Constraints for multimedia documents

Full contact details:

Address: Projet Opéra
INRIA Rhône-Alpes
655, avenue de l'Europe
38330 Montbonnot Saint-Martin
France

Phone: +33 4 76 61 53 83
Fax: +33 4 76 61 52 07
E-Mail: {Laurent.Tardif, Frederic.Bes, Cecile.Roisin}@inrialpes.fr

Abstract:

In constraint based authoring tools, one of the major issues is to provide efficient constraint solvers in term of time performances and solution quality. The solver has a central position in this kind of authoring environments.

In this paper, some important requirements implied by multimedia authoring process are described. This study intends to provide a better understanding of what would be the best constraint solver for a multimedia authoring tool. At different stage of the authoring process, we present then our experiment using different kinds of solvers. The results allow us to propose a solution that combines different approaches in order to fit better the requirements.

Constraints for multimedia documents

Laurent Tardif, Frédéric Bes and Cécile Roisin

OPERA project - INRIA Rhône-Alpes
{Laurent.Tardif, Frederic.Bes, Cecile.Roisin}@inrialpes.fr
http://www.inrialpes.fr/opera/

1. Introduction

Multimedia documents compose in time and space different media like video, sound, picture or text. Three important reasons make the multimedia authoring a complex task.

- The inherent temporal dimension of media (like audio and video).
- The organization of the media over time.
- The incremental nature of the authoring process.

Thus the static WYSIWYG approach used in traditional text-based system cannot be applied. To address these difficulties, we promote two ideas: direct editing in synchronized views and specification of both the temporal and spatial organizations using constraints.

These ideas are experimented in a prototype named Madeus (Layaïda, 97). Madeus takes in charge the temporal and spatial constraints expressed by the author in order to check their consistency and to compute one solution. In this paper we focus on authoring requirements for designing multimedia documents and on the ability of the authoring tool, through the use of different constraint solvers, to fulfil these requirements. Since the time performances of constraint algorithms and their capacity to compute a relevant solution are major issues, we mainly present in this paper the results of our experiments while using different kinds of solvers and the conclusion we learned from these experiments.

2. Context of our work

We describe in this section Madeus the authoring and presentation environment for multimedia documents which is the basis of the work presented in this paper.

2.1 Madeus multimedia documents

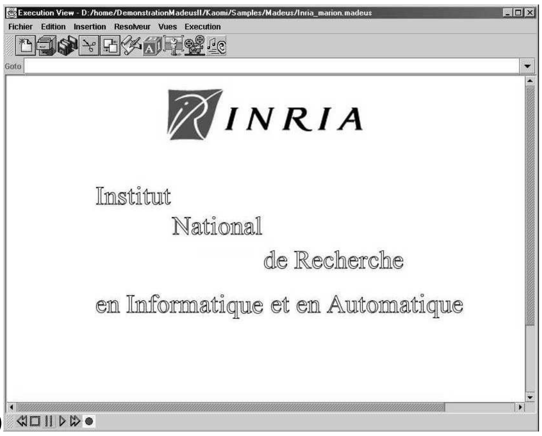
In Madeus, a multimedia document is defined by a set of media, which can be either dynamic such as video or audio, or static like text or picture. These media define the basic elements of a multimedia document called in the following *objects*. Objects are linked by relations. The relations define the temporal and the spatial organization of the document in a high level of abstraction: temporal relations are based on Allen relations (Allen, 83), this means that we use operators like *Before*, *Equals*, *Meets* which allow the author to define temporal placements between objects. The same approach is taken for the specification of the spatial relations through the use of relations like *TopAlign*, *LeftAlign*, *HorizontalCenter.* Some relations accept a parameter: *A TopAlign 10 with B* means that the top of object A is 10 units greater than the top of object B. The Madeus environment also provides scale relations, for expressing properties like *the width of A is equal to the height of B*.

Finally the author can organize his document into groups of objects called *composites*. This structure helps the author in managing complex documents and in reusing subparts of presentations in other presentations. This organization is frequently used for the spatial dimension, but less known in the temporal one. The semantics of composites are the same both in the temporal and in the spatial dimension: the dimensions and positions of a composite are defined as the minimal (temporal/spatial) bounding box of the objects that belong to the composite.

2.2 A multiview authoring environment

Madeus provides different views to offer a better perception of multimedia documents together with authoring facilities through each of the views (Jourdan et al, 98b).
These views are:

- The *execution view* (Figure 1a) which provides the basic functionality to play/stop/resume the document. He can also insert objects or change spatial attributes on an object by selecting it and moving (or scaling) it.

(a) 

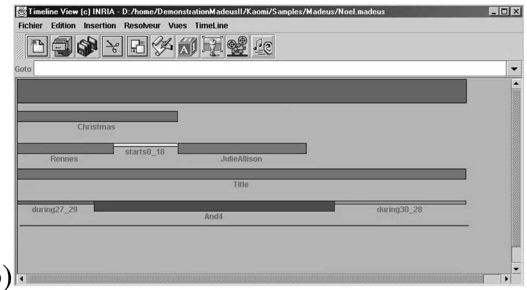
(b) 

Figure 1: (a) Execution view and (b)Timeline view in Madeus

- The *timeline view* (Figure 1b) which offers a graphical visualization of the temporal dimension of the document. In this view, the author can select an object and change its position along the timeline.
- The *hierarchical view* in which the structure of a document is displayed and can be edited.

The system synchronizes these authoring views in such a way that every modification applied on a view is transmitted to the other views.

2.3 Constraints in Madeus

Let's now focus on how the high level relations are converted into constraints[1] inside Madeus.
The internal model of Madeus uses constraints to handle and maintain temporal and spatial relations between objects during the authoring process. Each time a user-defined relation is inserted in the document, it is translated into constraints upon internal variables representing the object dimensions. These constraints are given to a constraint solver, which checks the consistency of the resulting constraint network. The solver computes then a new solution of the constraint network that takes into account the new inserted relation. We develop below how this action is performed, first for temporal relations, then for spatial relations and finally we give an example issued from a sample document used for the presentation of INRIA (Figure 1).

[1] Notice that we use in the following, the word *relation* to denote the relation set upon media objects and the word *constraint* to denote the relation set between variables of the constraint network.

