# View Management in Multimedia Databases*

**K. Selçuk Candan** † **Eric Lemar**‡ **V.S. Subrahmanian**‡

**Abstract**

Though there has been extensive work on multimedia databases in the last few years, there is no prevailing notion of a multimedia view, nor there are techniques to create, manage, and maintain such views. Visualizing the results of a dynamic multimedia query or materializing a dynamic multimedia view corresponds to assembling and delivering an interactive multimedia presentation in accordance with the visualization specifications. In this paper, we suggest that a non-interactive multimedia presentation is a set of *virtual objects* with associated spatial and temporal presentation constraints. A virtual object is either an object, or the result of a query. As queries may have different answers at different points in time, scheduling the presentation of such objects is nontrivial. We then develop a probabilistic model of interactive multimedia presentations, extending the non-interactive model described earlier. We also develop a probabilistic model of interactive visualization where the probabilities reflect the user profiles, or the likelihood of certain user interactions. Based on this probabilistic model, we develop three utility-theoretic based types of prefetching algorithms that anticipate how users will interact with the presentation. These prefetching algorithms allow efficient visualization of the query results in accordance with the underlying specification. We have built a prototype system that incorporates these algorithms. We report on the results of experiments conducted on top of this implementation.

## 1 Introduction

Over the last few years, there has been considerable interest in multimedia databases [64] – most such research has been devoted to content management[60, 52, 32], design of hierarchical media storage servers [54, 55, 44], and networked media servers [59, 61]. Excellent work on querying media databases has been done – ranging from querying image data[53, 48, 8, 32], video data[53, 2], audio data[69] and mixes of all the above[46].

Since, multimedia objects have temporal and spatial aspects that do not exist in more traditional data objects, visualization of the results of a multimedia query requires specification of the visualization

*This research was supported by the Army Research Office under grant DAAH-04-95-10174, by the Air Force Office of Scientific Research under grant F49620-93-1-0065, by ARPA/Rome Labs contract Nr. F30602-93-C-0241 (Order Nr. A716), and by an NSF Young Investigator award IRI-9-57756.

†Contact author: Computer Science and Engineering Department, Arizona State University, Tempe, AZ 85287-5406. Phone: (602) 965-2770. Fax: (602) 965-5142. Email: candan@asu.edu

‡Department of Computer Science, Institute for Advanced Computer Studies & Institute for Systems Research, University of Maryland, College Park, Maryland 20742. Phone: (301) 405-2711. Fax: (301) 405-6707 Email: {elemar, vs}@cs.umd.edu.

parameters along with the query. Consequently, multimedia databases are starting to integrate query and presentation specification languages. Examples of such frameworks include [9], [45, 46] and [51]. In [45, 46], Marcus and Subrahmanian introduce a multimedia system which presents the answers to queries not in terms of traditional tables, but in terms of multimedia presentations, where the structure of the presentation reflects the preferences of the user as well as the semantics of the query. [9] proposes a language to specify visualization specifications along with the query (SQL+D). In [51], Özsoyoglu and his colleagues discuss a system called VIDQUERY which assembles a presentation based on a given query and a set of inclusion exclusion constraints. However, none of these papers specify how they solve the following technical challenges:

- *Interactive Presentations.* The multimedia presentation generated by a query must support interaction between the user and the presentation. None of the above papers support this.
- *Living Presentations.* A living presentation is a multimedia presentation where the content of the presentation dynamically change over time. Query visualization mechanisms assume the query is the "master" object which drives a presentation engine. In contrast, in a living presentation, the presentation is authored once, but the object (and where and when it is) shown in the presentation changes over time with the object-ids and -contents being dynamically determined by queries or updates. None of the above papers support this.
- *Unpredictable number of Answers.* It is not even possible to write constraints (spatial or temporal) on presentations of objects if the number of objects being presented are not known. The above frameworks do not support this.

For example, [9], does not address temporal aspects of query visualized. In contrast, [51] provides a more complete treatment of the temporal aspects of the visualized results, but does not address the spatial and interactive aspects of the result visualization problem. Other related works include DOODLE [28] and Delaunay [29], a visual query language for object-oriented databases and an associated constraint-based language for visualizing objects on the screen. Note that, since the objects treated in this framework are not multimedia objects, DOODLE and Delaunay neither address temporal aspects of the visualization problem nor try to handle the resource requirements associated with presenting multimedia objects.

The blurring of boundaries between querying and interactive visualization is also highlighted by researchers in the computer human interaction (CHI) domain. For instance, VOIR[35] is an information exploration interface that uses a search engine for querying a document database. It then automatically assembles a hypertext document linking the query result into each other using the newspaper metaphor. The links that a user chooses utilized while browsing through the final document are, then, used in expanding the initial query.

Our aim in this paper is to give a complete account of how to deal with building multimedia presentations whose content changes dynamically based upon queries. This also provides a solution to multimedia visualizations of queries to multimedia databases. Below, we provide an example to motivate the concepts introduced in the rest of the paper (we will later provide a more detailed example).

**Example 1.1** Let us assume that a retail store wants to prepare kiosks and Internet homepages which will help customers search for goods in their inventory. However, since each customer has a different personal purchase history (types and quality of goods purchased, average number of money spent per good, number of purchases within the year, amount of money spent within the season), the automated guides prepare *personalized* presentations to the customers in the hope of maximizing sales.

For this purpose, the queries that the automated guide issues to the inventory database must contain personal information (if available) associated with the customers. Hence, for each customer, the result of the query will contain a different set of multimedia objects (images of the goods, videos describing the goods, etc.). Note that each set may contain a different number of multimedia objects, and in addition, corresponding objects in different sets may differ in size and temporal length.

Hence, the automated guide must present different object sets, i.e. query results, to each customer. However, these objects must be presented to the customers in a form that more or less fit a uniform presentation format. Consequently, the database needs a flexible result visualization specification mechanism.

Let us also assume that the retail store wants to let its users interact with the kiosk while viewing the query results. For this purpose, the result visualization system as well as the multimedia database must provide dynamic features that enable user interaction. □

**Challenges.** The specific challenges associated with the above example can be stated as follows:

- *Traditional presentation techniques are not suitable or desirable for multimedia databases.* Generally, traditional databases present answers to a query in tables. Due to the spatial and temporal characteristics of multimedia objects, however, this mechanism is not suitable for presenting answers to a multimedia query unless all the results are converted into a uniform representation. We have already proposed two mechanisms called Video Mosaic (VIMOS) [22] and SEMCOG/IFQ [24, 25] for this purpose. In this paper, however, we are proposing a result presentation framework which maintains the spatial and temporal characteristics of the objects, and present the information within an interactive scenario.

- *Given a multimedia query, the number of relevant results is not known in advance.* In some cases, users may specify an upper bound on the number of results; however, the actual number of results may be less than the upper-bound. Consequently, the interactive scenario must be flexible to accommodate varying number of results.

- *Temporal and spatial sizes of the objects are not known.* Since a multimedia database may store objects with different characteristics, at the querying time the user may not know the sizes and temporal lengths of the objects that the system will return. Consequently, specification mechanisms which express only predetermined object sizes and lengths are not suitable for specifying how query results should be presented to the users.

- *Resource requirements of the objects are large and they cannot be known in advance.* Since it is not known which objects will be returned by a query, the amount of system resources required to present these objects is not known a priori. Note that in traditional databases, since the resource requirements of the textual and numeric results are not great, this fact did not cause any challenges.

- *Different objects may have different cost/quality trade-offs.* The results of multimedia queries are usually returned as ranked lists (in contrast to the bags returned by traditional databases). While creating a presentation, this rank and the presentation costs associated with each result must be considered together (note that these two criteria may be in conflict with each other). Furthermore, the visual quality of multimedia objects, such as images, can be degraded to reduce the associated presentation costs. Such degradations have to be taken into consideration while creating result presentations.

- *The set of objects that users will select for viewing are not known.* If the presentation scenario is interactive, it may not be known in advance which objects in the result will be selected for viewing by a particular user.

**Focus of the paper and the contributions.** In this paper, we describe our integrated approach to the above challenges. We build on top of the Collaborative Heterogeneous Interactive Multimedia Platform (CHIMP) [16, 17, 18] developed at the University of Maryland at College Park. The structure of this paper is as follows:

- In Section 2, we provide a brief overview of the flexible visualization mechanism. More specifically, we describe the flexible presentation specification mechanism provided by CHIMP. Note that the collaborative aspects of CHIMP are not relevant to this paper. Instead, we will benefit from the heterogeneous and interactive functionalities of CHIMP.

- In Section 3, we describe how the query results are matched into the presentation scenario. More specifically, we introduce the concept of a multimedia query, we provide the basic constructs for an object assignment and presentation language, and we describe the challenges associated with achieving the required flexibility. The proposed assignment mechanism leads into a spatio-temporally rich visualization framework.

- In Section 4, we describe mechanisms to cope with interactivity in result visualization and the varying resource requirements of the multimedia objects in the result. We develop on the multimedia object assignment language introduced in Section 3, to define the concept of interactive result (scenario) visualization. In this section, we assume the presence of an access, preference, or interaction profile of the user; and, we benefit from this profile and the resource requirements of the objects in maximizing the relevance and quality of the final presentation while making sure that retrieved objects are not overflowing the available resources. The algorithms presented in this section build on top of the previous two bodies of work [16, 17, 18] and [37].

- In Section 5, we discuss implementation issues and the experiment results that validates the algorithms introduced in the paper.

- Finally, in Section 6, we provide our conclusions. The appendices contain proofs of the theorems presented in the paper.

# 2 An Overview of the Flexible Visualization Mechanisms

In this subsection, we will provide an overview of the visualization specification mechanisms. Since the flexibility is more essential and challenging in temporal domain, we focus the discussion on temporal specifications. But, we will also describe, how we uniformly address spatial specifications.

Significant contributions have been made in the area of temporal specification of multimedia presentations. James F. Allen [5, 6] proposed a new formalism based on intervals. In his formalism, Allen differentiates between *properties*, *processes*, and *events*, but describes each of them using time intervals. He also provides thirteen qualitative temporal relationships (*before, meets, overlaps, during, starts, finishes, equal, before*$^{-1}$, *meets*$^{-1}$, *overlaps*$^{-1}$, *during*$^{-1}$, *starts*$^{-1}$, and *finishes*$^{-1}$) that can hold between two intervals, and he shows how these relationships can be used in reasoning about time. Note that, although there are at most thirteen definite relationships among two intervals, the maximum number of indefinite relationships (with disjunctions) is $2^{13}$; hence, computing the relationships among intervals may be a computationally expensive operation. Allen's algorithm is based on a technique called *constraint propagation* which computes all consequences of a new relation using the transitive closure of the temporal relations. Allen proves that his algorithm requires $O(N^2)$ time and space, and he also shows how these requirements can be reduced using reference intervals. Many researchers including [66, 43] use intervals for the scheduling of multimedia documents.

Interval algebra, however, is not the only framework for describing temporal Information. In[67], Vilain and Kautz show that Allen's algorithm is not complete. Furthermore, they show that both finding the closure of assertions in the interval algebra and determining the satisfiability of the assertions are NP-Hard. Interval scripts [57], a methodology proposed to describe user interactions and sensor activities in an interactive system, benefits from a restriction on the allowed disjunction combinations in rendering the problem more manageable [58]. WebPersona [7], a system for automated generation of WWW presentations, on the other hand uses more simple timeline structures. It, then, achieves interactivity using navigation graphs.

Several other researchers, including Candan et.al. [16, 17], Buchanan and Zellweger [12, 13, 14], and Kim and Song [38, 39], on the other hand, proposed the use of a highly-structured class of linear constraints called *difference constraints*, based on the point formalism proposed by Vilain and Kautz [67]. Vilain and Kautz introduce a point based temporal algebra with three definite relationships and eight indefinite relationships between two points in time. They show that this algebra, although more efficient then the interval algebra, represents a large subset of the interval based model. In [10], Peter van Beek shows that the completeness claim of Vilain and Kautz is not correct. He first proves that the algorithm Vilain and Kautz states is not complete in certain cases, and then he provides a sound and complete algorithm for complete point algebra.

One advantage of the difference constraints over intervals is that they can describe quantitative and uncertain information more easily. In [16, 17], Candan et.al further introduces *prioritized differ-*

*ence constraints* and they show how to relax presentation constraints (and modify the presentation in the most desirable manner) when there are conflicts within presentation specifications or there are resource limitations. Creation of retrieval schedules has also been studied by Hwang, Prabhakaran and Subrahmanian [37] who have proposed a method to create retrieval schedules for non-interactive presentations.

In this paper, we benefit from the CHIMP framework introduced by Candan et.al. CHIMP project includes a multimedia document authoring framework [16], a document retrieval and resource management framework [17], and a formal media transmission, conversion, and quality management framework [20, 21]. Below, we provide a brief overview of the system to help the reader follow the discussions presented in the rest of the paper. For more detailed information about CHIMP, we encourage the reader to refer to the mentioned publications.

Within this framework, *each and every temporal and spatial constraint associated with a multimedia document is a difference constraint.* With each object $o \in O_D$ in a multimedia document $D$, we associate a set, $T_o$, of temporal constraints and a set, $S_o$, of spatial constraints. These constraints describe when and where these objects are displayed to the viewers after the presentation starts.

**Temporal Information.** As it is customary in operations research[36], constraints are constructed from *variables*. In the case of multimedia documents, we associate, with each multimedia object $o$ in the document, the following *temporal variables*:

- $st(o)$ : denotes the start time of the display of the object $o$
- $et(o)$ : denotes the end time of the display of the object $o$

Note that using these constraints, not only we can specify Allen's 13 temporal relationships[5, 6] between events, but also specify more complex quantitative relationships that cannot be expressed in Allen's framework.

**Spatial Information.** Spatial constraints are defined for all objects in a multimedia presentation who can be displayed on the screen. In order to specify spatial constraints, we use the following *spatial variables*:

- $w$ and $h$ denote the width and the height of the multimedia document.
- $x_{l,m}$ and $y_{b,m}$ denote the coordinates of the lower left corner of the document on the screen.
- $x_{r,m}$ and $y_{t,m}$ denote the coordinates of the upper right corner of the document on the screen.
- $x_r(o)$ and $x_l(o)$ denote the positions of the right and left borders of the object w.r.t. $x_{l,m}$.
- $y_b(o)$ and $y_t(o)$ denote the positions of the bottom and top borders of the object w.r.t. $y_{b,m}$.

**Difference Constraints.** CHIMP uses a small class of the language of real valued linear constraints. This class of constraints are referred to in the Operations Research literature as *difference constraints*. While generalized linear constraints [36] have the form

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \quad \leq \quad b$$

where $a_1, \ldots, a_n, b$ are *rational numbers* (positive and negative), and $x_1, \ldots, x_n$ range over the *real numbers* (positive and negative), difference constraints have the form

$$x_1 - x_2 \quad \leq \quad b.$$

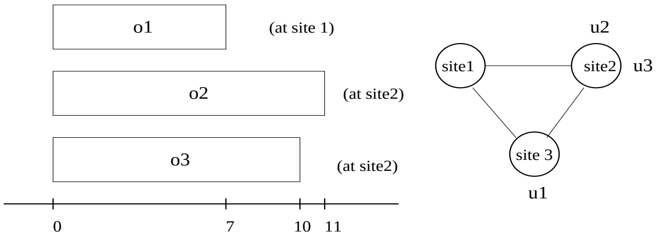



Figure 1: Duration of objects in example multimedia document

The advantage of using difference constraints instead of general constraints is that, there are efficient (polynomial time) algorithms that solve these constraints [27].

**Example 2.1** Let us assume that there exist two objects $o_1$ and $o_2$ that we want to display simultaneously, i.e. we want them to start and finish simultaneously. This requirement can be described using the following constraints:

$$\begin{array}{ll} st(o_1) - st(o_2) \leq 0 & st(o_2) - st(o_1) \leq 0 \\ et(o_1) - et(o_2) \leq 0 & et(o_2) - et(o_1) \leq 0 \end{array}$$

□

**Example 2.2** Figure 1 shows a very simple multimedia document which consists of three multimedia objects: $o1$, $o2$ and $o3$. The multimedia author(s) state the following presentation constraints:

- $o2$ should appear on the screen only after the presentation of $o1$ is completed.
- $o3$ must start at the same time as $o2$.

Now consider three users $u1$, $u2$, and $u3$ with the following facts and requirements:

- It takes 3 seconds for $u1$ to get $o1$ across the network, and it takes 5 seconds for $u1$ to get $o2$ and $o3$.
- $u2$ has immediate access to objects $o2$ and $o3$, but it takes 1 second for him to get $o1$. Moreover, the user $u2$ wishes to start the presentation of object $o3$ within 8 seconds of the beginning of the presentation.
- $u3$ also has immediate access to the objects $o2$ and $o3$, and it takes 1 second to get $o1$. However, $u3$ does not want to see $o2$.

