authoring multimedia documents involves coordinating various types of media items in time and space. To be effective, authoring requires integration tools that are highly interactive. In addition, the notion of "time" must be made explicit and flexible so that users may define and manipulate the temporal behavior of documents directly and flexibly.

We describe an interactive framework for multimedia authoring, which provides users with immediate feedback. It is based on the elastic time model, where the temporal behavior of multimedia documents can be handled explicitly yet flexibly. In its implementation (the Isis authoring system), users directly manipulate time via timebox diagrams, and the system immediately checks the validity of each user interaction. The system also provides guidance for recovery in anomalous situations. Users can query various temporal properties of the document and use automatic scheduling mechanisms. These system support tools are based on a fast* incremental algorithm.

Through this approach, multimedia documents can be designed so that they are adaptable to the needs of different users and to changing environments.

KEYWORDS: Multimedia documents, multimedia authoring, interactive systems, temporal constraint systems, incremental algorithms.

1 Introduction

A notable characteristic of evolving multimedia information systems is that information comes from diverse sources at various levels of granularity and will constantly be changing. Furthermore, users have different requirements, and presentation devices on which they view the information will require vastly different design solutions. Thus, multimedia systems require flexibility: a flexible system provides adaptability to the needs of many different users and to changing environments.

Authoring multimedia systems, or documents, is a complex and iterative process which involves coordination of various types of media data, such as video, images, graphics, audio, and text both in time and space. In this paper, we focus on time and describe a framework for providing temporal flexibility in multimedia documents. Authoring the temporal behavior of multimedia documents poses difficulty in traditional programming paradigms and their supporting systems, mainly due to their inability to treat "time" effectively. We use the elastic time model[13] as a foundation for temporal flexibility, where users can define and manipulate the temporal behavior of multimedia documents explicitly yet flexibly. In the elastic time model, the playback duration of each media object, or composite object, can stretch or shrink to incorporate various uncertainties in the user's design choices or playback environments. In the Isis* authoring system, users directly manipulate the timeboxes, and thus explicitly control the temporal behavior of the document. After each user interaction, the system immediately checks its validity and, if it is not valid, the causes of the anomaly are reported to the users along with possible ways of resolving the problem. In addition, the system's query processing feature can guide users' design steps. By appropriately combining the interactive support and the system's automatic layout generation mechanism, users can fully utilize the flexibility embedded in the elastic time model.

Our elastic time model is based on the temporal constraint formalism. Other constraint based approaches for handling time in multimedia documents have been reported in [3, 9]. [3] reported a mechanism for automatically obtaining an optimal temporal layout using a linear programming technique. [9] adopted the notion of temporal glue and box, along with the constraints to specify relationships between events. Their focus was on the development of automatic mechanisms for generating temporal layouts and mentioned the issue of providing the system support for detecting anomalies in the document specification as an important open problem [3, 9]. In [13], we described a minimal cost fair scheduling algorithm for the elastic time model.

*The author's current address is: University of Maryland, College Park, MD 20742, JUNESONG @ CS.UMD.EDU

This paper describes the interactive aspect of our system and discusses various ways in which the authoring process can take advantage of the system's interactive support. The system support facilities for interactive authoring include: anomaly handling, temporal query processing, and automatic scheduling, which utilize a fast incremental algorithm. A more complete description of our model of multimedia documents, including the mechanisms for treating asynchronous user events, can be found in [11, 17]. This paper is organized as follows. In Section 2, we provide a brief overview of the elastic time model and define a set of user interaction primitives. In Section 3, we provide a mechanism for translating the user interaction primitives into a temporal constraint system. Section 4 presents our incremental algorithm, Section 5 describes the system support facilities for interactive authoring: anomaly handling, temporal query processing, and the automatic scheduling mechanism. We provide an example that illustrates the authoring process aided by the system. Section 6 concludes the paper.

2 Elastic Time Model

Authoring is an iterative process involving repeated composition and validation cycles. Even a completed document may have to be presented idfferently in different presentation environments.

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", a "maximum", and an "at rest" length. Users can connect the (elastic) objects by defining temporal relationships among them. In doing so, users avoid tedious positioning and repositioning of media objects in time while composing a document. Furthermore, once constructed, a document is associated with a range of possible layouts, so that the users can choose one appropriate for their presentation.

2.1 Temporal specification

We base our specification language on Allen's temporal algebra[2], and take time intervals as primitives. Let a multimedia object m be associated with a triple of lengths, which we call spring constants:

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

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, where λ_m is the optimal length.