2.3.1. Temporal relations

Each multimedia object is defined by a set of three temporal attributes (*Start*, *End*, *Duration*). Each attribute is defined by a triplet of variables (*Min*, *Max*, *Current*) related by the constraint: Min ≤ Current ≤ Max.
The Min and Max variables define a range for the attribute, the Current variable defines the current value of the attribute. For example, an object starts at the Start.Current instant with the duration Duration.Current and finishes at the End.Current instant.
We also have some constraints implied by the semantics of the attributes:
Duration.Min = End.Min – Start.Max ;
Duration.Current = End.Current – Start.Current ;
Duration.Max = End.Max – Start.Min ;
Each high level relation used in Madeus is automatically converted into constraints between these variables. Table 1 gives the correspondence between relations and constraints for three relations.
In this table, we use a simplifyed notation.
A.Start = B.Start, means:

A.Start.Min = B.Start.Min,
A.Start.Current = B.Start.Current and
A.Start.Max = B.Start.Max.

Temporal Relation	**Constraints on variables**
Equals A B	A.Start = B.Start and A.End = B.End or A.Start = B.Start and A.Duration = B.Duration
Before A B t	If IsDefined(t) A.End + t = B.Start else A.End < B.Start
Inside A, Group	A.Start ≥ Group.Start and A.End ≤ Group.End

Table 1: Correspondence between some temporal relations and Madeus constraints

To maintain the semantics of the minimal bounding box on temporal composite objects we must insert two constraints:

- Group.Start= Min(Child.Start)
- Group.End= Min(Child.End)

Where Child represents all the objects which have a relation *Inside* with the object *Group*.

2.3.2. Spatial relations

Each multimedia object is defined by a set of spatial attributes (Right, Left, Width, Bottom, Top, Height, XCenter, YCenter). Each attribute is defined by a triplet of variables (Min, Max, Current) related by a constraint: Min ≤ Current ≤ Max.
Some constraints are implied by the semantics of the attributes (the positioning is defined from the bottom left corner):
Width = Left – Right and Left < Right
XCenter (The horizontal middle of the object) = Left + (Width /2)
Height = Top – Bottom and Top > Bottom
YCenter = Bottom + (Height /2)

The Madeus spatial relations are converted into constraints as follows (see Table 2):

Spatial Relation	Constraints on variables
TopAlign A B d	If IsDefined(d) A.Top + d = B.Top else A.Top = B.Top
Scale d1 d2 f (where d1 and d2 are dimension attributes width or height)	If IsDefined(f d1 * f = d2 else d1 = d2
Inside A Group	A.Top $\leq$ Group.Top A.Bottom $\geq$ Group.Bottom A.Left $\geq$ Group.Left A.Right $\leq$ Group.Right

Table 2: Correspondence between some spatial relations and Madeus constraints

To maintain the semantics of the minimal bounding box on spatial composite objects we must insert four constraints:

- Group.Top= Min(Child.Top)
- Group.Bottom= Min(Child.Bottom)
- Group.Left = Min(Child.Left)
- Group.Right= Min(Child.Right)

Where Child represents all the objects which have a relation *Inside* with the object *Group*.

2.3.3. A sample of Madeus document

To give a better idea of the constraints managed for a document, we show in Table 3 the set of temporal and spatial constraints generated for the simple document of Figure 1.

Document	Implicit temporal relations	Temporal constraints
<TemporalComposite Id=TC1 > < TE Id="O1" Logo /> < TE Id="O2" Institut /> < TE Id="O3" National /> < TE Id="O4" De_Recherche /> < TE Id="O5" en_Informatique /> < TE Id="O6" et_Automatique /> <Relation > <Finishes 01 02 > <Finishes 01 03 > <Finishes 01 04 > <Finishes 01 05> <Finishes 01 06> </Relation> </TemporalComposite >	Inside O1 TC1 Inside O2 TC1 Inside O3 TC1 Inside O4 TC1 Inside O5 TC1 Inside O6 TC1	O1.Top $\leq$ TC1.Top O1.Bottom $\geq$ TC1.Bottom O1.Left $\geq$ TC1.Left O1.Right $\leq$ TC1 .Right …… TC1.Start = Min(O1.Start, O2.Start, O3.Start, O4.Start, O5.Start, O6.Start) ……. O1.End = O2.End O1.Duration > O2.Duration ……….

Document	Implicit spatial relations	Spatial constraints
<SpatialComposite Id=SC1 > <Id="O1" Logo > <SpatialComposite Id=SC2> < SE Id="O2" Institut /> < SE Id="O3" National /> < SE Id="O4" De_Recherche /> < SE Id="O5" en_Informatique />	Inside O1 SC1 Inside SC2 SC1 Inside O2 SC2 Inside O3 SC2 Inside O4 SC2 Inside O5 SC2 Inside O6 SC2	O1.Top $\leq$ SC1.Top O1.Bottom $\geq$ SC1.Bottom O1.Left $\geq$ SC1.Left O1.Right $\leq$ SC1.Right …… SC1.Left= Min(O1.Left,SC2.Left)

< SE Id="O6" et_Automatique		…….
/>		
<Relation >		
< LeftAlign O2 O5 />		O2.Left = O5.Left
</Relation >		
</SpatialComposite>		
<Relation >		
<HorizontalCenter SC2 O1>		
<Below SC2 O1>		SC2.Ycenter = O1.Ycenter
</Relation >		SC2.Top<O1.Bottom
</SpatialComposite>		

Table 3: Correspondence between relations and constraints on an example

This example illustrates the huge number of variables and constraints that must be managed even for a very simple multimedia document.

In this sample the temporal structure involves 6 basic objects (6*3 attributes, thus 6*3*3 variables that produce 6*9 intrinsic constraints) and 1 composite object (1*3 attributes, thus 1*3*3 variables that produce 9 + 2*3 constraints). Due to the composite semantics, 6 *inside* implicit relations (6 * 2 * 3 constraints). The author has defined 5 explicit relations each one translated into two constraints (5 *2 *3 constraints). So, for this sample we have 63 temporal variables and 135 temporal constraints.

The number of variables and constraints for the spatial dimension is higher because there are two axes to manage.