Figure 2 shows some possible realizations of this multimedia document for users $u1, u2$ and $u3$, respectively. It is easy to see that when object $o2$ is omitted from the presentation by $u3$, all related constraints are suppressed, thus leading to alternative renderings of the multimedia document. Hence, the display of $o3$ can overlap with the display of $o1$. □

**Prioritized Difference Constraints.** The major difference between our approach and the other systems which benefit from the difference constraints [13, 39] is that, CHIMP is specifically directed towards flexible multimedia document authoring, where a set of constraints specifying a given multimedia document may be inconsistent due to many reasons:

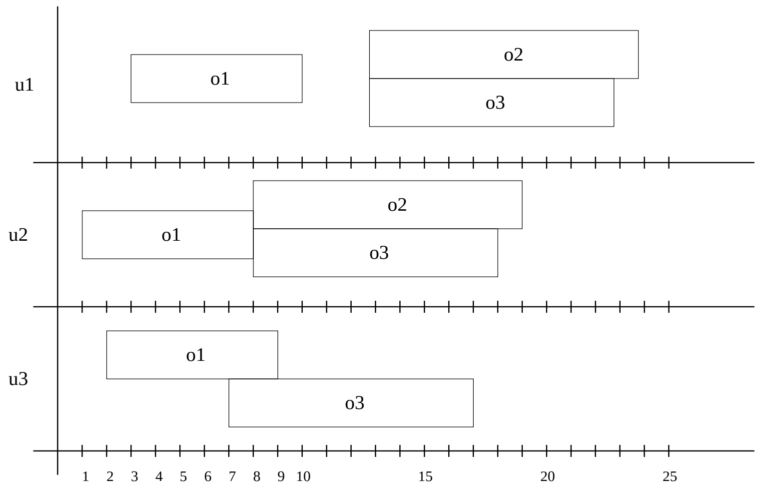



Figure 2: Different presentations of the same document

- different viewers/authors of the document may have different preferences, and such preferences may be inconsistent with each other,
- the constraints introduced by user interactions may be inconsistent with the rest of the constraints, or
- the constraints themselves may be consistent, but they corresponding presentation may be *non-realizable* due to resource limitations.

If such inconsistencies exist, the authoring or the presentation system should (1) warn the user about the existence of the inconsistency, (2) suggest ways to remove the inconsistency, or (3) (if there is not enough time for user feedback or the user is not knowledgeable enough to understand and use the suggestions) take immediate action and remove inconsistencies automatically. To achieve these goals, in [16, 18], we defined prioritized difference constraints:

**Definition 2.1 (Prioritized Difference Constraint)** A prioritized difference constraint is a 2-tuple of the form $< c, p >$, where $c$ is a difference constraint and $p$ is a positive number. $\diamond$

Intuitively, the higher the value of the corresponding priority, the higher the importance of the constraint. Hence, unlike the previous works which assume that a given set of presentation specifications is consistent, we refrain ourselves from making such an assumption. Instead, CHIMP identifies inconsistent requirements, and *optimally* removes the inconsistencies through polynomial time constraint relaxation techniques [18]. In [18], we also presented efficient and incremental algorithms to handle prioritized difference constraints.

In this work, we benefit from the flexibility provided by CHIMP in dealing with the unknowns associated with query result visualization. Although, it is clear that for temporal consistency and synchronization requirements of the visualized results, the flexibility provided by CHIMP is crucial, its benefits of the spatial flexibility may not be as clear. To account for the varying needs of different applications the prioritized difference constraints approach allows the author of the visualization scenario to select the level of temporal and spatial flexibility.

## 3 Multimedia Database Result Visualization

In [16, 18], we discussed how to increase the flexibility of temporal and spatial presentation of multimedia presentations and how to benefit from this flexibility in creating object retrieval schedules in distributed environments. In this section, we discuss how this flexibility allows us to use this framework in multimedia database result visualization. We start with an example query and its result.

**Example 3.1** Let us assume that the multimedia database of a news agency contains a set of maps, aerials views, and textual information regarding various European cities. Let us also assume that, the President of United States gives a speech about American position on Bosnia. During the speech, the news agency, may want to retrieve relevant visual and textual information from their database and present them to the viewers, along with the speech video and proper audio effects.

To achieve this goal, they have two options. Either retrieve all the relevant data from the database and author a multimedia presentation out of it, or use a declarative query language along with a set of *flexible* visualization specifications to have the presentation created automatically (Figure 3).

The advantage of the first approach is that the result will probably be artistically superior to the automatically generated presentation. However, the second approach will be more effective when the time for manually editing the document is not plenty. For instance, during the time-out of a basketball game, the news agency may automatically create a presentation that relates to a particular event in the current game and the similar events and statistics in the past games.

The query-driven, automatic presentation generation will also be effective in digital library applications. □

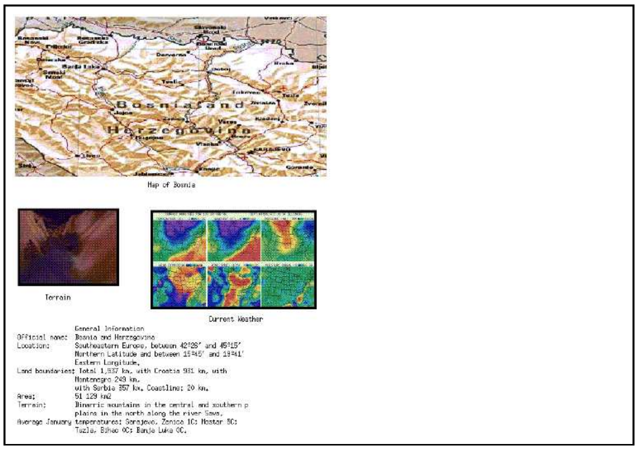
Next, we introduce the concept of a *virtual object*, which is, although simple, a powerful tool for achieving automated result visualization.

**Definition 3.1 (Virtual Object)** A *virtual object* is a polymorphic multimedia object which may take different identities when different conditions hold. ◇

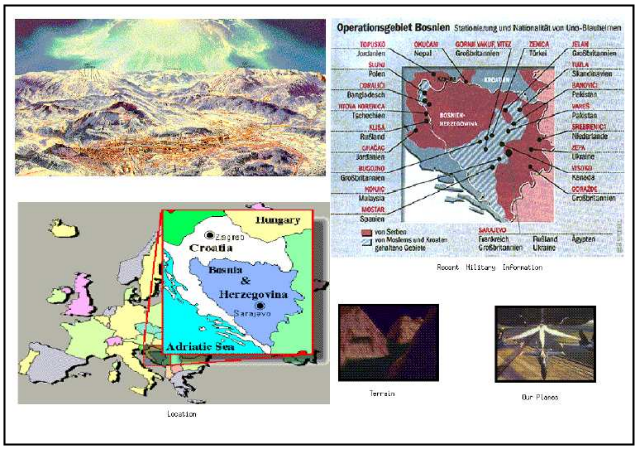
A virtual object is described by its virtual object descriptor.

**Definition 3.2 (Virtual Object Descriptor)** A *virtual object descriptor* is a set, $\mathcal{R} = \{r_1, r_2, \ldots\}$, of triplets of the form $r_i = \langle id_i, con_i, desc_i \rangle$, where $id_i$ describes the multimedia object to be displayed when the condition $con_i$ holds. Note that if there are $n$ rows in the table $\forall_{i \neq j}((con_i \wedge con_j) = false)$ and $\bigvee_i (con_i) = true$. The third value, $desc_i$, on the other hand, describes the additional constraints and presentation specifications (e.g. length, quality constraints) associated with multimedia object $id_i$. ◇

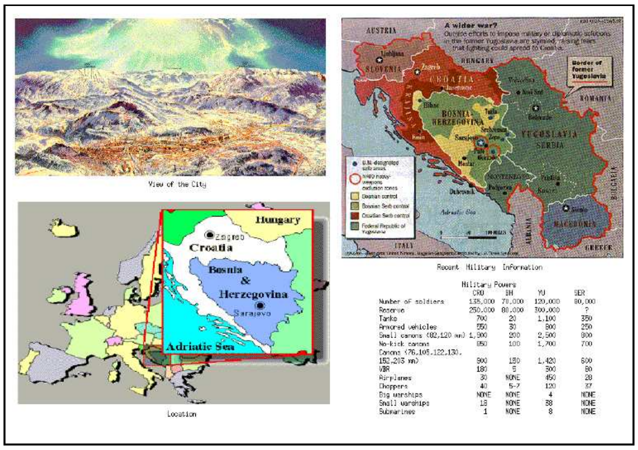
**Example 3.2** Table 1 shows an example virtual object descriptor. In this example, the conditions are based on the user's security clearance level and his preferences. In the first two rows, object identities, i.e. the actual objects to be displayed, are given by the explicit location information. In the third row, however, the location of the object is not given. Hence, if there are alternative sources, the system




(a) $1^{st}$ snapshot




(b) $2^{nd}$ snapshot




(c) $3^{rd}$ snapshot

Figure 3: Three snapshots from the visualized query results for Example 3.1

| Identity | Condition | Description |
|---|---|---|
| `loc`$_1$`:file`$_1$ | `(security` $\geq$ `1) and (preference = 1)` | `length=10` |
| `loc`$_2$`:file`$_2$ | `(security` $\geq$ `1) and (preference = 2)` | |
| `file`$_3$ | `(security` $\geq$ `1) and (preference = 3)` | `(length=10) or (length=20)` |
| $\perp$ | `(security` $<$ `1)` | |

Table 1: An example virtual object descriptor

| Identity | Condition | Description |
|---|---|---|
| $ID$ | $< ID, len > = \text{query}(parameter\_list)$ | `length = `$len$ |

Table 2: Query result visualization example

must find the best place to fetch the object [21]. Note that in the last row, the virtual object is not mapped to a real object, i.e., the virtual object is not to be shown in the presentation. In all cases, the presentation must take into consideration the additional constraints introduced in the *descriptions* column. In this example, the description column contains information about object lengths. In other cases, object priorities, quality upper and lowerbounds, or other related constraints may be introduced in this column. □

When, the associated condition is a query, the virtual object is similar to a singleton database view. The major difference is that the visualization information is an integral part of the virtual object definition:

**Example 3.3** Table 2 shows a virtual object whose identity is decided through $parameter\_list$, which denotes a list of input parameters to the query. $ID$ and $len$ denote an object identity and object length returned by the query. Note that, in this case, the identity of the object is totally dependent on when the query is executed and what the parameters in $parameter\_list$ are. □

Next, we provide a generalization of the virtual object concept that can handle non-singleton query results; *set virtual objects* can be used to visualize queries which return a set (or bag) of results:

**Definition 3.3 (Set Virtual Object)** Let $S$ be a set of virtual objects and $q$ be a query of the form `query`$(parameter\_list)$ which returns media-objects. The pair $svo = \langle S, M, q\rangle$ is called a *set virtual object* if $M$ is a mapping from $S$ to the media-objects returned by the query $q$. The mapping $M$ is called the *virtual object assignment*. Query $q$ is called the *set virtual object descriptor* of $svo$ and it is denoted by $q_{svo}$. ◇

**Example 3.4** The following pair, $\langle S, M, q\rangle$, is a virtual object:

$q = \text{return\_travel\_images}(1998)$.

$S = \{\text{vo}_1, \text{vo}_2, \ldots, \text{vo}_n.\}$.

$M$ is a mapping from $q$ to $S$. □

Note that, in the above example, if $q$ returns $m$ images where $m < n$, then some of the virtual objects in $S$ are assigned to null objects. If on the other hand, the number of tuples ($m$) returned by $q_S$ is larger than $n$, then some of the tuples are not assigned to any virtual objects. We will later describe how to deal with both of these cases.

In the next subsection, we describe the virtual object assignment process in greater detail.

## 3.1 Virtual Object Assignment

Before describing how virtual object assignment is handled, we want to provide an overview of the terminology we use in the rest of the section.

### 3.1.1 Terminology

- $\mathcal{F} = \{f_1, \ldots, f_n\}$ denotes the set of media object formats recognized in the framework.
- $\mathcal{S} = \{s_1, \ldots, s_m\}$ denotes the set of information sources.
- $\mathcal{Q} = \{q_1, \ldots\}$ denotes the infinite set of multimedia queries that can be posed within the framework.

  The queries in $Q$ are posed in an SQL-like language extended with the following constructs. Note that the contributions in this paper do not depend on this particular language. We are introducing this language in this paper to motivate the rest of the discussions.

  - **SOURCE:** This is a construct similar to the FROM clause in standard SQL. It describes which information sources must be used in generating the answer.
  - **REPEAT:** The REPEAT clause describes how often the query must be repeated. It is useful when the underlying database is dynamically updated and when the user wants to retrieve the latest content of the database.

- $\mathcal{P} = \{p_1, \ldots\ p_j\}$ denotes a set of pre-prepared presentation specifications. For a given presentation specification $p_i$, $virtual(p_i)$ denotes the set $\{vo_1,\ \ldots,\ vo_{num(p_i)}\}$ of virtual objects in $p_i$. Note that $num(p_i)$ denotes the number of virtual objects in $p_i$.

**Example 3.5** The following is an example query that can be posed in this framework:

| | |
|---|---|
| **SELECT** | $O_1, O_2$ |
| **FROM** | $O_1\ PICT$, $O_2\ VID$ |
| **SOURCE** | $PICT$ SENSOR1, $VID$ SENSOR2, $PICT$ SENSOR3 |
| **REPEAT** | 30 |
| **WHERE** | $O_1$.target = earth and |
| | $O_2$.target = earth |

In this example, the query uses three sensors (SENSOR1, SENSOR2, and SENSOR3) to generate pictures and videos of the *earth* in 30 second intervals. The pictures of the *earth* are supplied by sensors SENSOR1 and SENSOR3, whereas the video is provided by SENSOR2. □

Let $q$ be a query generated using this syntax, let $t$ be a row in the result, and let $o$ be an object in one of the columns of $t$. Then

- $src(t)$ denotes the set of sources involved in the generation of tuple $t$,
- $time(t)$ denotes the time at which the tuple is generated,
- $row(t)$ denotes the row number of the tuple $t$ in the result table returned by source $src(t)$ at time $time(t)$.
- $src(o)$ denotes the source that provided $o$, and
- $time(o)$ denotes the time at which the object $o$ is generated $(time(o) = time(t))$.
- $tuple(o)$ denotes the tuple in which $o$ is returned.
- $pos(o)$ denotes the position of $o$ in the tuple in which it is returned.

In addition to these

- $format(o)$ returns a format describing the encoding mechanism as well as the display, synchronization, and manipulation properties of the object $o$, i.e. $format(o) \in \mathcal{F}$.
- $size(o)$ returns the size of $o$ in bytes, and
- $length(o)$ returns the length of the object $o$ in seconds (if object $o$ is a text or number, which do not have a temporal dimension, then $length(o)$ returns the flag $N/A$ which denotes *not-applicable*.

Note that when a source fails to provide an object at the requested time, then it is assumed to provide a null object $o_\perp$.

### 3.1.2 Assignment Mechanism

In this subsection, we describe how virtual object assignment is performed in our framework.

Let query $q \in \mathcal{Q}$ be a query formed in the syntax described in the previous subsection and let $p \in \mathcal{P}$ be a presentation specification that we want to use for the visualization of the results of query $q$. The pair $\langle o_i, vo_j \rangle$, where $o_i$ is an object returned by $q$ and $vo_j$ is in $virtual(p)$, is called an *assignment*.

An *assignment criterion* is a boolean function which takes an assignment, and which returns *true*, if the assignment is valid, or $false$, if the assignment is invalid.

Note that given an assignment criterion, a $vo_j \in virtual(p)$, and a set of objects $O$, there can be more than one valid assignment or no valid assignments. In the case when there are multiple valid assignments, either a valid assignment is chosen randomly, or the mitosis process we will describe in Section 3.4 is applied to the virtual object. If there are no valid assignments, $vo_j$ is assigned to the null object $o_\perp$.

In the next subsection we describe the syntax of the assignment criterion.

### 3.1.3 Assignment Criterion Syntax

As described earlier, an assignment criterion is a boolean function which takes an assignment as input and returns *true* or *false* depending on whether the criterion holds. We use the expression *assignment criterion* to denote any disjunctive normal form formula, i.e. any statement of the form $\Theta_1 \vee .. \vee \Theta_c$ where each $\Theta_i$, $1 \leq i \leq c$, is a conjunction of literals. In this subsection we will describe the literals accepted in our framework.