The following set of primitive relations can hold between any number of objects:

- *co-start*$(m_1, \ldots, m_n)$: time intervals $I_{m_1}, \ldots, I_{m_n}$ share the same beginnings,
- *co-end*$(m_1, \ldots, m_n)$: time intervals $I_{m_1}, \ldots, I_{m_n}$ share the same ends,
- *meet*$(m_1, \ldots, m_n)$: For $i = 1, \ldots, n-1$, the end of I_{m_i} is shared by $I_{m_{i+1}}$ as its beginning,

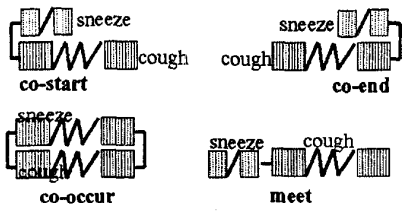


Figure 1: Four temporal relations for building elastic stories

- *co-occur*(m_1, m_2) : time interval I_{m_1} is set to be the same as that of I_{m_2}.

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 non-zero intervals. A dummy object can be used to cause a time delay. With this, relationships such as *overlap* can also be expressed using the four primitives.

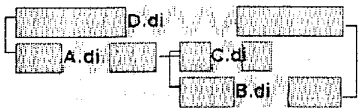
2.2 The time-box representation

We represent multimedia objects visually using a simple building block metaphor. Multimedia objects are treated as electronic building blocks, or time-boxes, such that the lengths of the time-boxes are proportional to the (optimal) lengths of the corresponding objects. See Figure 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 timeboxes to specify the temporal behavior of documents: they can drag and drop timeboxes, resize, connect, separate, remove them, using the system provided graphical user interfaces. We summarize below our set of user interaction primitives for designing multimedia documents interactively.

- $+timeBox(t_i)$: 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 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.



(a) TimeBox Diagram

media object	min	max
A	10	15
B	13	15
C	5	8
D	25	35

(b) Minimum and Maximum Play Duration

Figure 2: Authoring with Four Media Objects: media objects, A, B, C, D, are related by *co-start*(A, D), *meet*(A, C), *co-start*(B, C), and *co-end*(B, D).

3 Construction of Temporal Constraint Network

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

In our system, the state of a document is maintained as a *temporal constraint system*, which is represented by a directed, weighted graph called *temporal constraint network* (*TCN*). The system maintains the TCN incrementally. It translates each user interaction into a sequence of low level graph operations. 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 Section 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. In the rest of this paper, we use the example in Figure 2 and Figure 3.

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 $m_i.e$ denote the start time and end time of an object $m_i \in M$, respectively. For each $m_i \in M$, we have a constraint:

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

Each temporal relationship $r(m_i, m_j) \in R$ also induces constraints. For instance,

$$meet(m_1, m_2) \Longrightarrow m_1.e = m_2.s.$$

We call such linear constraints for a given document S "the constraint set of document S", which we denote by $C(S)$.

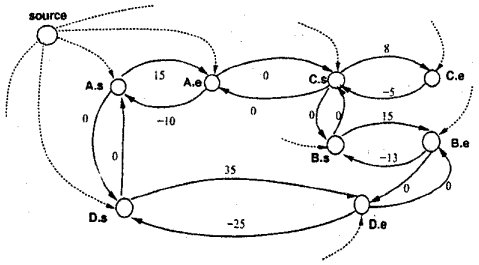


Figure 3: Temporal Constraint Network for the Document in Figure 2 : dotted arrows represent the arc from source to each node with weight 0

All these constraints can be expressed using constraints of the form $x - y \leq c$.[1] $C(S)$ is internally represented by a directed, weighted graph, called Temporal Constraint Network, as previously explained, which is denoted by

$$TCN(S) =< V, E, w >.$$

For each variable x in $C(S)$, we include a node in V [2]. For each constraint $x - y \leq c$ in $C(S)$, we include a directed arc $(y, x) \in E$ with a weight c, $w(y, x) = c$. We also include a special node *source* in V. For each node x in V, we include an arc $(source, x) \in E$, with $w(source, x) = 0$.