3. Authoring requirements and constraint techniques

Multimedia authoring requirements can be divided in two categories of needs:

- Expressive power (i.e. the ability of the system to cover a broad range of temporal scenarios required by the author).
- Authoring capabilities (for example the level of expertise required for an author to use the system, the straightforward design or abstraction capabilities).

Constraint authoring tools offer a good compromise between these two criteria (Jourdan et al, 98a). One of their main advantages is their capability to cope with the incremental nature of the authoring process. In fact, when an author needs to change a document, he is not required to reconsider the whole structure of the document but only to add or remove a relation. The authoring system automatically adjusts the displayed document to the current set of constraints.

During the authoring process the author describes relations between objects and these relations can introduce some flexibility in the document (for example the relation *Before* when the *t* parameter is not specified). Thus, when the author specifies a document, he has to manage the set of possible solutions in order to play the document. This flexibility is useful during the authoring process and even during the execution process but the use of constraints implies some new requirements during the authoring process. They are introduced in following sections.

3.1 Authoring process: general overview

During the authoring process (Figure 2) every editing action must be immediately applied and its resulting effect must be immediately displayed to the author. Moreover, the visual distortions must be minimized (i.e. the results must be gradually and smoothly displayed so that the author notices what are the changes that occurred). Therefore, after each change in the document, the new solution worked out by the constraint solver must be as near as

possible to the previous one (a more formal definition for a function to evaluate the distance of two solutions will be given in section 3.5.1).
Let's consider how the current valuation of the variables of the constraint network changes during the authoring process of a document. We can represent the interactions between the constraint solver and the author during the authoring process by the two following cases:

- The author changes the current solution by an editing operation that only changes the value of some current variables (in the range of [Min, Max]).
- The author changes the constraint network itself by adding or removing an object or a relation.

In both cases, the new valuation of the variables will be performed by the constraint solver according to some criteria and/or distance function that aim at fulfilling authoring needs.

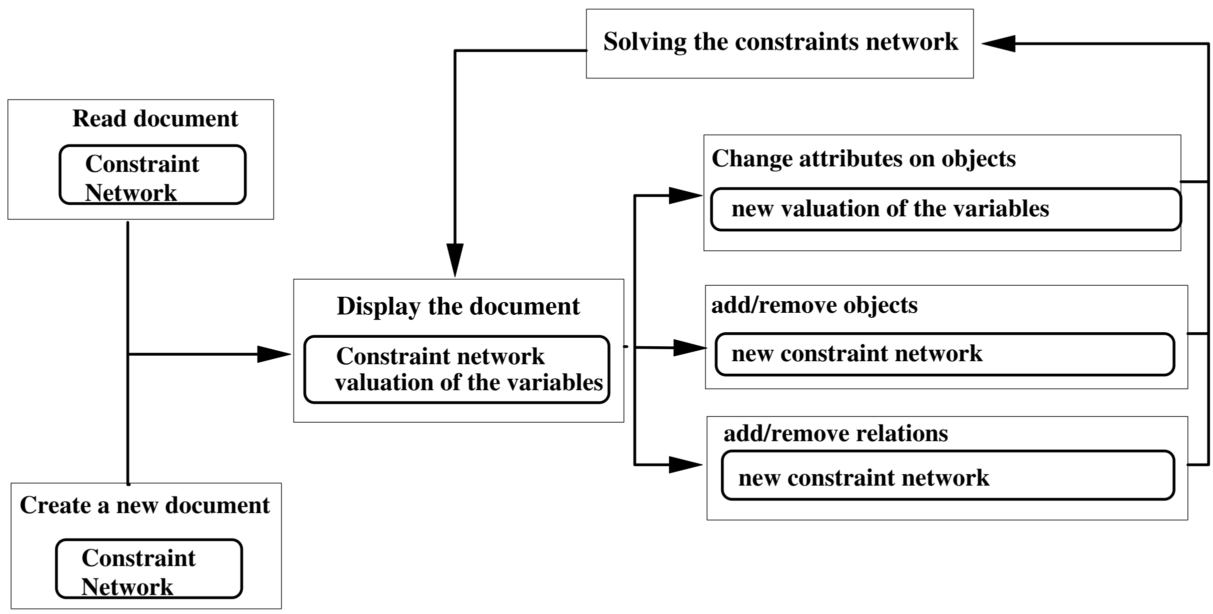


Figure 2: The authoring process

So the authoring process comes with special needs for constraint management as developed below:

1. Consistency checking
2. Anticipation
3. Completeness
4. Finding the best solution

3.2 Consistency checking

At each step of the authoring process the system must check if the new action is consistent with the current solution or not. In fact, the author can manage neither the complexity of the document nor the consequences of every action. This checking must be time efficient. It is not liable for an author to wait for a long time each time he applies changes in the document.
When an inconsistency is detected, the author knows that it was caused by his last authoring action. But it would be useful that the system gives more details on how the inconsistency occurred. Displaying the relation(s) that has caused the inconsistency is not enough. The system must give a more complete answer to allow the author to understand the problem. For instance, by giving him some hints for changing the constraint network in such way that the last authoring action becomes consistent (Song et al, 1999).

3.3 Anticipation

An important need for authoring is anticipation. For example, when the author selects an object in order to move it in the spatial dimension, it is important that the system informs the author about the range of possible displacements. This interval is basically defined by the value of the Min and Max variables.

Notice that this interval is not only due to the object properties and the media context but also to the document context (i.e. the relation in which the object is involved).

For example (see Table 4), if the author defines the following document:

Objects	Relations
A: [4 4 4]	Meets A B
B: [10 10 10]	Meets B C
C: [4 4 4]	Before A D
D: [1 3 6]	Before D C

Table 4: An example to illustrate the requirement for anticipation

The current solution is displayed in Figure 3, where A=4, B=10, C=4, D=3. Let's suppose that when moving the object D, the author does not want to change the duration of the objects but only the time position of D.

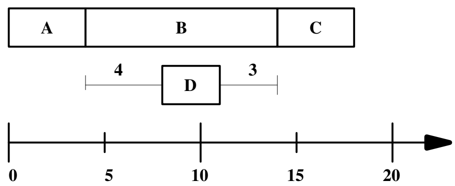


Figure 3: Anticipation

If the author selects the object D, he can move it from four units in the left direction and three units in the right direction. This information results from the analysis of the temporal relations between the objects and is complex to calculate even for simple situations. So, it is important that constraint solvers help authoring systems in presenting anticipation information to the author.