Let $\mathcal{T}vars = \{T_1, T_2, \ldots\}$ be an infinite set of tuple variables, $\mathcal{O}vars = \{O_1, O_2, \ldots\}$ be an infinite set of object variables, $\mathcal{F}vars = \{F_1, F_2, \ldots\}$ be an infinite set of format variables, $\mathcal{N}vars = \{N_1, N_2, \ldots\}$ be an infinite set of numeric variables, and let $\mathcal{X}vars = \{X_1, X_2, \ldots\}$ be an infinite set of alpha-numeric variables.

Let $Numerics$ denote the infinite set of all numeric constants, $Alpha\text{-}Numerics$ denote the infinite set of all alpha-numeric constants, and $Attributes$ denote the infinite set of attribute names. Let also $Obj$ be a special variable instantiated to the object in the assignment pair, and $VObj$ be a special variable instantiated to the virtual object in the assignment pair.

Then, the following is a list of literals accepted in the framework:

1. $(src(Obj) \in s)$, where $s \subseteq \mathcal{S}$ and $s \neq \emptyset$,
2. $(time(Obj) = t)$, $(time(Obj) < t)$, $(time(Obj) > t)$, where $t \in Numerics \bigcup CURTIME$, where $CURTIME$ denotes the current time.
3. $(format(Obj) \in f)$, where $f \subseteq \mathcal{F} \bigcup VObj.preferedformats$ and $f \neq \emptyset$,
4. $(size(Obj) = n)$, $(size(Obj) < n)$, $(size(Obj) > n)$, where $n \in Numerics$,
5. $(length(Obj) = n)$, $(length(Obj) < n)$, $(length(Obj) > n)$, where $n \in Numerics \bigcup$ $\{VObj.maxlength, VObj.maxlength\}$,
6. $(pos(Obj) = n)$, where $n \in Numerics$,
7. $(src(tuple(Obj)) = s)$, $(src(tuple(Obj)) \subset s)$, where $s \subseteq \mathcal{S}$ and $s \neq \emptyset$,
8. $(row(tuple(Obj)) = n)$, where $n \in Numerics$,
9. $(Obj.attr = n)$, $(Obj.attr < n)$, $(Obj.attr < n)$, where $attr \in Attributes$ and $n \in Numerics$,
10. $(Obj.attr = x)$, $(Obj.attr = xs)$, $(Obj.attr \subset xs)$, where $x \in AlphaNumerics$ and $xs \subseteq AlphaNumerics)$ and $xs \neq \emptyset$

In the above definitions, $VObj.preferedformats \subseteq \mathcal{F}$ denotes the set of formats $VObj$ is limited to, $VObj.maxlength$ denotes the maximum allowed length of the virtual object $VObj$, and $VObj.minlength$ denotes the maximum allowed length of the virtual object.

Note that if $Obj$ is a null object, then items 4, 5, and 6 above return true independently of what $f$ and $n$ are. Also, if $length(Obj)$ is $N/A$, then item 6, holds whatever $n$ is. The $CURTIME$ dynamic constant is used when the query is being repeated periodically and when the user wants to use the most recent object each time. In 9 and 10, if $attr$ is not an attribute of the object $Obj$ holds for, the literal automatically returns $false$.

#### 3.1.4 Virtual Object Assignment Defined

Finally, a virtual object assignment, $M$, for a set virtual object $SVO = \langle S, M, q \rangle$ is a set of pairs, $p_i = \langle vo_i, a_i \rangle$, where $vo_i \in S$ and $a_i$ is an assignment criterion as described earlier, such that

- $p_i = \langle vo_i, a_i \rangle$ and $p_j = \langle vo_j, a_j \rangle$ are two pairs in $VOA$, then if $vo_i = vo_j$, then $a_i = a_j$, and hence $p_i = p_j$; and
- for each object $vo_i \in S$, there exists one pair $p_j = \langle vo_j, a_j \rangle \in VOA$, such that $vo_j = vo_i$.

In the next section, we describe how a multimedia document containing virtual objects are presented to users.

### 3.2 Visualization of a the Results of a Multimedia Query using Virtual Objects

A visualization specification for a multimedia query is a set $(VO)$ of virtual objects and a set $(SP)$ of temporal and spatial specifications describing how the virtual objects must be shown to the users.

The personalized presentation seen by the user $u_i$ is called $O_i$. $O_i$ is described by a set of multimedia objects and their associated visualization constraints $(con(O_i))$. The set of constraints that a multimedia presentation system must solve before generating a presentation is $SP \cup con(O_i)$. Each user $(u_i)$ may have a different view of the document depending on his or her preferences.

The set of temporal constraints in the virtual object definition of an object $o_j$ depends only on two variables: the start time of $o_j$ $(st(o_j))$ and the end time of $o_j$ $(et(o_j))$. As a result, while the set of constraints that are in $SP$ defines the relative positioning of the objects in $VO$, the set of constraints $con(O_i)$ describes lengths of the objects in $O_i$ (the set of specifications that can be generated using $st(o_j)$ and $et(o_j)$ using difference constraints is limited to the specifications about the length of $o_j$).

In the case when virtual objects are static, i.e. their identities do not change during the lifetime of the query result visualization, we first identify the objects that need to be presented to the user using the techniques described in the previous section, and then we present them using the techniques we describe in [16, 18]. However, as described in the previous section, in some cases, more than one media-object, and in some other cases, no media object may be assigned to a given virtual object.

When, virtual objects are dynamic, i.e. their identities can change during the presentation lifetime, however, there are some complications in the presentation scheduling.

### 3.3 Dynamic Virtual Objects

A virtual object is said to be dynamic if the corresponding assignment criterion is time dependent.

**Example 3.6** The following is an example of a time dependent assignment criterion:

$((pos(Obj) = 3) \wedge (time(Obj) < 100) \vee$
$(pos(Obj) = 5) \wedge (time(Obj) = 100)) \vee$
$(Obj.creator = \text{``}JohnDoe\text{''}) \wedge (time(Obj) > 100))$

This criterion says that,

- during the first $<100$ units of the presentation, we use the third column of the tuple,
- during the $100^{th}$ unit of the presentation, we use the fifth column (attribute) of the tuple, and
- after the $100^{th}$ unit of the presentation, we use an object whose creator is `John Doe`, independent of on which column (attribute) such an object appears in the result. □

**Example 3.7** The following is an other example of a time dependent assignment criterion:

$((pos(Obj) = 3) \wedge (time(Obj) = CURTIME) \wedge (time(Obj) > 100))$

This criterion says that after the $100^{th}$ unit of the presentation, we need to execute the query at every quanta of time, and we should assign the object returned in the third column (attribute) of the tuple in the presentation. □

The absolute time values above are, in fact, relative to the start of the presentation, and $CURTIME$ denotes the current presentation clock, again assuming that when the presentation starts $CURTIME$ is equal to 0.

When scheduling a presentation, we convert the assignment criteria of time dependent objects, along with their corresponding *length* properties, into constraints and we add them into $SP$ to find final schedules. Note that since we may not be able to guess the *length* properties of all identities of the dynamic virtual objects at time 0, we may need to reschedule presentations and resolve conflicts as presentations progress.

If a time dependent assignment criterion depends on $CURTIME$, then the reassignment check of the virtual object must be done at each time clock (or periodically at each predefined time quantum), and if the virtual object indeed get reassigned to a new identity, then presentation must be rescheduled.

For other time dependent assignments, the reassignment check must be done only at the time instants declared in the assignment criteria, and again, if objects get reassigned then presentation must be rescheduled.

### 3.4 Mitosis of Virtual Objects

Note that the language described so far does not provide a means to handle multiple media objects satisfying a given selection criterion. For instance, if the author does not explicitly specify, using a selection criterion of the form $(row(Obj) = rownum)$, the row to be used, then multiple rows in the table may satisfy the rest of the criterion. However, since the number of rows that will satisfy the criterion can not be known beforehand, this causes complexities in the object assignment.

We solve this problem by letting virtual objects multiply through a process called *mitosis of virtual objects*. In this process, a virtual object give rise to multiple virtual objects each differing in their *identities*.

Since the virtual objects created by the mitosis are the children of the initial virtual object, they have to share the temporal and spatial specifications of the initial object. However, since all of the child objects can not appear at the same place at the same time, there must be a way to divide the temporal and spatial space owned by the initial virtual object among the children.

For this purpose, we define three mitosis types:

- *Space filling:* In space filling mitosis, the display of all children start at the same time. The space of the initial virtual object is divided among the children, so that they can be placed adjacently.
- *Time filling:* The temporal length of the initial object is divided among children, each children is shown one after the other at the same spatial location.
- *Space and time filling:* Both space and time dimensions are divided so that both spatial and temporal dimensions of the initial virtual object is used by the children.

Figure 4 shows different types of mitosis.

Let $O = \{o_1, o_2, \ldots, o_m\}$ be a set of objects that map to a virtual object $vo$, and let the index of the objects denote an arbitrary order. Then, depending on the mitosis strategy assigned to the virtual object $vo$, the object in $O$ are presented to the user as follows:

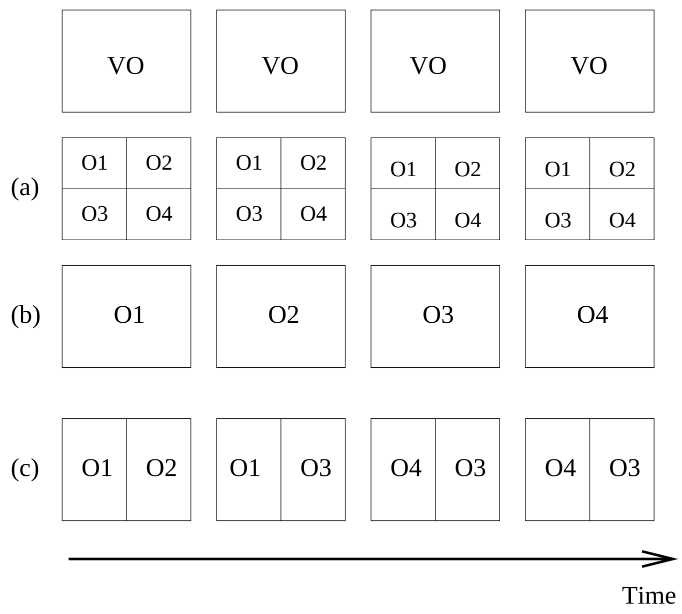



Figure 4: Different types of mitosis: (a) space filling, (b) time filling, and (c) space and time filling

#### 3.4.1 Space Filling Mitosis

As we described earlier, there exist two temporal variables, $st(vo)$ and $et(vo)$, corresponding to the virtual object $vo$, the first denoting the start time of the presentation of $vo$ and the second denoting its end time. Also, there exist four spatial variables, $x_r(vo)$, $x_l(vo)$, $y_t(vo)$, and $y_b(vo)$ corresponding $vo$. Let $C_T(vo)$ denote the set of constraints describing the temporal position and length of $vo$, and let $C_S(vo)$ denote the set of constraints describing the spatial layout of $vo$. Also, let $C_T(D)$ be the set of constraints describing the temporal properties of the multimedia document and let $C_S(D)$ be the set of constraints describing the spatial properties of the multimedia document. Let $k$ be equal to $\sqrt{(m)}$, where $m$ is the number of objects in $O$. Then, we add the following constraints to $C_T(D)$ and $C_S(D)$ to perform space filling mitosis:

- For every object $o_i$ in $O$ add the following constraints in $C_T(D)$:
  - $et(o_i) = et(vo)$ and
  - $st(vo) = st(o_i)$.
- For every object $o_i$ in $O$ such that $i \, mod \, k = 0$ add the following constraint to $C_S(D)$:
  - $x_l(vo) = x_l(o_i)$ and
  - $x_r(o_i) = x_l(o_{i+1})$ (if $i \, mod \, k \neq k-1$).
- For every object $o_i$ in $O$ such that $i \, mod \, k = k-1$ or $i = m$ add the following constraint to $C_S(D)$:
  - $x_r(o_i) = x_r(vo)$.
- For every other object $o_i$, we add the following constraint to $C_S(D)$:
  - $x_r(o_i) = x_l(o_{i+1})$.

- For every object $o_i$ in $O$ such that $i\, div\, k = 0$ we add two constraints of the following form to $C_S(D)$:
  - $y_t(o_i) = y_t(vo)$ and
  - $y_t(o_{i+k}) = y_b(o_i)$ (if $i\, div\, k \neq k-1$).
- For every object $o_i$ in $O$ such that $i\, div\, k = k-1$ we add the following constraint to $C_S(D)$:
  - $y_b(vo) = y_b(o_i)$.
- For every other object $o_i$, we add the following constraint to $C_S(D)$:
  - $y_t(o_{i+k}) = y_b(o_i)$.
- For every object $o_i$ in $O$, we add the sets $C_T(o_i)$ and $C_S(o_i)$, describing the temporal and spatial sizes of $o_i$, into $C_T(D)$ and $C_S(D)$, respectively. Note that these constraints use only the temporal and spatial variables corresponding to $o_i$.

Note that every equality in the above list is in fact represented by two difference constraints with different priorities. For every equality of the form $a = b$ above, $a - b \leq 0$ is inserted with priority $p_1$, and $b - a \leq 0$ is inserted with priority $p_2$, where $p_2 < p_1$. This way, although the mitosis structure is maintained (no overlapping objects within mitosis), it is possible to have gaps if necessary (in Section 3.5, we describe how to balance the sizes of the gaps)

### 3.4.2 Time Filling Mitosis

We add the following constraints to $C_T(D)$ and $C_S(D)$ to perform space filling mitosis:

- For every object $o_i$ in $O$, add the following constraints in $C_T(D)$:
  - $st(vo) = st(o_i)$ (if $i = 0$),
  - $et(o_i) = st(o_{i+1})$ (if $i \neq m$), and
  - $et(o_i) = et(vo)$ (if $i = m$).
- For every object $o_i$ in $O$, such that $i\, mod\, k = 0$ add the following constraint to $C_S(D)$:
  - $x_l(vo) = x_l(o_i)$,
  - $x_r(o_i) = x_r(vo)$,
  - $y_t(o_i) = y_t(vo)$, and
  - $y_b(vo) = y_b(o_i)$.

### 3.4.3 Space and Time Filling Mitosis

Space and time filling mitosis is a hybrid of the space filling and time filling mitosis methods. The user (or the system) creates a set of temporary virtual objects and places the temporary virtual objects in a space filling fashion within the original virtual object. Then, s/he (or it) places the media objects in a temporal-mitosis fashion in the corresponding temporary virtual object. It is also possible use hierarchies of temporary virtual objects to create hierarchies of space and time filling mitoses.

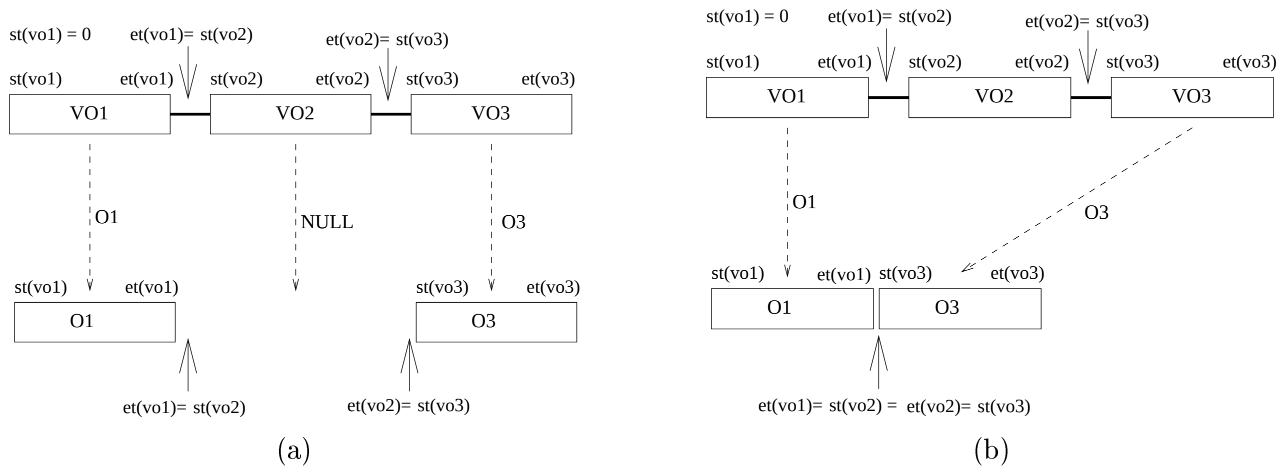



Figure 5: (a) Null object mapping and (b) null object mapping with 0 object length

### 3.5 Presentation Gaps

When an object in $VO$, say $vo_k$ is mapped to a null object, it can not be shown to the user. However, since the temporal specifications of objects are given relative to one another, when one or more of the objects in the presentation are missing, the relative ordering may lose its intended structure.