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

- $+node(n_i)$: to add a node n_i to V and the special arc $(source, n_i)$ to E with $w(source, n_i) = 0$.
- $-node(n_i)$: to remove a node n_i from V and the special arc $(source, n_i)$ from E.
- $+arc(n_i, n_j, c)$: to add an arc (n_i, n_j) to E with the weight c, i.e., $w(n_i, n_j) = c$.
- $-arc(n_i, n_j)$: to remove an arc (n_i, n_j) from E[3]
- $weight(n_i, n_j, c)$: to modify the weight of the arc $(n_i, n_j) \in E$ to c, i.e., $w(n_i, n_j) = c$.

Each user interaction is translated into a sequence of graph operations as summarized below. Note that initially a TCN consists only of a special node *source*.

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

[1] We call this kind of linear constraints difference constraints. Difference constraints have been used as a popular tool to specify temporal behavior in many applications [3, 4, 5, 13, 22].

[2] Without loosing generality, we use the same name both for variables in $C(S)$ and nodes in V.

[3] We assume that there is only one arc for an ordered pair of nodes.

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

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 illustrates this:

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

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. Consider the example in Figures 2 and 3. Figure 3 is the TCN of the document which is represented by the timebox diagram in Figure 2. From the current state, if the user adds a new relation: $+relation(co\text{-}end, B, C)$, two graph operations

$+arc(B.e, C.e, 0)$; $+arc(C.e, B.e, 0)$

will be generated.

4 Incremental Algorithm

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

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

The system $C(S)$ is said to be feasible, or satisfiable, if there exists an assignment of values to variables in $C(S)$ which simultaneously satisfies all the constraints. The feasibility of $C(S)$ is equivalent to the absence of a negative cycle in $TCN(S)$ [4, 15]. 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 creating 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. For each addition of a new arc, our algorithm takes $O(n \log n + e)$ 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 [4], which takes $O(n^3)$ time.

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

- $+node(n_i)$: 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 n_i with $w(source, n_i) = 0$, and extends $\vec{S}$ by setting $\vec{S}(n_i) = 0$.
- $-node(n_i)$: 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 n_i is an isolated node. The node is simply removed from V, and from the solution vector $\vec{S}$.
- $-arc(n_i, n_j)$: A deletion of an arc $(n_i, n_j) \in E$ 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, $n_j - n_i \leq w(n_i, n_j)$.
- $+arc(n_i, n_j, c)$: An addition of an arc potentially introduces a negative cycle. The procedure *AddArc* in section 4.1 handles this operation.
- $weight(n_i, n_j, c)$: Let $w(n_i, n_j) = c_1$ before the *weight* operation. If $c_1 \leq c$, i.e., if it is to increase the weight of an arc, 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 it is to decrease the weight of the arc, it is treated as a deletion of the arc followed by an insertion of a new arc with the corresponding weight.

4.1 Adding a new arc

Given a TCN(S) without any negative cycle and an operation $+arc(u, v, c)$, we 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 $O(nlogn + e)$[1]. Below is the procedure *AddArc* to handle $+arc(u, v, c)$.

Dijkstra's algorithm works only for directed graphs with non-negative weights. But our TCN may contain negative weight arcs as shown in Figure 3. We use the weight transformation technique of [6] to handle negative weight arcs. Details of how we apply this technique can be found in [16].

Procedure $AddArc(TCN(S), (u, v), c, \vec{S})$
$TCN(S) =< V, E, w >$
(u, v) : arc to be inserted to E
c : the weight of the arc (u, v)
$\vec{S}$: the feasible solution vector

[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

[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] $w(u, v) = c$;

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]. This transformed weight, $w_{transf}(x, y)$, is always non-negative 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 by computing the shortest distance from the *source* node to each node x in the updated TCN (Step [5.3])[4]. 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 Steps [5.1] and [5.2], and cannot use the current solution $\vec{S}$.

5 System Support for Interactive Authoring

Although the elastic time model provides a foundation for temporal flexibility in the document, it should 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 automatic scheduling. We describe each of the features below and provide a simple example to demonstrate how they work in a cooperative manner and support interactive multimedia authoring.

Our system support tools are based on the incremental algorithm described in Section 4.

[4] We cannot 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 non-negative.

5.1 Anomaly handling

An interactive programming system —which is what an interactive authoring system is— should offer mechanisms for diagnosing anomalous behavior. One of the problems with declarative specifications in general is that it can be difficult to identify the causes of infeasibilities. Infeasibility is not always generated by a single media object or a single temporal relationship. Rather, it is formed by imposing multiple temporal relationships imposed on multiple media objects. Treating infeasibility in a document is complicated when there are multiple sources for infeasibility, possibly interrelated.