3.4 Completeness

The system must allow all possible changes as defined by the space of solutions of the constraint network. In other words, it must not prevent the author to reach a solution if it is a possible solution of the constraint network.

3.5 Finding the best solution

The basic need for an authoring environment is to obtain a solution (when it exists) for the set of relations defined by the author. This solution is used to spatially display the document in the execution view and to play it time over.

The document defined by the author can produce a wide set of solutions and the system must choose one solution among this set. In fact most of the time, the set of constraints is not precise enough for the constraint system to find the solution the closest to the solution that the author want. Moreover, the author builds his presentation step by step. During a large

period of this process (and possibly even when the authoring is finished) the document is extremely under-constrained.
To compute a new solution we must guide the solver to find a good solution. For example, when the author defines a sequence of 10s for 2 objects, where each object is defined in a range of [1..10], it would be useful if the system chooses 5s for each object. The system could also choose the *best* value according to the intrinsic duration of each media. So, if the first object is a video with duration of 6s, the system could choose 6s for the video and 4s for the other object.
There are two ways to guide the constraint solver. The first one uses hierarchical constraints, the second one uses heuristic functions. The choice between the two methods depends on the class of the solver used (see section 5).
Below, we define a distance function to evaluate the quality of the solution and we present then the two ways to guide the constraint solver.

3.5.1. Distance function

The quality of a solution S_2, obtained after an authoring action applied on the object *O*, can be measured by evaluating its distance from the previous one (called S_1).
The first parameter (called P1) that can be taken into account is the *distance* between the object *O* and the other objects. Intuitively, the objective is to choose local changes in priority. Such a distance clearly depends on the set of constraints between the objects. So we define the distance between two variables as the minimal number of constraints (direct or indirect relation) between the two variables in the constraint network. For example, if we have the following five constraints:
(A > B + C), (C > D), (B = E + F), (E = G), (G = D)
the distance between the variables A and D is: distance(A,D) = min(2, 5) = 2.
Other parameters can be chosen to better reflect the distance between two solutions:
- P2: the standard variation of the variable values of the objects.
- P3: the number of modified objects.
- P4: the number of modified objects weighted by the type of each object. Indeed, this parameter allows choosing solutions in which the change of duration is preferred for a text that for a video or a sound.

Finally the distance function is a combination of these parameters, which means that the distance function F is the following:

$$F(S_1, S_2) = aP1 + bP2 + cP3 + dP4$$

where a, b, c and d are such as $a + b + c + d = 1$.
This function has been adjusted (evaluating a, b, c and d) to cover the needs of the Madeus environment and correspond so to a particular context. It is general enough to be adapted to other contexts by simply changing this valuation.
In this paper we also use this function to evaluate the different studied solvers (see section 6).

3.5.2. Hierarchical constraints

In this technique, the idea is to add specific constraints with weights (Borning et al, 87) to the current constraint network in order to guide the research of a pertinent solution. These constraints can be processed by the authoring system in order to take into account external information such as the intrinsic duration of a media. Each authoring step requires not only to add/change/remove variables or constraints related to author operations (see section 2.3) but also to manage these specific added constraints.

3.5.3. Heuristic function

The system can use heuristic functions at two different steps of the solving process: the first one to choose the order in which the variables will be evaluated, and the second one to choose a value for the variables.

For instance, the first function can sort the variables according to the type of the object (text, sound, and video…) and the number of relations in which they are directly or indirectly implied.

The second function can choose a value for the variables according to the current value of Min and Max and to the previous value of Current.

4. Choosing a compromise

Due to previous experiences with constraint solvers in the context of authoring multimedia (Carcone et al, 97; Brados, 98) we know that we cannot have all the requirements fulfilled at the same time. So we must do some compromises. In this part, we describe different compromises that can be taken, and we explain the different choices according to the objectives.

4.1 ... between time performances and the pertinence of the solution

We need to find the best solution of the constraint network. This problem is time-complex. To reduce the time, we can handle a relaxed definition of the best solution and therefore allow the system to find a good solution, but not necessarily the best one. This solution can be reached by using heuristic functions in global techniques or by orienting the evaluation in local techniques such as in Deltablue (see section 5).

4.2 … between time performances and direct authoring facilities

One way to obtain better performances is to introduce some restrictions on author actions. For instance in the spatial dimension, one of the most significant gain in time is obtained by preventing the author to change the size of a composite through the resizing of one of its children. The author can change the size of a composite by a direct action on it or by an action of one of its neighbours in the spatial structure.

A small example is shown in Figure 4: the author cannot move the object B vertically, because this object defines the bounding box of the composite but he can move it horizontally. For the same reason he can move the C object vertically but not horizontally. The object D cannot be moved at all because it defines two edges of the bounding box, a horizontal one, and a vertical one.

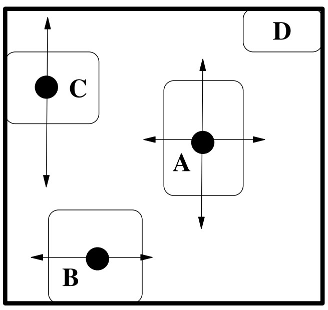


Figure 4: Spatial authoring action on objects inside a composite

The author can use indirect authoring actions to scale the composite (changing its corresponding attributes of width or height) or he can explicitly select the composite and scale it. So we do not prevent the author to access any spatial position.
Maintaining the minimal bounding box is not easily achievable by linear solvers. There are many ways to cover this problem. The first way is to have a more flexible definition of a composite: it is defined by the bounding box of all the objects inside it, but not necessarily the minimal one. The second way is to prevent the scaling of a composite. And the third way is to maintain the minimal bounding box of the object as far as possible.

4.3 … between time performances and expressive power

We can enhance time performances in constraint management by slightly reducing the set of user specified relations. Some local propagation solvers are time efficient, but are not able to handle cyclic constraint networks (see section 5.3). It is the reason why the temporal *center* relation, which could be useful, is usually suppressed from the set of temporal relation available. So we can use more specific algorithms, which does not handle this relation.