Figure 5(a) shows this problem. In this figure, $SP$ consists of the following constraints: (1) $et(vo_1) = st(vo_2)$ and (2) $et(vo_2) = st(vo_3)$; hence, $SP$ guarantees a presentation sequence of the form $vo_1$, $vo_2$, and $vo_3$. Note that if object $vo_2$ maps to a null object, then the presentation sequence must be $vo_1$ followed by $vo_3$. However, if we simply remove the constraints associated with $vo_2$ (constraints 1 and 2), then there will not remain any specification which links $et(vo_1)$ to $st(vo_3)$; hence, it is not possible to expect that in the resulting presentation $vo_3$ will start immediately after $vo_1$.

A solution to this problem is, instead of removing null-mapped object and related specifications, to set its length to zero. Hence, if $vo_k$ is mapped to a null object, then $con(O_i)$ will contain a constraint of the form $st(vo_k) = et(vo_k)$. Figure 5(b) shows this case. Note, however, that this solution may, in general, be too restrictive. For instance, there may be cases in which, the zero length constraint may be in conflict with some other constraints in $SP \cup con(O_i)$. For instance, in Figure 5(b), if $SP$ contains a constraint of the form $et(vo_3) - st(vo_3) \geq 100$ and if the sum of the lengths of objects $o_1$ and $o_3$ is less than 100, then the given set of constraints is inconsistent when the length of the object $o_2$ is set to 0. Therefore, it is not always possible to set the length of a null-mapped object to 0.

**Definition 3.4 (Presentation Gap)** The difference $et(vo_i) - st(vo_i)$ for a null-mapped virtual object $vo_i$ is the presentation gap for the virtual object $vo_i$. ◇

Presentation gaps are not desirable. Hence, it is necessary to minimize the number and the lengths of the gaps in a multimedia presentation. Furthermore, it is desirable to do this minimization in a balanced fashion. Since the shortest path based framework we use minimizes the length of the presentation, unnecessarily long gaps will automatically be removed or shortened. However, the original framework will not perform this gap elimination in a balanced way. In the next subsection, how the

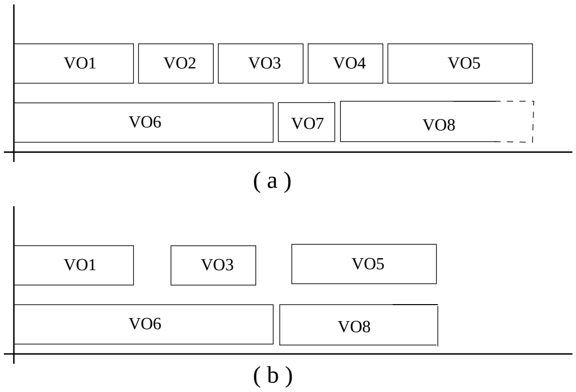



Figure 6: Balanced minimization of gaps: $vo_2$, $vo_4$, and $vo_7$ are mapped to null objects

CHIMP framework can be used to perform this balanced minimization using prioritized difference constraints.

#### 3.5.1 Handling Null-Mapped Virtual Objects

Let $VO$ be a set of virtual multimedia objects, and let $SP$ be a set of relative temporal specifications in difference constraint form. Let also $VO_\perp \subseteq VO$ be the set of virtual objects that are mapped into null objects and let $VO_\top$ be $VO - VO_\perp$. If $vo_k \in VO$, $con(vo_k)$ is the set of associated constraints from its virtual object definition (note that if $vo_k \in VO_\perp$, then $con(vo_k) = \emptyset$). Lastly, if $S$ is a set of virtual objects, then $con(S) = \bigcup_{vo_j \in S} con(vo_j)$.

Given the above definitions we calculate a schedule for the objects in $VO$ with respect to specifications in $SP$ as follows:

1. We let $SP_1 = SP$. For each virtual object $vo_k \in VO_\perp$, we add the constraint $st(vo_k) - et(vo_k) \leq 0$ into $SP'$ (In fact this is an implicit constraint that the system should maintain all the time).
2. We solve $SP_1 \cup con(VO)$ using the algorithms given in [18], i.e., we find the shortest possible distance from the end of the presentation to the start and end times of the objects.
3. We let $SP_2 = \emptyset$. For each constraint $C_i$ ($C_i \equiv X_i - Y_i \leq a_i$) in $SP_1$, we add a constraint $C_i'$ of the form $Y_i - X_i \leq -a_i$ in $SP_2$.
4. We solve $SP_2 \cup con(VO)$ using the algorithms given in [18], i.e., we find the shortest possible distance from the start of the presentation to the start and end times of the objects.

   Hence for each virtual object $vo_k$, we have the $st_{ep}(vo_k)$, $st_{lp}(vo_k)$, $et_{ep}(vo_k)$, and $et_{lp}(vo_k)$, where subscript $ep$ denotes the earliest possible scheduling time and subscript $lp$ denotes the latest possible scheduling time.

   Note that the length ($mL_k$) of the largest possible gap for null-mapped virtual object $vo_k$ is equal to $et_{lp}(vo_k) - st_{ep}(vo_k)$.
5. We let $SP_3 = SP$. We set the priority of all constraints in $SP_3$ to $pr_{max}$. For each virtual object $vo_k \in VO_\perp$, we add the following constraints into $SP_3$:

- $st(vo_k) - et(vo_k) \leq 0$ with priority $pr_{max}$,
- $et(vo_k) - st(vo_k) \leq mL_{max}/gran$ with priority 1,
- $et(vo_k) - st(vo_k) \leq 2 \times mL_{max}/gran$ with priority 2, ...,
- $et(vo_k) - st(vo_k) \leq (gran - 1) \times mL_{max}/gran$ with priority $gran - 1$, and
- $et(vo_k) - st(vo_k) \leq gran \times mL_{max}/gran$ with priority $gran$,

where $gran$ is the desired granularity and $pr_{max}$ is the maximum priority such that $pr_{max}$ is greater than $gran$, for all $vo_k \in VO_\perp$.

6. We solve $SP_3 \cup con(VO)$, where all constraints in $con(VO)$ also have $pr_{max}$ priority, using the algorithms given in [18]. The result is a presentation schedule where the gap sizes are reduced in a balanced manner when there are two or more gaps that depend on each other (Figure 6).

Next, we show how the above algorithm results in a presentation schedule where the gap sizes are reduced in a balanced manner.

**Definition 3.5 (Dependent Gaps)** Two gaps, $gap_1$ and $gap_2$, are *dependent on each other* if there exists at least a pair of inconsistent constraints, each specifying the length for one of the gaps.

Let $G$ be a set of gaps. We call $G$ a *dependent gap set* if there exists a negative cycle that includes one length constraint for each and every gap in $G$ (note that this definition does not exclude singleton dependent gap sets). $\diamond$

**Theorem 3.1** Let $VO$ be a set of virtual multimedia objects, and let $SP$ be a set of relative temporal specifications in difference constraints form. Let also $VO_\perp \subseteq VO$ be the set of virtual objects that are mapped into null objects. Let $S$ be the presentation schedule created by the above algorithm, and let $size(gap_i)$ denote the length of the gap $gap_i$. Then, the following properties hold:

1. If $G$ is a dependent gap set with at least two gaps and if $G$ is non-overlapping with any other dependent gap set in the presentation, then the above algorithm divides the lengths of the gaps in $G$ in a balanced fashion: Let $gap_1$ and $gap_2$ be two gaps in $G$. Then, if $size(gap_1) \geq size(gap_2)$, as the gap sizes and/or the granularity, $gran$, increases, $\frac{size(gap_1)}{size(gap_2)}$ gets closer to 1.

2. If $G_1$ and $G_2$ are two overlapping dependent gap sets and if $G_0 = G_1 \bigcap G_2$ is the set of gaps in the intersection of $G_1$ and $G_2$, then

   - If $gap_1$ and $gap_2$ are two gaps in $G_o$, then the first item holds also for $gap_1$ and $gap_2$.
   - If $gap_1$ and $gap_2$ are two gaps in $G_1 - G_0$, then the first item holds also for $gap_1$ and $gap_2$.
   - If $gap_1$ and $gap_2$ are two gaps in $G_2 - G_0$, then the first item holds also for $gap_1$ and $gap_2$. ♣

The proof of the theorem is presented in Appendix A.

### 3.6 Specification of Interactivity in Result Visualizations

So far, we discussed result visualization techniques for multimedia databases and we suggested that multimedia query and visualization specification languages may eventually be more tightly integrated. We already see such trends in [9, 51]. As we have seen in the previous section a multimedia query and visualization definitions consists of two parts:

- A specification stating what media objects are included in the result. For example, this specification might say that the latest weather map of Korea will be presented, together with the latest weather video, and textual data obtained from a weather server.
- A specification describing how these media objects will be presented to the user. This specification might state that the weather map is presented first, followed by the textual weather data, followed by an *option* to the user asking whether he wants to see the video (which he might decide to exclude if he does not require it).

Since both *querying* and *visualization* terms are not enough for describing the integrated act of requesting visualized content, we need a different term to describe this operation. We use the term *multimedia view specification* (or, in some sense, multimedia form specification) for this purpose.

In traditional databases, *view materialization* uses a view definition to compute the contents of the view. Informally, a multimedia view definition (MVD)[1] is a *declarative* specification of what is to be shown to the user and how.

Note that a query result in relational DBMSs is static; in contrast, a multimedia result must be *presented* to the user and, the user may want to have the opportunity to interact with the presentation, making choices on what he wants to see, thus guiding the presentation along one path or another. Consequently, the main difference between MVD and set virtual objects introduced earlier is that MVD also describes how user can *interact* with the visualized results:

**Definition 3.6 (MVD)** Let $\mathsf{MDB}$ be any multimedia database, and let $QL$ be any query language for querying this database, and let a *content framework*, $\mathsf{CF}$, be a pair, $(\mathsf{MDB}, QL)$. An MVD is a 5-tuple, $(\mathcal{V}, \mathcal{S}, \mathcal{C}, \mathcal{A}, Act)$, based on a content framework $\mathsf{CF}$, where

- $\mathcal{V} = \{vo_1, vo_2, \ldots\}$ is a set of *virtual objects*,
- $\mathcal{S} = \{\langle S_1, M_1, q_1\rangle, \langle S_2, M_2, q_3\rangle, \ldots\}$ is a set of *set virtual objects* such that
  - $\forall i, S_i \subseteq V$ and
  - $\forall i \neq j, S_i \cap S_j = \emptyset$,
- $\mathcal{C}$ is a set of presentation constraints specified on virtual objects in $V$,
- $\mathcal{A} \subseteq \mathcal{V}$ is the set of action objects, i.e., those objects that user can interact with.
- $Act = (ObjId, InObjects, OutObjects, InConstraints, OutConstraints)$ is a relation specifying, for each action object, $ObjId \in \mathcal{A}$, the objects and presentation specifications to be inserted or deleted from the presentation as a result of its invocation. $\diamond$

[1]Note that this could also be read as *multimedia visualization definition*; however, since query specifications are an integral part of MVD, we prefer the other reading

**Example 3.8** The following is an example `action` relation:

| ObjId | InObjects | OutObjects | InConstraints | OutConstraints |
|---|---|---|---|---|
| $o_3$ | $\{o_5\}$ | $\{o_{20}, o_{21}\}$ | $C_{in}$ | $C_{out}$ |

Here, the sets $C_{in}$ and $C_{out}$ describe the sets of presentation constraints that are inserted to and removed from the original presentation constraints. □

Presenting multimedia results is not trivial, and needs to take into account, the locations of the objects being presented as a result, the bandwidth offered by the network, the load on different servers, the buffer space offered, etc. In the next section, we explain how we are addressing the challenges associated with the *interactive* and *resource-hungry* nature of multimedia query results.

# 4 Coping with Interactivity and Resource Requirements

In this section, we discuss the issues related with interactive result visualization and view rendering. A main challenge associated with interactive visualization is that the system should benefit from the user profiles [31, 68] and the database feedback [63, 42] in deciding which objects are the most relevant for the user and these objects should be fetched from the corresponding sources without causing an interruption in the presentation of the results [2]

Suppose $\mathcal{M}$ is an MVD, $\mathcal{O}_a$ is the set of all action objects, and $\mathcal{O}_p^\star = \{o_1, \ldots, o_n\}$ is the set of all objects that are not action objects. We define $\mathcal{O}_p = \mathcal{O}_p^\star \cup \{start, end\}$ where `start` and `end` are special objects denoting the start and end of the presentation. $\mathcal{O}_p$ is the set of all objects to be shown to the user, based on $\mathcal{M}$. Let also $\mathcal{Q}$ be the power set of $\mathcal{O}_p$. We can define the space of all presentations described by $\mathcal{M}$ as follows:

**Definition 4.1 (Interactive Multimedia Presentation (IMP) Space)** The *Interactive Multimedia Presentation (IMP) Space* w.r.t. a given MVD, $\mathcal{M}$, consists of a labeled tree $\Upsilon$ where

- Each node $N$ is labeled by a triple $(\mathcal{QO}_N, \mathcal{CONS}_N, \mathcal{ACT}_N)$, where $\mathcal{QO}_N \in \mathcal{Q}$, $\mathcal{CONS}_N$ is a set of *presentation* constraints (to be described below) on the objects in $\mathcal{QO}_N$, and $\mathcal{ACT}_N \subseteq \mathcal{O}_a$ is a set of interaction objects.
- The root of $\Upsilon$ is labeled by $(\{\mathtt{start}\}, \emptyset, \emptyset)$ (and no other node in $\Upsilon$ is labeled by $(\{\mathtt{start}\}, \emptyset, \emptyset)$;
- Each leaf in $\Upsilon$ is labeled by the triple $(\{\mathtt{end}\}, \emptyset, \emptyset)$;
- Each edge (from node $N$ to node $N'$) in $\Upsilon$ is labeled by a 7-tuple ($a$, $\mathcal{QO}^+$, $\mathcal{QO}^-$, $\mathcal{CONS}^+$, $\mathcal{CONS}^-$, $\mathcal{ACT}^+$, $\mathcal{ACT}^-$)[3], where $a \in \mathcal{ACT}_N \cup \{\perp\}$, $\mathcal{QO}^+$ and $\mathcal{QO}^-$ are sets of objects, $\mathcal{CONS}^+$ and $\mathcal{CONS}^-$ are sets of constraints (to be described below) on the objects in $\mathcal{QO}_N \bigcup \mathcal{QO}^+$, $\mathcal{ACT}^+$ and $\mathcal{ACT}^-$ are sets of action objects, and $a$ is an interaction object or $\perp$. If $a$ is an action object, then

[2]Since, multimedia query processing is fuzzy by nature, if the system makes an error and retrieves a less relevant object, $A$, instead of a more relevant object, $B$, $A$ can still be presented to the user instead of $B$. Of course, in general such replacements are not desirable and should not be performed unless it is non-avoidable. Consequently, the system must minimize the likelihood of these replacements.

[3]This 7-tuple corresponds to the *Act* relation in $\mathcal{M}$.

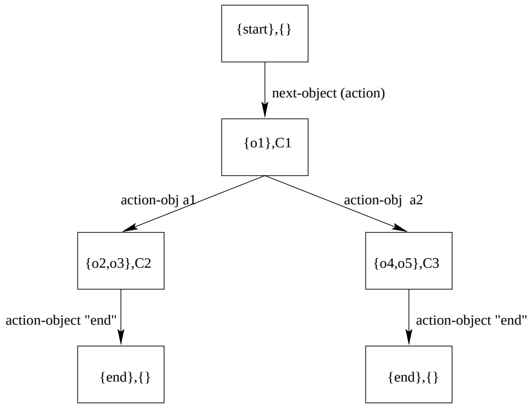



Figure 7: Example multimedia presentation

- $\mathcal{QO}_{N'} = (\mathcal{QO}_N \bigcup \mathcal{QO}^+) - \mathcal{QO}^-$,
- $\mathcal{CONS}_{N'} = (\mathcal{CONS}_N \bigcup \mathcal{CONS}^+) - \mathcal{CONS}^-$, and
- $\mathcal{ACT}_{N'} = (\mathcal{ACT}_N \bigcup \mathcal{ACT}^+) - \mathcal{ACT}^-$.

If, on the other hand, $a$ is $\perp$, then

- $\mathcal{QO}_{N'} = \mathcal{QO}_N$,
- $\mathcal{CONS}_{N'} = \mathcal{CONS}_N$, and
- $\mathcal{ACT}_{N'} = (\mathcal{ACT}_N \bigcup \mathcal{ACT}^+) - \mathcal{ACT}^-$. $\diamond$

Intuitively, each node in the IMP-space specifies a *non-interactive* presentation that is delivered to the user. The transitions between nodes occur if there is a user interaction or if the action objects on the screen are modified without the user action.