Our approach of interactive authoring with immediate evaluation simplifies the 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, it rejects the interaction and provides appropriate feedback to the user. 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.

When infeasibility is detected, the easiest way to remedy it is to force the user to give up the current user interaction primitive. However, this may not be acceptable to the user. To address this issue, the system provides two types of services: those to be used if the user want to preserve the current interaction primitive and those to be used if the user is willing to relax the constraint.

5.1.1 Preserving the current interaction primitive

If the user wants to preserve the current choice, the user will have to change other parts of the document, changing relationships, durations of time-boxes, adding or removing delay objects, and so on, keeping the design choices in mind.

In general, this recovery process can be tedious. Consider the simple example in Figures 2 and 3. Suppose the user wants to impose a relationship $+relation(co\text{-}end, B, C)$ and the system detects infeasibility when $+arc(C.e, B.e, 0)$ is applied to $TCN(S)$. To remedy the infeasible document, the user first needs to recognize that media objects B and C and their relationships $co\text{-}start(B, C)$, $co\text{-}end(B, C)$ are not compatible. The user may choose to remove the relationship $co\text{-}start(B, C)$, which will remove the incompatibility. However, the document is still infeasible. In fact, the entire document consisting of media objects A, B, C, D, and the three relationships, $co\text{-}start(A, D)$, $meet(A, C)$, $co\text{-}end(B, D)$, are incompatible with the new relationship $co\text{-}end(B, C)$. Such a situation calls for a more careful treatment in anomaly handling. Our system helps the user resolve inconsistencies by highlighting the groups of objects and relationships that are incompatible with the new user action, one by one.

5.1.2 Relaxing the current interaction primitive

Instead of going through the possibly tedious recovery process outlined above, the user can ask the system to suggest a relaxed form of the current user interaction that avoids creating

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. $l \geq -length(P)$
$+relation(co\text{-}end, t_i, t_j)$	$+arc(t_i.e, t_j.e, 0)$	$+timeBox(delay)$; $+relation(meet, t_i, delay)$; $maxTime(delay, l)$, s.t. $l \geq -length(P)$

Table 1: Possible corrective actions: $length(P)$ is the length of shortest path P with which a negative cycle has been detected in the procedure *AddArc*, and *delay* is a time-box which represents a dummy, or delay object.

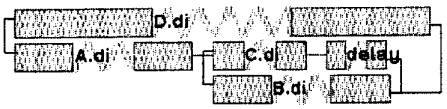


Figure 4: Suggested Correction: when a new, invalid, relationship $co\text{-}end(B, C)$ is added to the document in Figure 2, system suggests to add a delay after the media object C

inconsistency.

To see how the suggestions are constructed, let's consider a set $C(S)$ and a $TCN(S)$ for a document S. Assume that $x - y \leq c$ is the constraint that creates inconsistency 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 a negative cycle. If we change $TCN(S)$ so that $dist(y, x) + dist(x, y) \geq 0$ (for example, by changing $w(y, x)$ to c', such that $c' + dist(x, y) \geq 0$), all the negative cycles will become non-negative. The system suggests a sequence of user interaction primitives to achieve this effect. See Table 4 for an example.

Consider the example in Figures 2 and 3. Upon the user interaction

$+relation(co\text{-}end, B, C)$,

which is infeasible, the system suggests "add a delay object (with maximum duration of at least 5 seconds) after the object C, and connect it with the object B by $co\text{-}end(delay, B)$" as is shown in Figure 4. This results in adding intermediate nodes and arcs between $C.e$ and $B.e$ to $TCN(S)$, and thus making $dist(B.e, C.e) + dist(C.e, B.e) \geq 0$. Suggestions such as this may not be acceptable the user, but they provide useful information and make the system 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 users do not have to go through tedious computation of time positions of media objects. However, users may still want to know certain temporal properties of the document during the authoring process.

As a document becomes flexible (with elastic time), inferring even 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[5] in a document can vary within a range, and the total presentation time of the document may vary.

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

- What is the distance between the events e_i and e_j in the current document?
- How long does it take to show 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[4]. Both can be computed by Dijkstra's algorithm modified by the weight transformation technique[16].

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

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

Then, the *GlobalDuration* can be defined by,

GlobalDuration = [MinD, MaxD],

where, $MinD = min_{\vec{S}}\{D(\vec{S})\}$ and $MaxD = max_{\vec{S}}\{D(\vec{S})\}$.