5. Constraint solvers

In previous parts we have precisely described the requirements needed in constraint techniques. We present here different constraint solvers we have studied and experimented in our Madeus system. The aim of this study is not to compare all existing solvers, but to have a base of comparison between different constraint techniques. The choice of the solvers has been done through two criteria:

- The representativity of the principal implemented techniques.
- The implementation language (Java language) of the solver, to integrate it more easily in our system.

We have experimented five different solvers and have implemented two other ones in our authoring environment.

5.1 The DCSP context

Because of the incremental nature of authoring actions performed by the author, the system has to solve a succession of Constraint Satisfaction Problems (CSP). The problem of checking the consistency of the constraint network and finding a solution in this context is part of the Dynamic Constraint Satisfaction Problem (DCSP) class introduced in (Dechter and Dechter, 88). In this approach the different CSP form a sequence, each CSP differs from the previous one from an addition or a removal of a constraint. The principle of the DCSP is to avoid considering each CSP as new one, but instead, to reuse the results obtained during previous step (like deducted constraints, values of precedent solution). In fact it takes advantages of the properties associated to the addition or the removal operations: removing a constraint preserves the current solution and adding a constraint preserves deducted constraints (for example the filtered minimal and maximal values). The different techniques to deal with DCSP can be grouped in three families, as described in (Verfaillie and Schiex, 95). They are:

1. Memorization and reusing of solutions
2. Memorization and reusing of constraints
3. Memorization and reusing of both solutions and constraints

In practice, few of these methods are integrated in existing solvers, even if there are various experimentations of DCSP. This is the reason why most of the tested solvers are not DCSP solvers.

Next sections present global solvers, then local solvers and finally the two solvers we have developed in our environment.

5.2 Global solvers

We are not going to describe the main categories of global solvers with backtrack mechanism and so on… but just three algorithms or software we have experimented in our toolkit. The first one is Cassowary (Brados, 98) a constraint solver of the Washington university. The second one is jSolver (Andy Hon Wai H.A., 99) and the last one is Maple (Maple, 98).

5.2.1. Cassowary

Cassowary is a solver for linear arithmetic constraints based on the simplex method to solve the constraint network. To find a solution Cassowary uses a locally-error-better or a weight-sum-better comparator approach. We use Cassowary because it seems interesting in the case of the spatial placement of objects and it is used elsewhere for this purpose (Brados, 98).

5.2.2. jSolver

jSolver is a Java library to handle CSP problems. All variables are defined in domains. The jSolver search algorithm is performed in the four following steps.

1- take a variable in the set of variables to instanciate
2- select a value for the variable from the domain: possible backtracks may occur
3- assign value to the variable
4- propagate constraints

The jSolver implementation allows us to define different strategies for the choice of the next variable to process and for the choice of its value (as defined in section 3.5.3). As for local solvers like Deltablue (see section 5.3), jSolver provides a way for working out a Plan or a Goal and for keeping it during an author action such as object displacement.

5.2.3. Maple

Maple (Maple, 98) is a special case in our study. It has been considered as a reference solver for computing best solutions. Moreover this solver is complete and allows us to handle some non-linear functions. So, we also use it to work out pre-conditions for other solvers (like the minimal bounding box) or to compute the first solution of the constraint network.

5.2.4. An enumerative solver

As all our variables are defined as numerical intervals, it is natural to think of using enumerative solvers. One of the problems here is to handle the infinite bound for the max value of the intervals. In fact multimedia documents introduce such values for non-deterministic temporal objects. So it is necessary to set arbitrary bound for such intervals.
Another problem is to convert each user-defined relation into a set of equations.
For example a Before relation between two objects A and B, which both have a duration between the interval [540.000ms 600.000ms] (between 9 and 10 minutes) introduces 60000 * 60001 /2 equations. This huge number of resulting relations for simple cases led us to abandon considering this approach.

5.3 Local solvers

The second category of solvers we have experimented is local propagation solvers. These solvers seem to be interesting because they do not handle all the set of constraints (at each time). When the author selects an object and moves it, the solver only manages the set of useful during all that displacement. In the Deltablue solver (Sannela et al, 93) for example it is done by the use of *plan*. One of the problems in this approach is the incompleteness of the method and the impossibility to handle cycles in the constraint graph. In the authoring phase we can't remove cycles. For example, if we say that two objects A and B are left aligned and right aligned, we have a cycle in the constraint graph, due to the width relation. If we suppress this relation during the authoring phase then when the author scales the document by changing its width, the relation will not be maintained. There is a way to avoid this problem: for each authoring action we can compute all the constraints which will be implicitly maintained and we can remove them from the constraint graph. But it is a complex task and doing it at each authoring step is not time-efficient.

The hierarchical organisation of the document raised another difficulty when using local solvers. The *inside* relations, which are automatically added for each object that is included into another one, introduce cycles in the constraint graph. One way to avoid this problem is to prevent the scaling of a composite object when scaling or moving the objects that compound it. Notice that this operation can be replaced by another authoring action directly performed on the composite (as discussed in section 4.2).

The last drawback of local solvers results from their two step process: plan computation and constraint solving. Indeed, given a set of constraints, a set of variables and an authoring action, the same solution is always computed. In an authoring context different results need sometimes to be provided. For example, when we have 3 objects in sequence, we move the right one, the middle object first scales, but to prevent it from becoming too small, the right object must also be scaled. We can also prefer the modification to be applied on the two objects with equivalent scaling.

5.4 Specific solvers developed in our system

To take advantages of Madeus internal structure, which is basically a temporal graph (as described in section 5.4.1), we have designed two specific algorithms. The first one is an adaptation of the path consistency algorithm PC2 to check the temporal consistency of a temporal graph and to compute a solution (Bes, 98). The other one is an algorithm to handle temporal displacements and scaling of temporal objects in the scenario view (Tardif, 97).

5.4.1. PC2 algorithm

The set of the temporal constraints can be directly translated into an oriented and acyclic graph. We call it the temporal graph. The Figure 5 shows the temporal graph of the scenario: A [1,2,4], B[3,5,9], C[2,3,5] with relations Starts A B, Meets A C, Finishes C B. The nodes of this graph represent the start and the end instants of the objects. An arc represents a duration constraint between the two instants represented by the two corresponding nodes: a continuous interval of duration is associated to each arc. This kind of problem corresponds to a well-known class of CSP: the Simple Temporal Problems (STP) described in (Dechter et al, 91).

