AUTOMATIC SCHEDULING OF HYPERMEDIA DOCUMENTS WITH ELASTIC TIMES

M.T. MEDINA, C.C. RIBEIRO, AND L.F.G. SOARES

ABSTRACT. The problem of automatic scheduling hypermedia documents consists in finding the optimal starting times and durations of objects to be presented, to ensure spatial and temporal consistency of a presentation while respecting limits on shrinking and stretching the ideal duration of each object. The combinatorial nature of the minimization of the number of objects whose duration is modified makes it the most difficult objective to be tackled by optimization algorithms. We formulate this scheduling problem as a mixed integer programming problem and report some preliminary investigations. We propose an original approach to the minimization of the number of objects which are shrinked or stretched. A simple primal heuristic based on variable fixations along the solution of a sequence of linear relaxations of the mixed integer programming formulation is described. Computational experiments on realistic size problems are reported. The effectiveness of the heuristic in finding good approximate solutions within very small processing times makes of it a quite promising approach to be integrated within existing document formatters to perform compile time scheduling or even run time adjustments. We also discuss results obtained by Lagrangean relaxation and propose a dual heuristic using the modified costs, which consistently improves the solutions found by the primal heuristic.

1. MOTIVATION

Hypermedia systems should introduce several desirable facilities, among them supporting a rich set of capabilities for each media segment (e.g. presentation duration flexibility, alternatives for different presentations, and behavior changes during presentation); allowing the explicit definition of temporal relationships among media segments with non-deterministic times; and supporting a temporal formatting algorithm that considers non-determinism and allows correcting the presentation on-the-fly whenever unpredictable events (such as network delays and user interactions) occur.

A presentation event is defined by a media segment presentation. The specification of the duration of a presentation event is associated with a quadruplet $(d^{min}, d^{ideal}, d^{max}, c)$, where d^{min} is the minimum allowed presentation duration, d^{max} is the maximum allowed presentation duration, d^{ideal} is the presentation duration that gives the best presentation quality, and $c(.)$ is the cost function associated with shrinking or stretching the event duration of the presentation. The optimal duration d^{opt} is the presentation duration that will be tried by the document formatter. It should be computed so as to warrant spatial and temporal consistency of the document. The main focus of this paper consists in efficient algorithms for the

Date: February 2001.
Key words and phrases. Hypermedia, scheduling, heuristics, mixed-integer programming .

scheduling of hypermedia documents, requiring the determination of the starting time and the optimal duration of each presentation event. Due to its speed and effectiveness, the algorithm can also be used for online adjustments at execution time.

Values d^{min}, d^{ideal}, and d^{max} are usually known, but they can also be unspecified or even unpredictable. The cost function $c(.)$ may be a discrete function, not necessarily a linear or convex function. It is also usual to take $c(.) = 0$, e.g. when presenting a text event for a duration longer than d^{min}. The specification of the cost function $c(.)$ depends on a user subjective perception of a segment presentation in relation to other media segment presentation perceptions. For example, when presenting an audio in parallel with a video, an author should specify the cost of shrinking or stretching a video taking in account not only the degradation of the video quality itself, but also the relationship between this degradation and the audio degradation, i.e. the cost of shrinking or stretching the audio.

Spatio-temporal relationships among events can have causal or constraint semantics. Causual relationships are based on conditions and actions. Conditions are related to source events, while actions are operations that must be applied to target events. When a condition is satisfied, its associated action is fired. An example of a causal relation is: "when the presentation of video A finishes, start playing audio B". Constraint relations specify restrictions that must be satisfied during a presentation, without any causality involved. An example of a constraint relation is: "video A must finish its presentation at the same time that audio B starts is presentation". Although similar, the latter does not imply in starting audio A. The spatio-temporal formatter is responsible for controlling the document presentation based on its specification and on the environment description. As an example, Figure 1 displays the HyperProp [14] spatio-temporal formatter architecture, which is composed by four elements: the pre-compiler, the compiler, the executor and the viewer controllers (or simply controllers). Only the compiler and the executor are important for the discussion involved in this paper.

Considering the complete document presentation specification and the environment description, the compiler is responsible for generating the data structure used by the spatio-temporal formatter to build an execution plan (that will guide the scheduling of the object presentation). During the document authoring, the best presentation duration d^{opt} of each event is initially set to be equal to its ideal value. Then, it should be computed based on a scheduling model at compilation time, before presentation. Several objectives may be the goal of the scheduling model:

(1) minimize the total cost of shrinking and stretching objects;
(2) minimize the number of presentation events that will have a duration different than the ideal one;
(3) spread the changes in object duration proportionally to their duration; or
(4) find the shortest and longest time overall presentation duration (makespan), among others.