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 R as follows,

$F = \{n_i |$ if $(n_i, n_j) \in E$, then $w(n_i, n_j) \geq 0\}$,

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

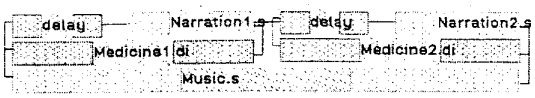


Figure 5: Example

$R = \{n_j \mid \text{if}(n_i, n_j) \in E, \text{ then } w(n_i, n_j) \geq 0\}$.

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

$MinD = -min_{n_i \in F}\{min_{n_j \in R}\{dist(n_j, n_i)\}\}$

$MaxD = max_{n_i \in F}\{max_{n_j \in R}\{dist(n_i, n_j)\}\}$.

That is, we can compute the $GlobalDuration$ by computing the shortest distances between the nodes in F and R. This can be done by using the Dijkstra's algorithm repeatedly or by using an all-pairs-shortest-paths algorithm, such as Floyd-Warshall's[4], just once.

5.3 Automatic scheduler

If a document does not include an anomaly, the automatic scheduler can generate a schedule. The system provides two scheduling mechanisms. Upon each user interaction, our system immediately generates a schedule. This schedule is generated by the incremental algorithm for the temporal constraint system. It can be used if the user is satisfied with this schedule, which does not ensure optimality.

Our system can also generate a minimum-cost fair schedule. 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 minimal-cost fair scheduling algorithm selects a schedule which minimizes the sum of the costs while considering "fairness" in the distribution of the total cost over the corresponding set of media objects. The details of the minimal-cost fair scheduler are explained in [13].

5.4 Example

Consider the simple multimedia document with a set of objects and relationships shown in Table 2. The timebox representation of the document is in Figure 5.

The document is to advertise two drugs by a drug manufacturer. *Medicine1* and *Medicine2* are their two graphical 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. Background music also plays for the duration of the document.

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, which is summarized in Table 2 (c).

However, after some experiences with the presentation of the document, the user may find that there is more interest in *Medicine1*, and more than 25 seconds is desired to describe the medicine. Suppose the user increases the presentation duration of *Medicine1* to its maximum, 28 seconds. 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 seconds (as requested by the user) and optimize the remaining objects so that the total cost is minimal and fair. Now suppose that the user tries to increase the duration, beyond its maximum, to 33 seconds. The system immediately informs the user that this is not possible, and that if 33 seconds are needed, the objects (*Medicine1*, *Medicine2*, Music) or the relationships among them should be changed.

At this point, the user may want to know what temporal range is possible for *Medicine1*, using the $Projection$ query, and the system immediately returns $(24, 31)$. The users may then adjust the presentation duration of the image to 30 seconds, which will keep the document consistent. With this, the system computes a schedule with a fixed duration of 30 seconds for the 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 utility will grow rapidly as the complexity of the document increases and the users' requirements grow in sophistication.

6 Conclusion

We have presented the elastic time model and our prototype authoring system. The elastic time model provides a framework to flexibly handle the temporal behavior of multimedia documents. The prototype system provides a set of system support tools to facilitate the authoring process and to utilize the flexibility embedded in the elastic time model. Users can explicitly control the time in multimedia documents by directly manipulating the timebox diagram. Upon each user interaction, the system immediately checks its validity and, if it is not valid, suggests possible actions that will recover from the anomaly. The system also supports users in interactive authoring by answering the temporal queries: $Projection$ and $GlobalDuration$. Two mechanisms for automatic scheduling are given: a feasible schedule is always available, and the minimal-cost fair schedule can be computed upon user's request.

In the elastic time model, each media object is associated a range of presentation durations to provide flexibilities in design choices and in the presentation environment. Using the system support tools, users can easily reflect their changing design choices and presentation environments. The elastic time

name	type	min	opt	max
delay1	delay	8	10	12
delay2	delay	8	10	12
Narration1	audio	10	12	14
Narration2	audio	10	12	14
Medicine1	image	19	25	28
Medicine2	image	19	25	28
Music	audio	50	50	50

(a) Media Objects and Spring Constants

meet	(delay1, *Narration1*)
meet	(delay2, *Narration2*)
co-start	(delay1, *Medicine1*)
co-start	(delay2, *Medicine2*)
co-end	(*Narration1*, *Medicine1*)
co-end	(*Narration2*, *Medicine2*)
meet	(*Medicine1*, *Medicine2*)
co-start	(*Medicine1*, Music)
co-end	(*Medicine2*, Music)

(b) Temporal Relationships

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

(c) Minimal Cost Fair Schedule

Table 2: Example

model and the corresponding system support facilities will ease the complex authoring process involving iterative design and validation cycles.