To check the consistency of such graphs, several algorithms have been proposed. We can quote AC3 (Mackworth, 77), AC4 (Mohr and Henderson, 86), DPC (Dechter et al, 88) or AC6 (Bessière and Cordier, 94). They are based on the notion of arc-consistency or path-consistency. We use an algorithm based on path-consistency called PC2 (Mackworth, 77).

PC2 is an algorithm that filters out from the interval of duration associated to the arcs all the values that do not belong to a solution. To do that, it considers all the possible two arc long paths between all the couples of nodes of the graph. An inconsistent state is detected when all the values of an interval have been filtered out. The algorithm can be decomposed into the following steps:

– The completion of the graph to link each node to all the others (to permit to compute all the possible two arcs long paths between all the couples of nodes). Initially, the new arcs have an infinite duration [-∞,+∞].
– For each couple of nodes:
    - The duration of each possible two arcs long path between these two nodes is computed.
    - The intersection of all these durations and the duration of the arc linking the two nodes is computed.
    - If this intersection is empty, an inconsistency is detected. Otherwise, this value is taken as the new value for the duration of the arc linking the two nodes.

The second step is repeated until the stability of each duration value is obtained.

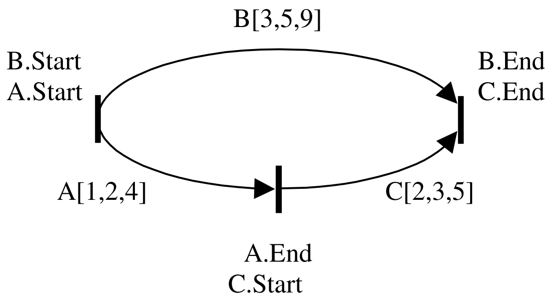


Figure 5: A temporal graph

The result of this algorithm is the minimal graph for the given constraint network. In the frame of the STP, this kind of graph has the property to be decomposable: choosing a value in any interval of duration, the system can find a solution for the rest of the variables without backtracking. This property allows us to use PC2 to compute a solution by successive filtering operations and assignments.
Adding a constraint is straightforward because of the temporal graph construction. Constraint removal is more difficult because some values of the domains that have been filtered must be re-introduced. Therefore in the current implementation of PC2, each removal implies to filter again the graph from the beginning state. Notice that solutions to reduce the re-introduction of values in the domains have been studied and tested (mainly on AC3 and AC4 (Bessière, 91)), and can be implemented in PC2 implementation.

5.4.2. Adaptation of PC2 for composites (HPC2)

With the composite objects, the temporal dimension of a document is not only represented by a temporal graph but also by a hierarchy of temporal graphs. An arc in a graph of a given level of the hierarchy can represent a graph of the lower level. So the propagation of a constraint occurs in the graph where the constraint is declared, and also in all the temporal graphs up or down through the levels. Figure 6 gives the algorithm called HPC2 we propose for handling such situation (Bes, 98).
The difficulty is the same than in spatial dimension to ensure the minimal duration bounding box. But contrary to spatial dimension we can take advantages of the minimal domains computed by PC2 to reduce the propagation between the graphs. In fact, starting with the current consistent temporal graphs (line 1), adding a constraint in a given level just triggers an execution of PC2 in the concerned level to check the consistency of all the constraint

system. Only one bottom-up propagation (line 10) and one top-down propagation (line 8-9) are needed to reduce the durations of the other arcs in all the other levels (graphs), without considering again the starting graph.
On the other hand, constraint removal and constraint relaxation force the solver to reconsider all the constraint system (line 3-6), for the same reason as PC2. In this case, the consistency of the graph hierarchy must be checked from the bottom to the top (line 4), with a final reduction from the top to the bottom (line 5-6).
The properties of minimality and decomposability of the temporal graphs permit to reduce the number of propagations through the hierarchy.

```
   HPC2(Graph G) {
1      If (φ(G))
2           CheckConsistency(G);
3      Else {
4           ReduceBottomUpFirst(G);
5           For each G1 ∈ Ω(G)
6                ReduceTopDown(G1);
       }
   }

   CheckConsistency(Graph G) {
7      PC2(G);
8      For each G1 ∈ Ω(G)
9           ReduceTopDown(G1)
10     CheckConsistency(Parent(G))
   }

   ReduceBottomUpFirst(Graph G) {
11     For each G1 ∈ Ω(G)
12          ReduceBottomUpFirst(G1)
13     PC2(G1)
   }

   ReduceTopDown(Graph G) {
14     PC2(G)
15     For each G1 ∈ Ω(G)
16          ReduceTopDown(G1)
   }
```

- φ(G)=true if G is consistent and if a new constraint has been added to the solver.
- Ω(G)= {Graphs represented by an arc in G for which the duration has been modified}

Figure 6: HPC2 algorithm

5.4.3. The TimeLine Algorithm (TLA)

The timeline view, as presented before, presents a solution of the temporal constraint network, and enables the author to navigate in the solution space. The TLA algorithm aims at providing a new solution each time an authoring action is performed in this view. The algorithm presented here requires that the intervals are minimal. Before applying the TLA algorithm, another algorithm has to compute a solution to the set of constraints, in order to reduce all the intervals of variables to their minimum value. With this hypothesis we can ensure that if the author chooses any value in any minimal interval, there exists a solution to the set of constraints.
The algorithm developed for the timeline view is a two step algorithm:

1. Collecting information: according to a set of constraints and an action (such as moving or resizing the object) the solver compute the maximal interval for the displacement (or resizing) of that object. It also computes the list of objects that will be modified by this action.
2. Applying the user action: each time the author applies an elementary action (for example at each step of a displacement) the system uses the previous computed information to apply the change. We can compare this approach with the *plan* of Deltablue.

The method is based on the temporal graph of the document, and not on the constraint network. The system uses a propagation method from the source of the modification (the selected object) to the intervals which are able to absorb this deformation.