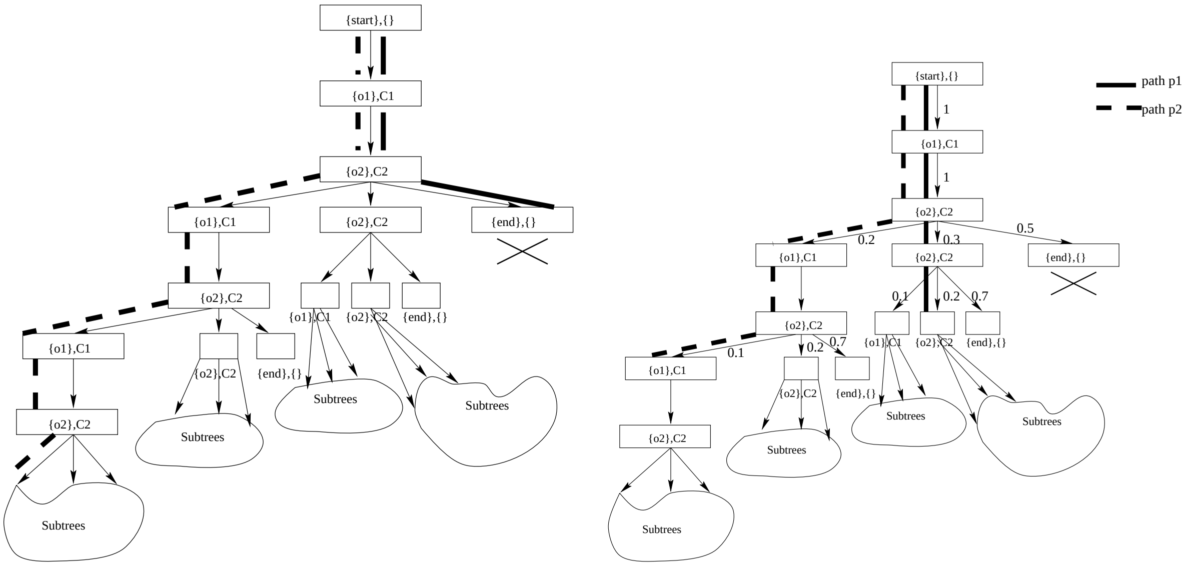
Given any IMP-Space, when the presentation is viewed by a specific user, the user follows one path in the IMP-Space from the root to a leaf. This is termed an interactive multimedia presentation (IMP) path and is described below.

**Definition 4.2 (Interactive Multimedia Presentation (IMP) Path)** An *Interactive Multimedia Presentation (IMP) Path* w.r.t. an MVD, $\mathcal{M}$, is a finite path from the root of the Interactive Multimedia Presentation Space (i.e., $(\{start\}, \emptyset, \emptyset)$) to a leaf node (i.e., $(\{end\}, \emptyset, \emptyset)$). $\diamond$

Figure 7, shows an IMP-Space with two possible IMP-paths. Each of these paths shows a possible playback of the presentation. Note that IMP-spaces of most interactive visualizations are liable to be infinite, as they will (in all likelihood) have loops in them. For example, Figure 8(a) shows an infinite IMP-space, where the user has the option of either "ending" the presentation, or viewing it again. The definition of IMP-path does not include infinite paths – thus not all paths in the IMP-space are IMP-paths.

## 4.1 Retrieving Query Results for Interactive Result Visualization

Interactivity in result visualization can be achieved in two ways: (1) retrieve all objects of interest and create the presentation afterwards and (2) retrieve the objects as necessary for the presentation. Although, implementationwise, the first option is easier, the second option has major advantages:




(a) (b)

Figure 8: (a) Example of infinite IMP-Space; (b) example of probabilistic IMP-Space

- Retrieval of all objects before the start of the visualization will increase the response time.
- Since the results are presented in an interactive setting, a particular user may choose not view all the results. The retrieval of such objects is a waste of time, network, and buffer resources.
- The retrieved query results may need to be stored locally before their presentation. The corresponding resource requirements may be to high for the clients, especially when queries involve media objects. such as video.
- In cases when data sources are continuously updated, retrieved query results may become obsolete before their presentations.

In this section, we provide *three* approaches to retrieve query results so as to facilitate the visualization of interactive multimedia views or results to dynamic multimedia queries. Since the *retrieval* operation is performed when the system decides a particular object **may** be required for the visualization (instead of waiting until the object is required), we will use the term *prefetch* instead. Note that the characteristics of the prefetching problem in this domain is different from those of

- prefetching in software systems [15]: In multimedia result visualization, the sizes of the data objects to be prefetched are very large and different from each other. Also, unlike a missed but required code statement which can be fetched into memory in a very short amount of time, if a media object is missed during prefetching, due to network delay and server response time its retrieval may not be feasible at all. On the other hand, although a missed code statement is crucial for the execution, a missed media object may be treated by place-holder objects or gap minimization techniques introduced earlier.
- prefetching in file and database systems [4, 34, 40, 56]: Most of the current work on prefetching documents/objects from databases use the semantic, reference, or access relationships between

information units(page, document, or object) to identify clusters. Prefetching is performed using these clusters. Note that these clusters generally depend on the properties (such as the schema) of data, not on the user or visualization specifications. Although whenever the necessary statics are available these techniques can also be useful in our framework, our focus is to use the visualization properties in prefetching.

- prefetching in multimedia servers [50]: Prefetching in stream information (such as video) servers is mostly based on the assumption that the users are more likely to follow a linear presentation sequence (such as during the regular viewing of a video stream). This, of course, does not hold in highly interactive situations. Also, generally, the sizes of the prefetched data are generally constant as the amount of frames that is viewed by the client in unit amount of time is known in advance and the average size of the frames are pretty much constant over time. In our domain, however, the number of objects to be viewed and the sizes of these objects are not constant.

LRU, or any other buffer management technique, could be used to free up space for new objects in the prefetch buffers. In this paper, we do not consider how does the buffer space is managed for those objects which have been viewed or omitted from the presentation by user interactions.

#### 4.1.1 Probabilistic Spaces

All three prefetch approaches depend upon a concept called a *probabilistic IMP-space*.

**Definition 4.3 (Probabilistic IMP-Spaces)** Suppose we consider an IMP-space $\Upsilon$. A *probabilistic extension of* $\Upsilon$ is a pair $(\Upsilon, \wp)$ where $\wp$ is a mapping that takes edges in $\Upsilon$ as input and produces real numbers in the $[0,1]$ interval as output such that for each non-leaf node $N$ in $\Upsilon$,

$$\sum_{N' \text{ is a child of } N} \wp(N, N') = 1.$$

Note that these are *conditional probabilities* – given the fact that the user has reached node $N$ in the playout, it estimates the probability of a specific child of $N$ occurring as the next selection. $\diamond$

Intuitively, whenever the user is presented with an interactive option, he chooses at most one of the options. $\wp$ measures the probability of these choices or the user profile [31, 68]. As such, $\wp$ can be *accurately* estimated once an MVD has been viewed by several users (the longer it has been in use, the more accurately $\wp$ can be estimated). Fig 8(b) provides an example of probabilistic extensions of the IMP-space.

Suppose now that the playout of the interactive multimedia presentation has reached node $N$ in a probabilistic IMP-space. To determine which objects may be needed in the future the system must look ahead in the tree $\Upsilon$, to determine what the probabilities of certain paths are.

**Definition 4.4 (Path Probabilities in Probabilistic IMP-Spaces)** Suppose $(\Upsilon, \wp)$ is a probabilistic IMP-space, and $N$ is a node in $\Upsilon$. Let $\Upsilon_N^k$ denote the subtree rooted at node $N$ $k$ levels down. For any path $p$ in this subtree consisting of at most $k$ edges $(N_0, N_1), (N_1, N_2), \ldots, (N_{k-1}, N_k)$

where $N_0 = N$, the *probability of path p being taken, given that node N has been reached in the playout* is given by
$\mathbf{P}(p \mid N) = \Pi_{i=0}^{k-1} \wp(N_i, N_{i+1})$. $\diamond$

The above definition associates a probability with each path of length $k$. $k$ can be chosen any way we want, but as we will see in Section 5, choosing a large $k$ causes the probabilities of most paths to become very small, making it difficult to identify "highly" likely paths. From an intuitive point of view, this says that looking ahead too far into the future is not helpful as the probabilities of predictions being right decreases substantially. Choosing a $k$ that is too small is not helpful either – it provides few opportunities for prefetching.

Note that we can define the probability, $\mathbf{P}_{j-bounded}(o, N))$, of a given object $o$'s occurrence within the presentation subtree of depth $j$ with root $N$ as follows:

**Definition 4.5 (j-bounded object appearance probability)** Suppose $PS = (\Upsilon, \wp)$ is a probabilistic IMP-space, $N$ is a node in $\Upsilon$, $\Upsilon_N^j$ denotes the subtree rooted at node $N$, $j$ levels down and $o$ is an object in the presentation. Then, the *j-bounded appearance probability of o given Hist(t)* is

$$\mathbf{P}_{j-bounded}(o, N) = \sum_{0 \leq i \leq j} \sum_{\forall path_m \in \Upsilon_N^i \text{ with } o \text{ in it}} \mathbf{P}(path_m).$$
$\diamond$

What this definition says is that if we "look ahead" $j$ levels in the tree, then the probability that object $o$ is needed in those $j$ levels is the sum of the probabilities of the paths containing $j$. Given this information we can do three kinds of result prefetching:

- Path Prefetching: This fetches as many objects along the most probable paths in $\Upsilon_N^k$ as will fit into the prefetch buffer.
- Object Prefetching: This fetches the objects that will most probably be required (within the limitations imposed by the size of the prefetch buffer).
- Hybrid Prefetching: This reserves part of the prefetch buffer for path prefetching and the rest is devoted to prefetching most probable objects that are not already prefetched by path-prefetching.

### 4.1.2 Path Prefetching

In this strategy, we find the most probable paths in $\Upsilon_N^k$ and consider them in descending order of probability. We first take the most probable path and follow it from node $N$, buffering all objects encountered on the way, till either the buffer is filled or till the path if exhausted. In the latter case, we pick the second most probable path and continue. Path prefetching is useful when the users preference do closely match the most likely paths as predicted by the probabilities. In other words, if our system has been running awhile (and hence the probabilities are "reliable") and if the user is a "normal" user, then path prefetching is likely to be useful. However, if the user is an "abnormal" user, then path prefetching is likely to provide poor service to him. The main challenge in path prefetching of course, is to accurately and rapidly compute most probable paths in descending order of probability. The algorithm shown in Figure 9 describes how we prefetch most probable paths.

**Algorithm 4.1 (Path prefetching algorithm)**
**Input:**

- A probabilistic IMP-space, $PS$,
- A presentation state, $N$,
- A set of objects $OBJ$,
- A $path-policy$ describing how the objects will be chosen within a given path (if the objects do not fit into the available buffer),
- A bound $j$ on the future, and
- A buffer size, $Size$.

1. Calculate the path probability of each path of length $j$ starting from $N$. Note that this process can be guided (using an A* like algorithm [49]) so that the paths are discovered in a decreasing probability order.
2. `objects_to_prefetch` $= \emptyset$
3. For each path $p_i$, in decreasing probability order, do
   (a) if the set of all objects $O$ in $p_i$ that do not exist in `objects_to_prefetch` (i.e., $O \bigcap$ `objects_to_prefetch` $= \emptyset$) fits into the buffer of size $Size$, then
      i. `objects_to_prefetch` = `objects_to_prefetch` $\bigcup O$.
      ii. $Size = Size - totalsizeof(O)$.
   (b) if the set of all objects $O$ in $p_i$ that do not exist in `objects_to_prefetch` does not fit into the buffer of size $Size$, then
      i. using the $path - policy$, choose a set, $O' \subset O$, of objects that fits into size $Size$,
      ii. `objects_to_prefetch` = `objects_to_prefetch` $\bigcup O'$.
      iii. $Size = Size - totalsizeof(O')$.

Figure 9: Path prefetching algorithm

### 4.1.3 Object Prefetching

In this strategy, we fill the prefetch buffer by first inserting the most probable object, then the second most probable object, and so on till the buffer is filled. Unlike path prefetching, object prefetching makes a difference when the user is *not* a conformist, but rather does something different from what the path probabilities predict he is likely to do – as such, object prefetching is the preferred approach when a new multimedia presentation is created, and user's behavior patterns are not well understood. This strategy requires an efficient way of identifying the most probably needed objects in decreasing order of probabilities. The following algorithm shows how object prefetching can be efficiently implemented.

**Definition 4.6 (j-bounded Object Retrieval Profit)** Suppose $PS = (\Upsilon, \wp)$ is a probabilistic IMP-space, $N$ is a node in $\Upsilon$, and $o$ is an object in the presentation. Then, the *j-bounded object retrieval profit* of $o$ is $profit_{j-bounded}(o, N) = \mathbf{P}_{j-bounded}(o, N) \times gain(o, N) - cost(o, N)$, where $gain$ is a function which evaluates how much system gains by retrieving object $o$ at time $t$ (considering only the next $j$ units of time), and $cost$ is a function which evaluates the cost of the retrieval of the object at node $N$ – and hence at the time node $N$ was reached in the playout. $\diamond$

Note that $\mathbf{P}_{j-bounded}(o,N) \times gain(o,N)$ is the expected gain of the system from the retrieval of object $o$ at the time node $N$ is reached. Hence, $profit_{j-bounded}(o,N)$ describes how desirable is the retrieval of object $o$ at node $N$ if one considers a future of $j$ time units.

**Definition 4.7 ((k,j,c)-retrieval Policy)** Suppose $PS = (\Upsilon, \wp)$ is a probabilistic IMP-space, $N$ is a node in $\Upsilon$, and $o$ is an object in the presentation. Then, a *(k,j,c)-retrieval policy* (w.r.t. $N$) is a set $O$ of objects, such that

- $\forall_{o_i \in O}(profit_{j-bounded}(o_i,N) \geq k)$.
- $\sum_{o_i \in O}(cost(o,N)) \leq c$. ◇

Intuitively, a (k,j,c)-retrieval policy dictates that the system chooses (after considering a future of $j$ time units) to retrieve a set of objects such that (1) each of these objects has a profit greater than or equal to $k$, and (2) the sum of the retrieval costs of these objects is less than or equal to $c$. The system manager can set $k, j, c$ as he feels appropriate.

**Definition 4.8 (Optimal (k,j,c)-retrieval Policy)** Suppose $PS = (\Upsilon, \wp)$ is a probabilistic IMP-space, $N$ is a node in $\Upsilon$, and $o$ is an object in the presentation. Then, an *optimal (k,j,c)-retrieval policy* (w.r.t. $N$) is a set $O$ of objects, such that

- $\forall_{o_i \in O}(profit_{j-bounded}(o_i,N) \geq k)$.
- $\sum_{o_i \in O}(cost(o,N)) \leq c$.
- $\sum_{o_i \in O}(profit_{j-bounded}(o_i,N))$ is maximum among all *(k,j,c)-retrieval policies.* ◇

An optimal (k,j,c)-retrieval policy dictates that the system chooses (after considering a future of $j$ time units) to retrieve a set of objects such that (1) each of these objects has a profit greater than or equal to $k$, (2) the sum of the retrieval costs of these objects is less than or equal to $c$, and (3) the sum of the retrieval profits of these objects is maximum among all (k,j,c)-retrievals.

Figure 10 shows the object retrieval-caching algorithm which uses the *optimal (k,j,c)-retrieval policy* to decide on which objects to cache for quick access in case of user interaction. The ORCA algorithm uses the following components:

**Incremental Computation of $\mathbf{P}_{j-bounded}(o,N)$:** $\mathbf{P}_{j-bounded}(o,N)$'s computation, in general, can be quite expensive. If the number of choices at each instant is bounded by $c$, then there are $c^j - 1$ sums in the calculation. Note however that if we know $\mathbf{P}_{j-bounded}(o,N')$ where $N'$ is the parent of $N$ (which is certainly a reasonable assumption), then we can incrementally compute $\mathbf{P}_{j-bounded}(o,N)$ in constant time.

**0−1 Knapsack Algorithm Component:** The ORCA algorithm builds on top of the well known `0-1 knapsack problem`. Once $profit_{j-bounded}(o_i,N)$ and $cost(o_i,N)$ are known for each object, by setting the knapsack size to $c$, object weights to $cost(o_i,N)$, and object values to $profit_{j-bounded}(o_i,N)$).

There exists a dynamic programming-based algorithm, for the 0-1 knapsack problem, that runs in $O(n.W)$ where $n$ is the number of objects and $W$ is the knapsack size ([27]). Although this algorithm

**Algorithm 4.2 (Object retrieval-caching algorithm ($ORCA$))**
**Input:**

- A probabilistic IMP-space, $PS$,
- A presentation state, $N$,
- A set of objects $OBJ$,
- A $(k, j, c) - retrieval$ policy,
- A bound $j$ on the future.