Providing a rich, interactive authoring environment is essential for authoring innovative and successful multimedia applications. Our approach can be generalized and adopted to other more general systems. In particular, the incremental algorithm and the mechanism for handling anomalies in the document can easily be utilized by other constraint based authoring systems.

Acknowledgement

The authors are grateful to J. Jaffar, L. Joskowicz, J. Lassez, M. Maher for their many useful suggestions.

References

[1] Aho, A., Hopcroft, J., Ullman, J., *The design and analysis of computer algorithms*, Addison-Wesley Publishing Company, 1974.

[2] Allen, J., "Maintaining Knowledge about Temporal Intervals", CACM, 26(11), 1983.

[3] Buchanan, M. C., Zellweger, P. T., *Automatically Generating Consistent Schedules for Multimedia Documents*, Multimedia Systems Journal, 1(2), pp. 55-67, Springer-Verlag, 1993.

[4] Dechter, R., Meiri, I., and Pearl, J., *Temporal Constraint Networks*, Artificial Intelligence, 49, pp61-95, 1991.

[5] Diaz, M., Senac, P. *Time Stream Petri Nets: A Model for Timed Multimedia Information*. Proc. 15th Conf. on Application and Theory of Petri-nets, 1994.

[6] Edmonds, J., Karp, R. M., *Theoretical improvements in algorithmic efficiency for network flow problems*, Journal of ACM, 19, pp. 248-264, 1972.

[7] Fiume, E. et al., *A Temporal Scripting Language for Object-Oriented Animation*, Eurographics, 1987.

[8] Freeman-Benson, B. N., Maloney, J., Borning, A., *An Incremental Constraint Solver*, Communications of the ACM, 33(1), 1990.

[9] Hamakawa, R. and Rekimoto, J., Object Composition and Playback Models for Handling Multimedia Data, ACM Multimedia 93, pp. 273-281, 1993

[10] Helm, R., Huynh, T., Marriott, K., Vlissides, J., *An Object-Oriented Architecture for Constraint-Based Graphical Editing*, Object-Oriented Programming for Graphics, pp. 217-238, Springer-Verlag, 1995

[11] Kim, M. Y., Song, J., *Hyperstories: Combining Time, Space, and Asynchrony in Multimedia Documents*, IBM Research Report RC 19277, 1994.

[12] Kim, M. Y., *Creative Multimedia for Children*, Conference on Human Factors in Computing Systems, 1995.

[13] Kim, M. Y., Song, J., *Multimedia Documents with Elastic Time*, ACM Multimedia, 1995.

[14] MacNeil, R., *Generating Multimedia Presentations Automatically using TYRO, the Constraint, Case-Based Designer's Apprentice*, IEEE Workshop on Visual Languages, 1991.

[15] Pratt, V. R., *Two easy theories whose combination is hard*, Massachusetts Institute of Technology, 1977.

[16] Ramalingam, G., Song, J., Joskowicz, L., Miller, R., *Solving Difference Constraints Incrementally*, IBM Computer Science Research Report RC 89573, 1995.

[17] Song, J., Doganata, Y., Kim, M. Y., Tantawi, A., *Modeling Timed User Interactions in Multimedia Documents*, IEEE International Conference on Multimedia Computing and Systems, 1996.

[18] Tarjan, R. E., *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, 1983.

[19] Tetzlaff, L., Kim, M. Y., Schloss, R., *Home Health-Care Support System for Children with Leukemia*, Conference on Human Factors in Computing Systems, 1995.

[20] van Rossum, G. et al., *CMIFed: A Presentation Environment for Portable Hypermedia Documents*, ACM Multimedia, 1993.

[21] Weizman, L., Wittenburg, K., *Automatic Presentation of Multimedia Documents Using Relational Grammars*, ACM Multimedia, 1995.

[22] W. Wolfe, W., S. Davidson, S., Lee, I., *RTC: Language Support for Real-Time Concurrency*, IEEE Real Time Systems Symposium, 1991, pp. 43-52.