6. Results of constraint solver performances

All following results have been obtained on a Pentium II, 400 MHz

6.1 Description of the tests

During these tests we focus on two points: the time to compute a solution and the quality of this solution. That is why for each experience we store the time to compute the solution and we compute the distance from the new solution to the previous one.
To evaluate scalability of the solvers we have performed the tests on two groups of documents that have 100 objects (see Table 5) and documents of 1000 objects (see Table 6).
Each session consists of the following steps :

- Opening the document, which measures the time taken by the solver to compute a solution from scratch, starting from a set of constraints and a set of variables.
- Authoring :
  - Adding and deleting an object.
  - Adding and deleting a (spatial and temporal) relation.
  - Changing an (spatial or temporal) attribute, which means moving an object (in time or space) or scaling it.

For every solver, we use either the heuristic function or hierarchical constraints as defined in section 3.5 to guide the new solution.
For these experiments, we use two special solvers: the *Maple solver* that serves as the reference solver for the quality of the solution and the *null solver* that gives the time taken by the authoring environment to compute and to take into account an authoring action.
The symbol ☹ is used when the solver was unable to find a solution or when it crashes because of an out-of-memory error. The symbol *No* means that the solver is not usable for the considered authoring action. Table 7 shows the qualitative results in terms of distance between the solution given by the solver and the solution provided by Maple.

Solver		**Open**	**Authoring**									
			Object		Relation				Attribute			
			Add	Delete	Spatial		Temporal		Spatial		Temporal	
					Add	Del	Add	Del	move	Scale	move	Scale
Global	Cassowary	4.4s	0.4s	0.2s	0.6s	0.3s	0.4s	0.2s	0.5s	0.5s	0.4s	0.4s
	jSolver	4.5s	0.5s	4.5s	1.2s	4.5s	1s	4.5s	4.5s	4.5s	4.5s	4.5s
	Maple	18m	18m	18m	18m	18m	18m	18m	18m	18m	18m	18m
Local	Deltablue	4.5s	0.2s	0.1s	0.5s	0.3s	0.4s	0.3s	0.1s	☹	0.1s	☹
Null Solver		3s	0.1s	0.1s	0.1s	0.1s	0.1s	0.1s	0.05s	0.05s	0.05s	0.05s
Our solvers	HPC2	20.2s	0.15s	0.15	No	No	1s	20.2s	No	No	12s	12s
	TLA	No	No	No	No	No	No	No	No	No	0.1s	0.1s

Table 5: Quantitative results for 100 objects documents

Solver		Open	Authoring									
			Object		Relation				Attribute			
			Add	Delete	Spatial		Temporal		Spatial		Temporal	
					Add	Del	Add	Del	Move	scale	move	Scale
Global	Cassowary	192s	0.7s	0.2s	5s	0.3s	4s	0.2s	3.5s	3.5s	2.3s	2.3s
	jSolver	☹	☹	☹	☹	☹	☹	☹	☹	☹	☹	☹
	Maple	> 1h	> 1h	> 1h	> 1h	> 1h	> 1h	> 1h	> 1h	> 1h	> 1h	> 1h
Local	Deltablue	11.4s	0.4s	0.2s	1s	0.3s	1s	0.3s	0.3s	☹	0.3s	☹
Null Solver		8s	0.1s	0.1s	0.1s	0.1s	0.1s	0.1s	0.05s	0.05s	0.05s	0.05s
Our solvers	HPC2	☹	☹	☹	☹	☹	☹	☹	☹	☹	☹	☹
	TLA	No	No	No	No	No	No	No	No	No	0.1s	0.1s

Table 6: Quantitative results for 1000 objects documents

6.2 Analysis of the results

The quantitative and qualitative performances show that the results depend on the authoring actions. In fact these ones reveal the intrinsic properties of the solvers. For example good time performances are obtained by HPC2 when adding relations because it is able to maintain minimal domains. Whereas it has bad performances when removing relations due to the fact that HPC2 compute the whole constraint network again. We can observe the same difference for jSolver for the same reasons.

The best solver in time is Deltablue, but due to the limits we have described in section 5.3, the solver that better fulfils the two criteria (quality and time performance) is Cassowary.

The results concerning the HPC2 show its limits in term of the number of objects supported. This is due to its need of handling a complete graph.

Solver		Open	Authoring									
			Object		Relation				Attribute			
			Add	Delete	Spatial		Temporal		Spatial		Temporal	
					Add	Del	Add	Del	move	scale	move	Scale
Global	Cassowary	0.1	0	0	0.1	0	0.1	0	0.05	0.05	0.05	0.05
	jSolver	0.1	0	0	0.05	☹	0.05	☹	0	0	0	0
	Maple	0	0	0	0	0	0	0	0	0	0	0
Local	Deltablue	0.15	0	0	0.3	0	0.3	0	0.4	☹	0.4	☹
Our solvers	HPC2	0.1	0	0	No	No	0	0	No	No	0	0
	TLA	No	No	No	No	No	No	No	No	No	0.05	0.05

Table 7: Qualitative results for 1000 objects

The TLA solver needs minimal domains to be able to compute new solutions. This computation is complex and time consuming. Thus, this algorithm can not be used when the author adds/removes objects or relations. But this algorithm is very powerful to compute moving and scaling modifications (with the restriction described above). Thus, it is very useful for the timeline view, which is an exploration view for understanding the temporal behaviour of the document.

An interesting aspect that we can notice is that the most time efficient solvers are those providing the less satisfying solution from the qualitative point of view. In fact, these solvers are lacking some means that allow the system to guide them to the preferred solution.

7. Toward a polymorph algorithm

The previous results show that the best algorithm depends on the authoring action. From this analysis, we decided to combine several solvers in order to use the best one for each authoring action. The *polymorph algorithm* we use is a combination of Deltablue and Cassowary. We use Deltablue for opening documents and for adding objects. We choose Cassowary for all the other authoring actions because it is less restrictive than Deltablue. The use of Deltablue during the opening step is not restrictive: during this step it is not important to handle all the constraints, we can remove the constraints which induce redundant information (Carcone, 97).
As shown in Table 8, the polymorph algorithm is not the best in time performances. This is due to the fact that the time to take into account an authoring action is composed in two parts: the time for building information for the solver, and the time to compute the new solution. If we use the two combined algorithms, the time for building information to the solver is equal to the time for building this information for each solver, but the polymorph algorithm is the best in term of quality/time.