1. For each object $o_i \in OBJ$, calculate $gain(o_i, N)$ and $cost(o_i, N)$.
2. For each object $o_i \in OBJ$, calculate (incrementally) $\mathbf{P}_{j-bounded}(o, N)$.
3. For each object $o_i \in OBJ$, calculate $profit_{j-bounded}(o_i, N)$, eliminate all objects whose profits are smaller than $k$,
4. Solve the *0-1 knapsack algorithm* with the knapsack size $c$, object weights $cost(o_i, N)$, and object values $profit_{j-bounded}(o_i, N)$).
5. Return the set of objects returned by the *0-1 knapsack algorithm*

Figure 10: Object retrieval-caching algorithm ($ORCA$)

is not polynomial in the size of the problem, it is polynomial in the number of objects (i.e., it is pseudo-polynomial). Hence, we can use this algorithm to solve the *optimal (k,j,c)-retrieval policy* in $O(number\_of\_objects \times c + \Delta)$ (Figure 10), where $\Delta$ is time necessary to calculate $gain(o_i, t)$, $cost(o_i, t)$, $\mathbf{P}_{j-bounded}(o, Hist(t))$, and $profit_{j-bounded}$ $(o_i, Hist(t))$.

Since $gain(o_i, t)$ and $cost(o_i, t)$ are independent of other parameters, we can assume that their computation requires $O(1)$ time. Hence, once $\mathbf{P}_{j-bounded}(o, N)$ calculated, the calculation of the value of $profit_{j-bounded}(o_i, N)$ will take only $O(1)$ time.

Calculation of $\mathbf{P}_{j-bounded}(o, N)$, on the other hand, is *expensive*. It requires the creation of a *future* presentation tree of depth $j$, where each branch has an associated probability, and then it requires the calculation of the aggregate appearance probabilities of object $o_i$ on the nodes of the tree (Definition 4.5). On the other hand, once the tree is created, it is possible to incrementally update the tree at each state transition by pruning the branches that are not chosen at a given state and by adding new leaves to the chosen branch. Note that the number of nodes in the tree is $O(k^j)$, where $k$ is the maximum number of choices at every state ($k \leq (number\_of\_actionobjects + 1)$) and $j$ is the bound on the future. In Figure 11, we present a heuristic which prunes down the nodes in the future tree.

**Theorem 4.1** Let $m = number\_of\_actionobjects$ and let $s$ be an integer such that $0 < s << j$). The total number of nodes in the tree created by this heuristic is

$$\sum_{i=1}^{s}\left(\frac{s!}{(s-i)!} \times m^i + \frac{s!}{(s-i+1)!} \times m^{i-1} \times 2\right)$$

or, equivalently, it is $O(m^s)$. ♣

The proof of this theorem is presented in Appendix B.

**Algorithm 4.3 (Tree Creation Algorithm ($TCA$))**
**Input:**

- A probabilistic IMP-space, $PS$,
- A presentation node, $N$,
- A set of objects,
- A bound, $j$, on the future,
- A limit, $s$, on the time points per action object.

1. If $s = 0$ then stop
2. At node $N$ (and current time $t$) calculate the *j-bounded future* (assuming no user interaction).
3. For each action object $a_i$, where $t < et(a_i) \leq t+j$, choose $s$ time points $t_{i,b}$, where $1 \leq b \leq s$, $t < t_{i,b} < t_{i+1,b} < t+j+1)$, and $t_{a,b} \neq t_{c,d}$ if $a \neq c$ or $b \neq d$. Here, $et(a_i)$ denotes the end time of action object $a_i$.
4. Let $t_{\perp,\perp}$ correspond to time $t$ and let $t_{\top,\top}$ correspond to time $t+j+1$.
5. Let the probability that object $a_i$ is chosen at time $t_{i,b}$ be equal to

$$p(t_{i,b}) = \sum_{t'=t_{i,b}}^{t_{i+1,b}} (\mathbf{P}(a_i, t' | Hist(t_{i,b})))$$

   Let also $p(t_{\perp,\perp})$ be 1.
6. For each pair of time points ($t_{a,b} < t_{c,d}$) such that there exists no other time point $t_{e,f}$ satisfying $t_{a,b} < t_{e,f} < t_{c,d}$, create an edge with probability $1 - p(t_{a,b})$.
7. For each time point $t_{a,b}$ (not $\in \{t, t+j\}$) execute this algorithm recursively (with initial time set to $t_{a,b}$, the bound, $j$, set to $t+j-t_{a,b}$, and the limit set to $s-1$) to get a tree $T_{a,b}$. Connect $t_{a,b}$ with the root of $T_{a,b}$ with an edge of probability $p(t_{a,b})$.

Figure 11: Tree creation algorithm ($TCA$)

### 4.1.4 Hybrid Prefetching

Hybrid prefetching attempts to take advantage of the good features of both path and object prefetching. In hybrid prefetching, we first fix a *page-prefetch percentage*, denoted pp. This means that pp% of the prefetch buffer space is devoted to page prefetching. The rest is devoted to prefetching the most probable objects that have not already been prefetched by page-prefetching. For instance, suppose path prefetching causes objects $o_1, o_2, o_3$ to be prefetched. Suppose the most probable objects (in decreasing order of probabilities) are $o_3, o_5, o_1, o_2, o_4$, and we have space for only one object to be prefetched for object prefetching. Then $o_5$ is prefetched (as $o_3$ has already been prefetched by path prefetching).

Hybrid prefetching attempts to obtain an advantage over the other prefetching strategies by getting the best of both worlds – it attempts to serve "typical" clients by using path prefetching and "atypical" clients by object prefetching. The algorithm depicted in Figure 12 describes how the set of objects to be retrieved by hybrid prefetching is computed.

**Algorithm 4.4 (Hybrid Prefetching Algorithm ($HPA$))**
**Input:**

- A probabilistic IMP-space, $PS$, a presentation state, $\mathsf{ps}_t$, and the corresponding history, $Hist(t)$,
- A set of objects, $OBJ$,
- A bound, $j$, on the future,
- A $(k, j, c) - objectretrieval$ policy,
- A path-prefetching policy, $path\_policy$,
- A page-prefetch percentage, $\mathsf{pp}$,
- An integer, $Size$, denoting the size of the local prefetch buffers.

1. Let $Size\_path$ be $\mathsf{pp} \times Size$ and let $Size\_obj$ be $Size - Size\_path$.
2. Apply the path prefetching algorithm with parameters, $path\_policy$, $OBJ$, and $Size\_path$ to find a set a set of objects $OBJ_{path} \subseteq OBJ$,
3. Fetch the objects in $OBJ_{path}$.
4. Let $OBJ'$ be $OBJ - OBJ_{path}$.
5. Apply the object prefetching algorithm, $ORCA$ with parameters $(k, j, c) - objectretrieval$ policy, $OBJ'$, and $Size\_obj$ to find to find a set a set of objects $OBJ_{obj} \subseteq OBJ'$
6. Fetch the objects in $OBJ_{obj}$.

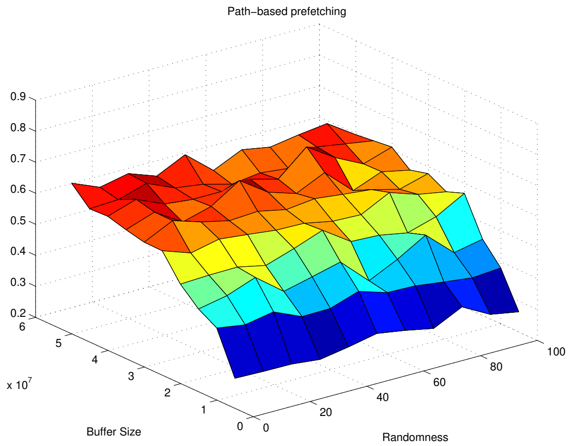
Figure 12: Hybrid prefetching algorithm ($HPA$)

# 5 Implementation and Experiments

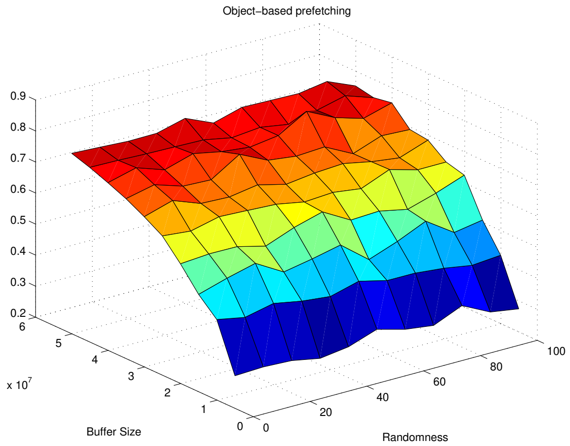
We implemented *MediaView*, an interactive visualization and multimedia view management system, based on the algorithms described in this paper. In this section we report on the results of the experiments we performed to evaluate the effectiveness of these algorithms.

We implemented the three object prefetching algorithms described in this paper (path-based, object-based, and hybrid) using C on Unix/Solaris platform. We have also implemented a multimedia view-definition/case generator to create the view definitions on which these object prefetching algorithms will be used. The view-definition/case generator uses 2000 objects of varying size (approximately 50% of the objects average 500,000 bytes each, 30% of the objects average 5 MBs each, and the rest average 12 MBs each) to create a document, where there are at most 50 objects visible at a time and there are at most 10 choices that a user can have at a time. The document/case generator also generates user profiles to attach probabilities to each choice. Note that these assumptions are quite reasonable – users typically are unlikely to want to simultaneously view more than 50 large media objects at a time, and usually, menus present only a few interaction options.

In addition to the view-definition/case generator, we developed a viewer/case generator which uses document information and user profiles to generate a walk-through (i.e., presentation realization) of the document. The walk-through uses a parameter called *randomness* to decide how much it follows the profiles. If this parameter is equal to 0.0, then the viewer is assumed to follow the user profiles,




(a) Path-based Prefetching




(b) Object-based Prefetching

Figure 13: Comparison of the *hit-ratio*s of pure path-based prefetching and pure object-based prefetching algorithms

whereas, if the randomness value is equal to 1.0, then the user behaves completely randomly.

We, then, used these case generators to run experiments which measure different properties of the prefetching algorithms. In these experiments, we varied the local buffer size between 5MBs and 50MBs, the randomness between 0.0 and 1.0, and the hybrid-ratio (0.0 - pure object-based, 1.0 - pure path-based) between 0.0 and 1.0. In each execution, we looked 5 steps ahead for prefetching. We ran each experiment 80 times to guarantee statistical significance.

**Effectiveness of Three Prefetch Strategies (Hit-Ratios):** Figure 13 shows the results of the experiments we ran to evaluate the effectiveness of pure object-based and pure path-based prefetching algorithms. These graphs shows how the hit ratio (i.e., the percentage of the required objects that were fetched by the prefetch algorithms) changes when the buffer size and the *randomness* value changes. Note that at higher randomness, the hit-ratio does not always increase monotonously with the increasing buffer size. An increase in the buffer size may cause an object with a higher resource requirement and higher access probability replace multiple smaller objects with lower access probabilities. If, the user does not behave as expected (i.e., higher randomness in the experiments) this may lead into reduction in the hit-ratio. Again, both algorithms showed an increase in the hit ratio as the local buffer size is increased. The path-based algorithm reached a plateau around 60% hit ratio, whereas the object-based algorithm reached a similar plateau at 70%.

We also ran experiments to measure the hit-ratio for the hybrid prefetching algorithm for different hybrid-ratios. Figure 14 shows how the hit-ratio changes as the buffer-size and the hybrid-ratio varies. The hybrid algorithm showed a similar increase as the buffer size increased, and the plateau was around to 70% hit for hybrid-ratios closer to 0.0 and it was around 60% hit for hybrid-ratios closer to 1.0.

**Buffer Utilization Results:** Figure 15 reports on the number of objects prefetched by the hybrid algorithm and on the resulting buffer utilization. Both Figure 15(a) and Figure 15(b) have varying

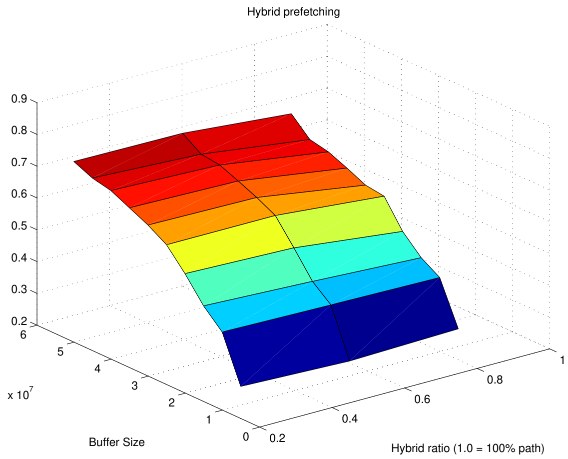
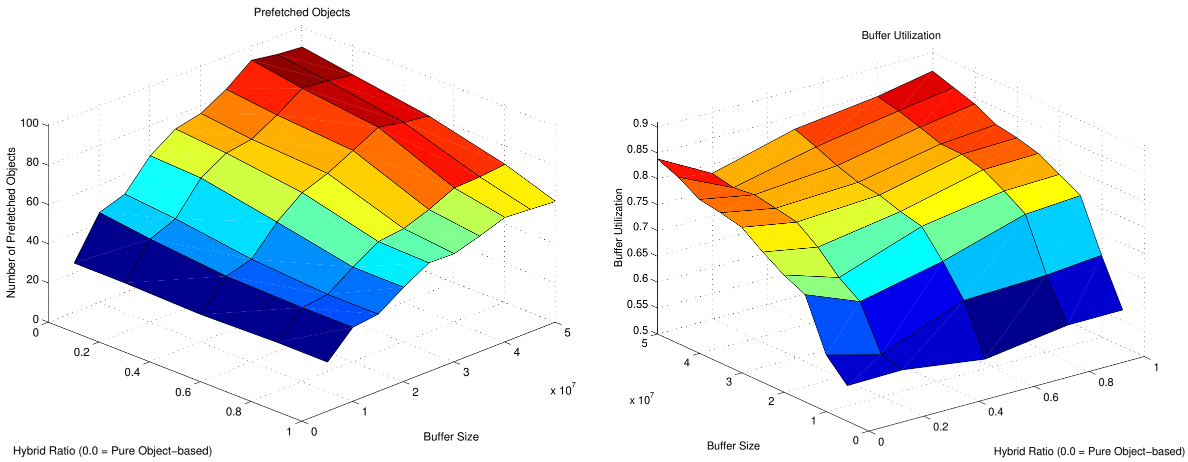



Figure 14: *Hit-ratio*s of the hybrid prefetching algorithm for different hybrid-ratios




(a) Number of objects prefetched

(b) Buffer utilization

Figure 15: Number of objects prefetched and buffer utilization

hybrid-ratio and buffer sizes. According to Figure 15(a), the number of prefetched objects increases as buffer size increases (as expected) and as the hybrid-ratio decreases. This observation, in fact, is in keeping with the fact that the hit-ratio was higher for lower hybrid-ratios. The difference between lower and higher hybrid ratios are more pronounced when the local buffers are large.

Figure 15(b), on the other hand, reports on the buffer utilization of the prefetching algorithms. According to this figure, buffer utilization increases with both buffer size and the hit ratio. The increase in the buffer utilization with the increase in the buffer size is due to the decrease in the percentage of the space where no object can fit in. The increase in the buffer utilization with the increase in the hybrid-ratio, on the other hand, suggests that the path-based prefetching algorithm works more effectively with larger buffer sizes (i.e., larger hybrid-ratio). Note that the careful examination of the graph shows that when the hybrid-ratio is exactly 0.0 (i.e., when the pure object-based algorithm is

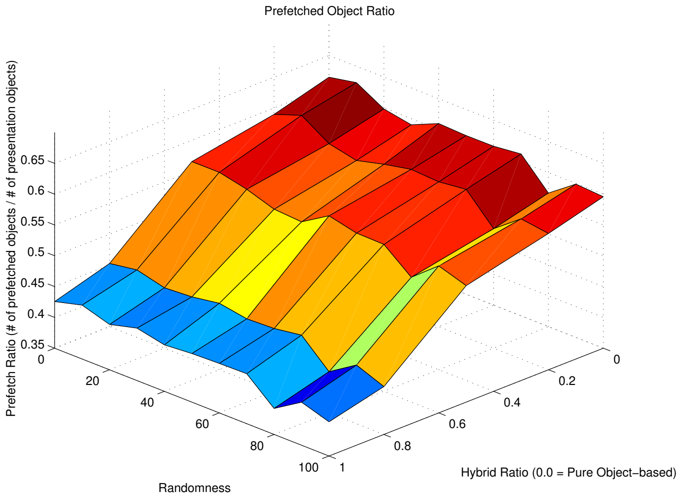



Figure 16: Prefetch ratio (i.e., number of prefetched objects/number of presentation objects)

used) the utilization is almost as high as the utilization when the hybrid-ratio is exactly 1.0 (i.e., when the pure path-based algorithm is used). This also justifies the above observation, because when pure object-based prefetching is used the whole buffer space is utilized effectively, whereas when the hybrid algorithm with hybrid-ratio is 0.2 is used 20% of the buffer is not effectively used.

**Prefetch-Ratio Results:** Figure 16 reports on the prefetch ratio, i.e., the ratio of the number of prefetched objects to the number of required objects, as the *randomness* value and the hybrid-ratio changes (note that this is different from the hit-ratio). According to this figure, the prefetch-ratio is mostly independent from the randomness value. On the other hand, the ratio increases as the hybrid ratio decreases. This observation elaborates on the earlier observation which stated that the number of prefetched objects as the hybrid ratio decreases; in fact, not only the absolute value of the number of prefetched objects increases, but the relative value of the number of prefetched objects to the number of required objects increases.

**Execution Times:** Last, we report on the execution times of the prefetch algorithms. For the configuration described above, the object-based algorithm (without the use of the TCA heuristic) takes approximately half a second to execute whereas the path-based algorithm takes on the order of 10 milliseconds. This suggests that, despite of its advantages, unless a heuristic (such as TCA) is used to reduce the number of nodes on the tree or the incremental algorithms are used, the object-based algorithm may be too costly to use for larger prefetch depths.

# 6 Conclusions

With the increased use in multimedia databases, visualization of multimedia queries is getting an important research area. The fact that multimedia queries can return multimedia objects with different characteristics makes it harder to create a visualizations for multimedia queries. The use of virtual objects along with flexible multimedia specifications provides a solution to this problem. In this paper we described a framework for multimedia database result visualization based on CHIMP [16, 17, 18].

In this paper, we have also taken the first steps towards the development of the concept of a *view* for multimedia database systems. We have stated that a multimedia view definition consists of two parts – a content specification that is specified through queries (whose answers can be dynamically computed), and a presentation component, that specifies in what form the results of materialization of the view must be presented.

In this paper, we emphasized that visualization of the results of a multimedia query may need be interactive, as users may want modify their queries dynamically while viewing query results. Multimedia views also produce interactive multimedia presentations when they are materialized.

Based on this paradigm, we have presented three result fetching techniques that allow us to prefetch objects that users viewing a presentation *might* want. As such prefetching techniques are based on probabilistic assessments (gleaned from past statistics) on what interactive choices users have made in the past, our three prefetch strategies are probabilistic. Our results show that these strategies may be efficiently implemented in large scale systems, and that on the average, they achieve about 70% correctness in their guess about what the user will do in the future. This means that 70% of the objects that a user will need in the future are in fact prefetched prior to the user's actually requesting the objects, and 30% of the objects are dynamically fetched when the user makes his selections.

Presenting the results of a multimedia query or materializing a multimedia view is not trivial, and it needs to take into account the locations of the objects being presented, the bandwidth offered by the network, the load on different servers, the buffer space offered, etc.

We have implemented *MediaView* based on the algorithms described in this paper. The experiments we ran using this system showed that the prefetching algorithms can be effectively used in improving the efficiency of rendering of multimedia views.

## References


[1] S. Abiteboul, S. Cluet, and T. Milo. Querying and updating the file. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, pages 73–84, 1993.

[2] S. Adali, K.S. Candan, S.-S. Chen, K. Erol, and V.S.Subrahmanian. *Advanced Video Information Systems*, ACM Multimedia Systems Journal, 4, pps 172–186, 1996.

[3] S. Adali, K.S.Candan, Y. Papakonstantinou and V.S.Subrahmanian. *Query Processing in Distributed Mediated Systems*, *Proc. 1996 ACM SIGMOD Conf.*, Montreal, Canada, June 1996.

[4] J.-H. Ahn and Hyoung-Joo Kim. *SEOF: An Adaptable Object Prefetch Policy for Object-Oriented Database Systems.* ICDE 1997, pp. 4-13, 1997.

[5] J.F. Allen. *Maintaining Knowledge about Temporal Intervals*, Communications of the ACM, vol. 26, no. 11, pp. 832-843, November 1983.

[6] J.F. Allen. *Towards a General Theory of Time and Action*, Artificial Intelligence, 23, pp. 123-154, 1984.

[7] Elisabeth André, Thomas Rist, and Jochen Müller. *WebPersona: A Life-Like Presentation Agent for Educational Applications on the World-Wide Web.* Proceedings of the Workshop "Intelligent Educational Systems on the World Wide Web", 8th World Conference of the AIED Society, Kobe, Japan, 18-22 August 1997.

[8] M. Arya, W. Cody, C. Faloutsos, J. Richardson and A. Toga. *Design and Implementation of QBISM, a 3D Medical Image Database System*, in: (V.S. Subrahmanian and S. Jajodia, eds.) "Multimedia Database Systems: Issues and Research Directions", Springer 1995.

[9] C. Baral, G. Gonzalez, and T.C. Som. *Design and Implementation of Display Specifications for Multimedia Answers*, ICDE'98, pp. 558-565, Orlando, Florida, 1998.

[10] P. van Beek. *Approximation Algorithms for Temporal Reasoning*, Proceedings of the 11th International Joint Conference on Artificial Intelligence. Detroit, MI, USA, Morgan Kaufmann, pp. 1291-1296, August 1989.

[11] J. Blakeley, N. Coburn, and P.-A. Larson. *Updating Derived Relations: DEtecting Irrelevant and Autonomously Computable Updates. ACM Trans. on Database Systems*, 14(3):369–400, 1989.

[12] M.C. Buchanan and P.T. Zellweger. *Automatic Temporal Layout Mechanisms* ACM Multimedia 93, pp. 341-350, 1993.

[13] M.C. Buchanan and P.T. Zellweger. *Scheduling Multimedia Documents Using Temporal Constraints*, Network and Operating System Support for Digital Audio and Video, P. Venkat Rangan (ed.), Springer-Verlag, Berlin Heidelberg New York, pp. 237-249, 1993.

[14] M.C. Buchanan and P.T. Zellweger. *Automatically Generating Consistent Schedules for Multimedia Documents*, ACM/Springer-Verlag Journal of Multimedia Systems, vol. 1, no. 2, 1993.

[15] D. Callahan, K. Kennedy, and A. Porterfield. *Software Prefetching.* ASPLOS 1991, pp. 40-52, 1991.

[16] K.S. Candan, B. Prabhakaran and V.S. Subrahmanian. *CHIMP: A Framework for Supporting Multimedia Document Authoring and Presentation*, Proc. 1996 ACM Multimedia 1996 Conference, Boston, MA, Nov. 1996.

[17] K.S. Candan, B. Prabhakaran, and V.S. Subrahmanian. *Retrieval Schedules Based on Resource Availability and Flexible Presentation Specifications,* ACM-Springer Multimedia Systems Journal, Vol. 6, No. 4, pp. 232-250, 1998.

[18] K.S. Candan, B. Prabhakaran, and V.S. Subrahmanian. *Collaborative Multimedia Documents: Authoring and Presentation,* Accepted for publication in the International Journal of Intelligent Systems on Multimedia Computing Systems.

[19] K.S. Candan. *A Framework for Distributed Multimedia Collaborations.* Ph.D. Thesis, University of Maryland at College Park, College Park, Maryland, 1997.

[20] K.S. Candan, V.S. Subrahmanian, and P.V. Rangan. *Towards a Theory of Collaborative Multimedia*, IEEE International Conference on Multimedia Computing and Systems, pp. 279-282, Hiroshima, Japan, June 96.

[21] K.S. Candan, V.S. Subrahmanian, and P.V. Rangan. *Collaborative Multimedia Systems: Synthesis of Media Objects.* IEEE Trans. on Knowledge and Data Engineering, Vol. 10, Nr. 3, pp. 433-457, May-June 1998.

[22] K.S. Candan, F. Golshani, S. Panchanathan, and Y.-C. Park. *VIMOS: A Video Mosaic for Spatio-Temporal Representation of Visual Information,* IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI) 98, April 5-7, 1998, Tucson, Arizona, 1998.

[23] T.H. Cormen, C.E. Leiserson and R.L. Rivest, "Introduction to Algorithms", McGraw Hill Publishers, 1989.

[24] W.-S. Li, K.S.Candan, K.Hirata, and Y. Hara *SEMCOG: An Object-Based Image Retrieval System and its Visual Query Interface.* Demonstration at the SIGMOD Conference on Management of Data, Tucson, Arizona, May 97. Also appeared in SIGMOD Record 26(2), June 1997.

[25] W.-S. Li, K.S. Candan, K.Hirata, and Y. Hara *IFQ: A Visual Query Interface for Object-based Image Retrieval.* Demonstration at the 1997 ACM CHI Conference, 1997.

[26] Stefano Ceri and Jennifer Widom. Deriving Production Rules for Incremental View Maintenance. In *17th VLDB*, 1991.

[27] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms*, McGraw Hill Publishers.

[28] Isabel F. Cruz. *DOODLE: A Visual Language for Object-Oriented Databases.* SIGMOD Conference 1992, pp. 71-80, 1992.

[29] Isabel F. Cruz, Michael Averbuch, Wendy T. Lucas, Melissa Radzyminski, and Kirby Zhang. *Delaunay: A Database Visualization System.* SIGMOD Conference 1997, pp. 510-513, 1997.

[30] U. Dayal. *Queries and Views in a Object-Oriented Databases.* In *Int. Workshop on Database Programming Languages*, 1989.

[31] R. Fagin and E.L. Wimmers. *Incorporating User Preferences in Multimedia Queries*, Int. Conference on Database Theory, 1997.

[32] C. Faloutsos.*Searching Multimedia Databases by Content*, Kluwer Academic Publishers, 1996.

[33] C. Faloutsos and H.V. Jagadish. *Hybrid Index Organizations for Text Databases*, Proc. EDBT-92, pps 310–327, 1992.

[34] C. A. Gerlhof and A. Kemper. *A Multi-Threaded Architecture for Prefetching in Object Bases.* EDBT 1994, pp. 351-364, 1994. Also in Lecture Notes in Computer Science (LNCS), vol. 779, pages 351-364, Cambridge, England, March 1994, Springer-Verlag.

[35] G. Golovchinsky. *Queries? Links? Is there a Difference?* In Proceedings of CHI '97 (Atlanta, GA, March 22-28). ACM Press, pp. 407-414, 1997.

[36] F. Hillier and G. Lieberman. *Operations Research*, Holden-Day, 1974.

[37] E. Hwang, B. Prabhakaran and V.S.Subrahmanian. *Distributed Video Presentations.* accepted for publication in IEEE intl. Conf. on Data Engineering, 1998.

[38] M.Y. Kim and J. Song. *Hyperstories: Combining Time, Space and Asynchrony in Multimedia Documents*, RC19277(83726) (revised 1995) IBM Computer Science/Mathematics Research Report, 1993.

[39] M.Y. Kim and J. Song. *Multimedia Documents with Elastic Time*, ACM Multimedia Conference '95, 1995.

[40] A. Kraiss and G. Weikum. *Integrated Document Caching and Prefetching in Storage Hierarchies Based on Markov-Chain Predictions.* VLDB Journal 7(3), pp. 141-162, 1998.

[41] W.-S. Li and K.S. Candan. *SEMCOG: A Hybrid Object-based Image Database System and Its Modeling, Language, and Query Processin* to appear in Proceedings of The 14th International Conference on Data Engineering, Orlando, Florida, USA, February 23 - 27, 1998.

[42] W.-S. Li, K.S. Candan, K. Hirata, and Y. Hara. *Facilitating Multimedia Database Exploration through Visual Interfaces and Perpetual Query Reformulations*, VLDB'97, 1997.

[43] T.D.C. Little and A. Ghafoor. *Synchronization and Storage Models for Multimedia Objects*, IEEE J. on Selected Areas of Communications, vol. 8, no. 3, April 1990, pp. 413-427. April 1990.

[44] D. Makaroff and R. T. Ng. *Schemes for Implementing Buffer Sharing in Continuous Media Systems*, Information Systems, 19, 4, pps 33–54, 1994.

[45] S. Marcus and V.S. Subrahmanian. Towards a Theory of Multimedia Database Systems. in: *Multimedia Database Systems: Issues and Research Directions* (eds. V.S.Subrahmanian and S. Jajodia), Springer Verlag,1995.

[46] S. Marcus and V.S. Subrahmanian. *Foundations of Multimedia Database Systems*, Journal of the ACM, Vol. 43, 3, pps 474–523, 1996.

[47] I. S. Mumick. *Query Optimization in Deductive and Relational Databases.* Ph.D. Thesis, Stanford University, CA 94305, 1991.

[48] W. Niblack, et. al. *The QBIC Project: Querying Images by Content Using Color, Texture and Shape*, IBM Research Report, Feb. 1993.

[49] N.J. Nilsson. *Problem Solving Methods in Artificial Intelligence.* McGraw-Hill Book Co, New york, 1971.

[50] R.T. Ng and J. Yang. *An Analysis of Buffer Sharing and Prefetching Techniques for Multimedia Systems.* Multimedia Systems 4(2), pp. 55-69, 1996.

[51] G. Özsoyoglu, V. Hakkoymaz, and J.D. Kraft. *Automating the Assembly of Presentations from Multimedia Databases*, Twelfth International Conference on Data Engineering, pp. 593-601, New Orleans, February 1996.

[52] V.E. Ogle and M. Stonebraker. *Chabot: Retrieval from a Relational Database of Images*, IEEE Computer, Sep. 1995.

[53] E. Oomoto and K. Tanaka. *OVID: Design and Implementation of a Video-Object Database System*, IEEE Trans. on Knowledge and Data Engineering, 5, 4, pps 629–643, 1993.

[54] B. Ozden, R. Rastogi, and A. Silberschatz. *A Low-Cost Storage Server for Movie-on-Demand Databases*, Proc. 20th Conf. on Very Large Databases, pps 594–605, Santiago, Chile, 1994.

[55] B. Ozden, R. Rastogi, and A. Silberschatz. *Research Issues in Multimedia Storage Servers*, ACM Computing Surveys, December 1995.

[56] R.H. Patterson, G.A. Gibson, E. Ginting, D. Stodolsky, and J. Zelenka. *Informed Prefetching and Caching.* SOSP 1995, pp. 79-95, 1995.

[57] Claudio S. Pinhanez, Kenji Mase, and Aaron F. Bobick. *Interval Scripts: a Design Paradigm for Story-Based Interactive Systems.* CHI 1997, pp. 287-294, 1997.

[58] Claudio S. Pinhanez and Aaron F. Bobick. *Fast Constraint Propagation on Specialized Allen Networks and its Application to Action Recognition and Control.* Submitted to AAAI'98. Perceptual Computing Tech. Rep. No. 456. January 1998.

[59] S.V. Raghavan, B. Prabhakaran and Satish K. Tripathi. *Synchronization Representation and Traffic Source Modeling in Orchestrated Presentation*, Special issue on Multimedia Synchronization, IEEE Journal on Selected Areas in Communication, January 1995.

[60] P. Venkat Rangan and H. Vin. *Designing File Systems for Digital Audio and Video*, Proc, 13th ACM Symp. on Operation Systems Principles, pps 69–79, 1991.

[61] P. Venkat Rangan, H. Vin and S. Ramanathan. *Designing an On-Demand Multimedia Service*, IEEE Communications Magazine, pps 56–64, July 1992.

[62] N. Roussopoulos, C. Faloutsos and T. Sellis. *An Efficient Pictorial Database System for PSQL*, IEEE Transactions on Software Engineering, 14, 5, pps 639–650, 1988.

[63] Y. Rui, T.S. Huang, and S. Mehrotra. *Content-based Image Retireval with Relevance Feedback in MARS*, IEEE Int. Conf. on Image Processing, 1997.

[64] V.S. Subrahmanian. *Principles of Multimedia Database Systems*, Morgan Kaufman Press, Dec. 1997.

[65] A. Sheth and V. Kashya. *Media-independent Correlation of Information: What? How?* Proceedings of First IEEE Metadata Conference, April 1996.

[66] T.K. Shih, S.K.C. Lo, S.-J. Fu, and J.B. Chang. *Using Interval Logic and Inference Rules for the Automatic Generation of Multimedia Presentations*, IEEE International Conference on Multimedia Computing and Systems '96, pp. 425-428, Hiroshima, June 1996.

[67] M. Vilain and H. Kautz. *Constraint Propagation Algorithms for Temporal Reasoning*, Proceedings of AAAI-86, Artificial Intelligence, pp. 377-382, August 1986.