Polymorph algorithm	**Open**	**Authoring**									
		Object		Relation				Attribute			
		Add	Del	Spatial		Temporal		Spatial		Temporal	
				Add	Del	Add	Del	move	scale	move	Scale
100 objects document	5.5s	0.3s	0.2s	0.7s	0.4s	0.5s	0.3s	0.5s	0.5s	0.4s	0.4s
1000 objects document	45s	0.6s	0.3s	2s	0.4s	1.7s	0.4s	3.7s	3.7s	2.4s	2.5s

Table 8: Results with polymorph algorithm

8. Conclusion

In this paper, we presented an authoring environment called Madeus, which is based on constraint techniques. We outlined some of the requirements in this framework that allow more powerful editing actions. We have described the different experimented solvers and the quantitative and qualitative results we have obtained in our experiments. We have also described a first work in the direction of a polymorph algorithm to handle the complete authoring process of multimedia documents. This work has also been tested in Smil-Editor, an authoring environment for SMIL documents.
Our conclusion on this work is that constraint techniques are useful to build such authoring environments, but they must integrate a better control on the obtained solutions. The work will be pursued in two axes: the first one is the adaptability of documents where constraint techniques could be efficient thanks to their flexibility, and the second one is the improvement of methods to handle hierarchical bounding boxes.

9. References

(Allen, 83) Allen J. F., "Maintaining knowledge about temporal intervals", Communication of the ACM, Vol. 26(11) pp. 8332-843, November 1983.

(Andy Hon Wai, 99) Andy Hon Wai C., "Constraint programming in Java with jSolver", Proceedings of PACLP99, The practical Application of Constraints Technologies and Logic Programming, London, April 1999.

(Berlandier and Neveu, 94) Berlandier P. and Neveu B., "Maintaining Arc Consistency through Constraint Retraction", *In: Proc. Of the 6th IEEE International Conference on Tools with Artificial Intelligence (TAI94)*, New Orleans, LA, 1994.

(Bes, 98) Bes F., "Spécification hiérarchique de scénario temporel à base de contraintes", Rapport de D.E.A. informatique, Joseph Fourier University of Grenoble, 1998.

(Bessière, 91) Bessière C., "Arc-consistency in Dynamic Constraint Satisfaction Problems", *Proc. Of AAAI-91*, pp. 221-226, Anaheim, CA, 1991.

(Bessière and Cordier, 94) Bessière C. and Cordier M-O., "Arc consistency and arc consistency again", *Artificial Intelligence*, 65, pp. 179-190, 1994.

(Borning et al, 87) Borning A., Duisberg R., Freeman-Benson B.,Kramer A. and Woolf M., "Constraint hierarchies", *In OOPSLA'87 Conference Proceedings*, ACM, pp. 48-60, New-York, 1987.

(Brados and Borning, 98) Brados Greg J. and Borning A., "The Cassowary Linear Arithmetic Constraint Solving Algorithm: Interface and Implementation", Technical Report UW-CSE-98-06-04, June 1998

(Carcone et al, 97) Carcone L., Jourdan M. and Roisin C., "Présentation de documents multimédia basée sur les contraintes", Workshop on Electronic Page Models, LAMPE, September 22-23, 1997.

(Chvatal, 83) Chvatal, "Linear Programming", W.H. Freeman and Company, New York, 1983.

(Dechter and Dechter, 88) Dechter R. and Dechter A., "Belief Maintenance in Dynamic Constraint Networks", *In: Proc. of AAAI-88*, pp. 37-42, St Paul, MN, 1988.

(Dechter et al, 91) Dechter R., Meiri I. and Pearl J., "Temporal constraint networks", *Artificial Intelligence*, 49, pp. 61-95, 1991.

(Jourdan et al, 98a) Jourdan M., Layaïda N., Roisin C., "A survey on authoring techniques for temporal scenarios of multimedia documents", *vol. Handbook of Internet and Multimedia Systems and Applications, part 1: Tools and Standards*, CRC Press, April 1998.

(Jourdan et al, 98b) Jourdan M., Layaïda N., Roisin C, Sabry-Ismail L., Tardif L., "Madeus, an Authoring Environment for Interactive Multimedia Documents", *ACM Multimedia'98*, pp. 267-272, ACM, Bristol (UK), September 1998.

(Jourdan et al, 2000) Jourdan M., Roisin C. and Tardif L., "A Scalable toolkit for desining Multimedia Authoring Environments", s*pecial number "Multimedia Authoring and Presentation: Strategies, Tools, and Experiences" of Multimedia Tools and Applications journal*, Kluwer Academic Publishers, 2000.

(Hsu W.and Teik Guan, 99) Hsu W.and Teik Guan T., "Rapid Prototyping with Constraint-based Scheduling for Multilmedia Application", *Multimedia Tools and Applications*,8, pp. 175-195, 1999.

(Layaïda, 97) Layaïda N., “Madeus: système d'édition et de présentation de documents structurés multimédia”, Doctorat Informatique, Joseph Fourier University of Grenoble, 1997.

(Mackworth, 77) Mackworth A.K., “Consistency in Networks of Relations”, *Artificial Intelligence*, 8 (1), pp. 99-118, 1977.

(Maple, 98) Maple, Reference Manual, 1998

(Mohr and Henderson, 86) Mohr R. and Henderson T., “Arc and path consistency revisited”, *Artificial Intelligence*, 28, pp. 225-233, 1986.

(Sannela et al, 93) Sannela M., Maloney J., Freeman-Benson J. and Borning A., “Multi-way versus One-way constraints in user interfaces: Experience with the DelatBlue algorithm”, Software-Pratice and Experience, Vol. 32(5), pp529-566, May 1993.

(Song et al, 99) Song J., Ramalingham G., Miller R. and Yi B.-K.,”Interactive authoring of multimedia documents in a constraint-based authoring system”, *Multimedia Systems*, Springer-Verlag, volume 7, pp. 424-437, 1999.

(Tardif, 97) Tardif L., “Visualisation du scénario temporel d'un document multimédia”, Rapport de D.E.A. informatique, Joseph Fourier University of Grenoble, 1997.

(Verfaillie and Schiex, 95) Verfaillie G. and Schiex T., “Maintien de solution dans les problèmes dynamiques de satisfaction de contraintes: bilan de quelques approches”, *Revue d’intelligence artificielle*, volume 9, n°3, pp. 269-309, 1995.