[68] C.T. Yu and W. Meng. *Principles of Database Query Processing for Advanced Applications.* Morgan Kauffman Publishers, 1998.

[69] E. Wold, T. Blum, D. Keislar, and J. Wheaton. *Content-based Classification, Search, and Retrieval of Audio*, IEEE Multimedia Magazine 1996.

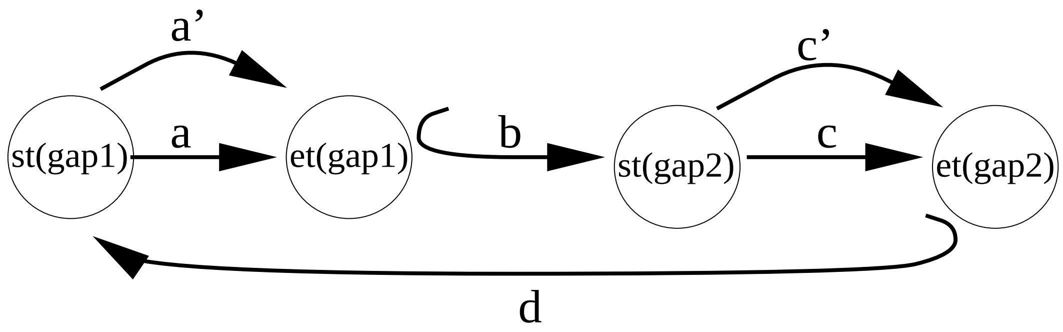



Figure 17: Negative cycle(s) caused by presentation gaps

# Appendix A

## Proof of Theorem 3.1

### Balanced Gap Division in Non-Overlapping Dependent Gap Sets

Let $G = \{gap_1, gap_2, \ldots, gap_g\}$ be a dependent gap set in the presentation. Let $st(gap_i)$ denote the node (in the constraint graph) corresponding to the start time of $gap_i$, and let $et(gap_i)$ denote the node corresponding to the end time of $gap_i$.

Since, $G$ is a dependent gap set, there exists a negative cycle of the form

$$st(gap_1) \rightarrow_1 et(gap_1) \hookrightarrow_1 \ldots st(gap_g) \rightarrow_g et(gap_g) \hookrightarrow_g st(gap_1),$$

where $\rightarrow$ denotes an edge created at step 5 of the algorithm described in Section 3.5.1, and $\hookrightarrow$ denotes a path (not passing through the other nodes in the negative cycle).

Since $G$ does not overlap with any other dependent gap set, the edges denoted by $\rightarrow$ are not included in any other negative cycle (except maybe in other negative cycles that pass through the exactly same set of gaps); furthermore, the paths denoted by $\hookrightarrow$ do not contain any $\rightarrow$ corresponding to the length specification of a gap not in $G$. Note that between each $et(gap_i)$ and $st(gap_{i+1})$, there may be more than one possible $\hookrightarrow_i$s. In the following discussion, since the shortest path corresponds to the most restrictive constraint set, we use the shortest such $\hookrightarrow_i$. We call the graph created by the $\hookrightarrow_i$s the *base* graph of $G$.

Figure 17 shows the constraint graph corresponding to a dependent gap set that has two gaps, $gap_1$ and $gap_2$.

Note that for each gap $gap_i$, the algorithm inserts a set of constraints of the form $c_{i,n} = \{et(gap_i) - st(gap_i) \leq n \times mL_{max}/gran\}$ with priority n, where $1 < n < gran$ (in Figure 17, $gran$ is equal to 2). Note that since $mL_i(\leq mL_{max})$ is the maximum possible length of gap $gap_i$, $c_{i,gran} = \{et(gap_i) - st(gap_i) \leq mL_i\}$ can never be in a negative cycle.

Here, we assume that the edges in the base graph have priorities larger than the priorities of $c_{i,n}$s ($1 < i < g$ and $1 < n < gran$).

Let $l_{base}$ be the *base length* of the negative cycle, where the base length is defined as the length of the negative cycle if all gaps were of 0 length. For instance, in Figure 18, the base length is $-40$.

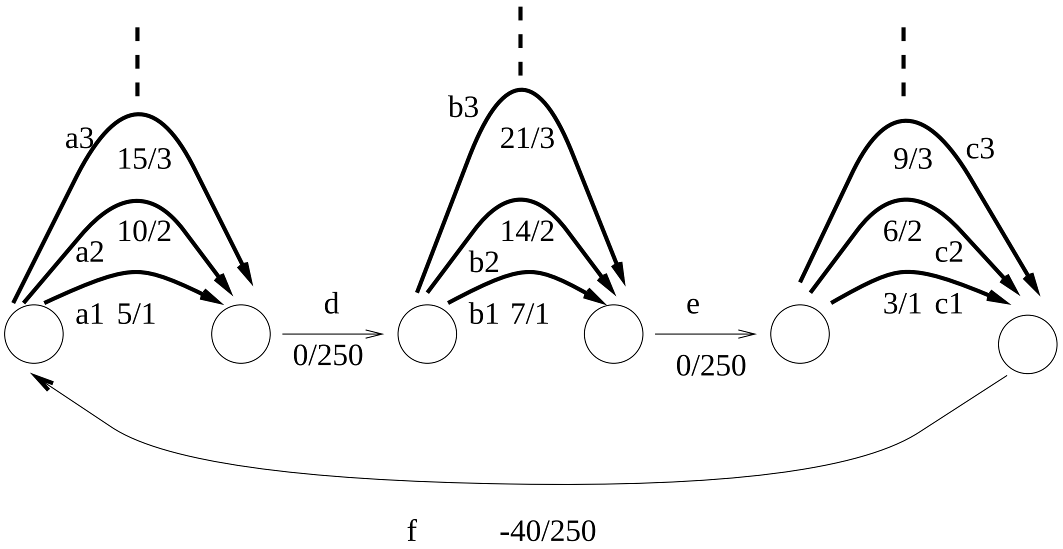



Figure 18: Edge weights and priorities that correspond to a dependent gap set (a label of the form $x/y$ means that the edge has $x$ weight and $y$ priority)

Let $\Delta_{unit}$ be the *unit increment* of the negative cycle:

$$\Delta_{unit} = \sum_{i=1}^{g} \frac{mL_{max}}{gran}$$

Then, we can define *balance level* (bl) of the graph as $\lceil -l_{base}/\Delta_{unit} \rceil$.

Note that the cycle which consists of $c_{i,bl}$s $(1 \leq i \leq g)$ and the base graph is non-negative (we call this cycle *safe cycle*). Therefore, the edges in $C_{safe} = \{c_{i,j} | (1 \leq i \leq g) \texttt{ and } (bl \leq j \leq gran)\}$ can not be removed by a priority optimal relaxation: each negative cycle that includes a constraint in $C_{safe}$ will also contain at least one constraint with priority less than $bl$, and such a low-priority constraint will be chosen for removal.

Let $lowest = \{l_1, l_2, \ldots, l_g\}$ be the set of edges, where $l_i$ is the edge with the lowest priority in gap $gap_i$. Using the above facts, we can see that every edge $l_j$ in *lowest* is in at least one negative cycle (as long as the corresponding graph has a negative cycle). For instance, in Figure 18, $lowest = \{a_1, b_1, c_1\}$. On the other hand, if $b_1$ get removed from the graph (by relaxation), then $lowest = \{a_1, b_2, c_1\}$.

Note that, if there exists a negative cycle in the graph, then *lowest* constitutes a negative cycle as well. And if $lowest = C_{safe}$, then the graph has no negative cycle.

If $lowest = C_{safe}$, then the gap sizes corresponding to the constraints on the base cycle satisfy

$$(bl - 1) \times \frac{mL_{max}}{gran} < size(gap_i) \leq bl \times \frac{mL_{max}}{gran}.$$

Let $gap_i$ and $gap_j$ be two gaps such that $size(gap_i) \geq size(gap_j)$. Then, if $bl \neq 1$ gap sizes corresponding to the constraints on the base cycle, satisfies

$$1 \leq \frac{size(gap_i)}{size(gap_j)} < \frac{bl}{bl - 1}, \qquad (1 \leq i, j \leq g).$$

On the other hand, $lowest$ may contain edges more restrictive than those are in $C_{safe}$: The length of the safe cycle is $l_{safe} = bl \times \Delta_{unit} + l_{base}$. Note that

$$\begin{array}{rcl} \frac{-l_{base}}{\Delta_{unit}} \leq & bl & < \frac{-l_{base}}{\Delta_{unit}} + 1, \\ -l_{base} \leq & bl \times \Delta_{unit} & < -l_{base} + \Delta_{unit}, \text{ and} \\ 0 \leq & bl \times \Delta_{unit} + l_{base} & < \Delta_{unit}. \end{array}$$

Since, the length ($l_{safe}$) of the safe cycle is positive, there may be a $lowest$ with edges more restrictive then $C_{safe}$, yet yielding a non-negative cycle. Let $lowest^+$ denote such a set. Let also $lowest^+_{min}$ denote the $lowest^+$ with the most restrictive edges.

Let $lowest^+_{min}$ be $\{c_{1,p_1}, c_{2,p_2}, \ldots, c_{g,p_g}\}$ (if for all $i$, $p_i$ is equal to $bl$, then $lowest^+_{min} = C_{safe}$). Then, we have

$$\begin{array}{rcl} 0 \leq & \sum_{i=0}^{g} (bl - p_i) \times \frac{mL_{max}}{gran} & < \Delta_{unit}, \\ \sum_{i=0}^{g} p_i \times \frac{mL_{max}}{gran} \leq & \sum_{i=0}^{g} bl \times \frac{mL_{max}}{gran} & < \sum_{i=0}^{g} (p_i + 1) \times \frac{mL_{max}}{gran} \\ \sum_{i=0}^{g} p_i \leq & \sum_{i=0}^{g} bl & < \sum_{i=0}^{g} (p_i + 1) \end{array}$$

Let $gap_i$ and $gap_j$ be two gaps such that $size(gap_i) \geq size(gap_j)$. If $bl > g$, then

$$\begin{array}{rcl} \frac{size(gap_i)}{size(gap_j)} & < & \frac{bl \times \frac{mL_{max}}{gran}}{(bl - g) \times \frac{mL_{max}}{gran}}, \\ & < & \frac{bl}{bl - g}. \end{array}$$

On the other hand,

$$\begin{array}{rcl} \frac{bl}{bl - g} & = & \frac{\lceil \frac{-l_{base}}{\Delta_{unit}} \rceil}{\lceil \frac{-l_{base}}{\Delta_{unit}} \rceil - g} \\ & < & \frac{\frac{-l_{base}}{\Delta_{unit}} + 1}{\frac{-l_{base}}{\Delta_{unit}} - g} \end{array}$$

Hence,

$$\begin{array}{rcl} \frac{size(gap_i)}{size(gap_j)} & < & \frac{-l_{base} + 1}{-l_{base} - g \times \Delta_{unit}}, \text{ or} \\ & < & \frac{-l_{base} + g \times \frac{mL_{max}}{gran}}{-l_{base} - g^2 \times \frac{mL_{max}}{gran}}. \end{array}$$

Note that as $-l_{base}$ or $gran$ increases, $\frac{size(gap_i)}{size(gap_j)}$ gets closer to 1.

### Balanced Gap Division in Overlapping Dependent Gap Sets

Let

$G_1 = \{gap_{(1),1}, gap_{(1),2}, \ldots, gap_{(1),g_{(1)}}, gap_{(1,2),1}, gap_{(1,2),2}, \ldots, gap_{(1,2),g_{(1,2)}}\}$

and

$G_2 = \{gap_{(2),1}, gap_{(2),2}, \ldots, gap_{(2),g_{(2)}}, gap_{(1,2),1}, gap_{(1,2),2}, \ldots, gap_{(1,2),g_{(1,2)}}\}$

be two overlapping dependent gap sets ($G_0 = G_1 \bigcap G_2 = \{gap_{(1,2),1}, gap_{(1,2),2}, \ldots, gap_{(1,2),g_{(1,2)}}\}$).

If $G = \{gap_1, gap_2, \ldots, gap_g\}$ is a dependent gap set, then there exists a negative cycle of the form

$$st(gap_1) \rightarrow_1 et(gap_1) \hookrightarrow_1 st(gap_g) \rightarrow_g et(gap_g) \hookrightarrow_g st(gap_1),$$

where $\rightarrow$ denotes an edge created at step 5 of the algorithm described in Section 3.5.1, and $\hookrightarrow$ denotes a path (not passing through the other nodes in the negative cycle).

Hence, both $G_1$ and $G_2$ has such a cycle. Using the discussion in the previous proof for the non-overlapping case, we can see that for each $G_i$ (if we do not consider the other set), there exists a solution such that if $gap_1$ and $gap_2$ in $G_i$, then

$$1 \leq \frac{size(gap_1)}{size(gap_2)} \simeq 1 \tag{1}$$

On the other hand, again using the previous discussion, we can also see that $size(gap_j)$ is dependent on the base length of the corresponding negative cycle.

We can divide the two dependent gap sets, $G_1$ and $G_2$, into three sets:

- $G'_1 = \{gap_{(1),1}, gap_{(1),2}, \ldots, gap_{(1),g_{(1)}}\}$,
- $G'_2 = \{gap_{(2),1}, gap_{(2),2}, \ldots, gap_{(2),g_{(2)}}\}$, and
- $G_0 = \{gap_{(0),1}, gap_{(0),2}, \ldots, gap_{(0),g_{(0)}}\}$ (where $gap_{(0),i} = gap_{(1,2),i}$ and $g_{(0)} = g_{(1,2)}$).

Since, $G'_1$, $G'_2$, and $G_0$ are non-overlapping with each other (and with any other set), if they can be thought as dependent gap sets, then each will independently satisfy the above equation. Hence, the question is "can we treat $G'_1$, $G'_2$, and $G_0$ as non-overlapping dependent gap sets?"

The answer to this question is "yes". To see this note that

- although neither $G'_1$, $G'_2$, or $G_0$ constitutes a negative cycle alone, each of them is on at least one negative cycle;
- the proof in the previous section was not dependent on the particular order of the graphs in the given negative cycle. The proof is dependent on how restrictive (or how negative) the corresponding base cycle is; and
- as we discussed in the previous proof, when the algorithm finishes for the non-overlapping case, the *safe cycle*, which is non-negative, is guaranteed to be left intact.

Hence, at any point in time, there exists a path from $et(gap_{(i),g_{(i)}})$ to $st(gap_{(i),1})$, for $i \in \{0,1,2\}$. Note that we do not know the weight of this path; However, since there exists such a path, even we do not know the actual length of the cycle, or the actual final sizes, we can see the truth of the equation 1.

## Appendix B

We can prove the theorem by induction on the value of $s$.

### Base Case ($s = 1$)

If $s$ is equal to 1, then the recursive calls at the last step of the algorithm will not add any new nodes. Hence, the number of nodes created by the algorithm are those nodes created during the first iteration. Since (1) for each action object, the algorithm creates $s$ points, (2) there are $m$ action objects, and (3) $s = 1$, the number of nodes created is $m + 2$ (there are 2 extra nodes for the start and end times).

If $s = 1$, according to the theorem, there must be $m^1 + m^0 \times 2 = m + 2$ nodes. Hence, the theorem holds.

### Inductive Case

Let us assume that the theorem holds for all $s$ values less than or equal to $j$. Then, it also holds for $s = j + 1$:

Let us assume that, the algorithm is called for input $s = j + 1$. Then, the algorithm will create $(j + 1) \times m + 2$ nodes during the first iteration. Next, again during the first iteration, the algorithm will recursively call itself ($(j + 1) \times m$ times) for input $s = j$. Hence, using the inductive hypothesis, we can see that the total number of nodes in the graph is

$$(j+1) \times m + 2 + (j+1) \times m \times \sum_{i=1}^{j} \left( \frac{j!}{(j-i)!} \times m^i + \frac{j!}{(j-i+1)!} \times m^{i-1} \times 2 \right)$$

or

$$(j+1) \times m + 2 + \sum_{i=1}^{j} \left( \frac{(j+1)!}{(j-i)!} \times m^{i+1} + \frac{(j+1)!}{(j-i+1)!} \times m^i \times 2 \right)$$

or

$$(j+1) \times m + 2 + \sum_{i=2}^{j+1} \left( \frac{(j+1)!}{(j+1-i)!} \times m^i + \frac{(j+1)!}{(j+1-i+1)!} \times m^{i-1} \times 2 \right)$$

or

$$\sum_{i=1}^{j+1} \left( \frac{(j+1)!}{(j+1-i)!} \times m^i + \frac{(j+1)!}{(j+1-i+1)!} \times m^{i-1} \times 2 \right)$$

which proves the theorem.