At execution time, mismatches between document specification and presentation can occur due to unpredictable events, such as network delays and user interactions. Run time mismatches are detected only when the document has been partially presented. This delay in detection limits the ability of a run time formatter to smooth out mismatches. Possible mismatches can be smoothed out at compilation time, improving the appearance of the resulting document. However, only at run

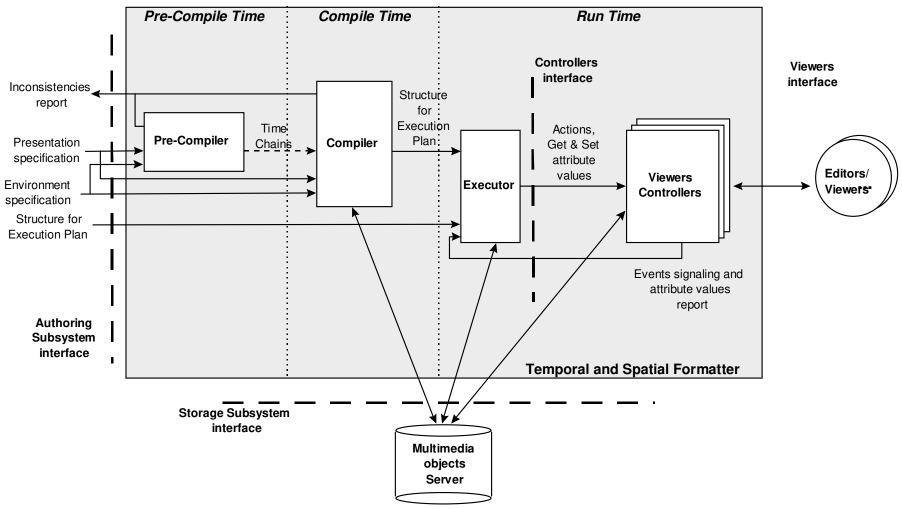

FIGURE 1. HyperProp spatio-temporal formatter architecture.

time unpredictable events can be handled. Accordingly, a formatter supporting both compile time and run time adjustments is needed.

The executor has to shrink or stretch object presentations to perform run time adjustments. The execution plan created by the compiler will guide the formatter in adjusting object durations on-the-fly, as well as in pre-fetching component contents to improve presentation quality and reduce the probability of temporal and spatial inconsistencies. Whenever run time adjustments are done, the best duration for objects remaining to be presented must be recomputed in real time. To speedup the optimization algorithm so as to perform scheduling decisions as fast as possible, some constraints can be relaxed. Several actions can be done to simplify real time computation, such as limiting the adjustments to a minimum number of objects or previewing possible adjustments and computing the new possible ideal presentation durations in advance. The difficulty of performing fast rescheduling computations during run time has led the very few systems that make time adjustmets to make them only at compile time.

2. PROBLEM STATEMENT

Objectives (1), (3), and (4) introduced in the previous section can be efficiently dealt with at compile time by linear programming models or even more sophisticated or faster approaches involving the optimization of convex functions [11].

Firefly [3, 4] was the first proposal of a document formatter to deal with both predictable and unpredictable events. At compile time, the proposed strategy computes a primary scheduling of the main part of the document (associated with predictable events) and several secondary schedulings (associated with separate groups of events which are fired by unpredictable events). Starting times of events within these groups are updated whenever they are fired by the user at execution time. Event durations and starting times within the main part of the document and within each group of separate events are computed by a simplex code solving a linear programming model minimizing shrinking and stretching costs of event durations. *Isis* [10] is another proposal using PERT and linear programming techniques to minimize shrinking and stretching costs with respect to ideal event durations. All computations are performed at compile time taking only predictable events into account. Finally, *Madeus* [2, 8, 9] also considers only predictable events to compute a preliminary scheduling of the whole document at compile time. Madeus is a constraint based system, which uses temporal constraint networks algorithms to incrementally detect temporal inconsistencies. The temporal specification of the document is used to update the scheduling at execution time. These proposals do not seem to have been integrated into actual document formatters. Only a few computational results illustrating their application to toy examples are reported in the literature.

Although conflicting, objectives (1) and (2) are the most appropriate. A suitable strategy to take both of them into account would consist in selecting the best solution with respect to objective (2), among all those which optimize (1). The combinatorial nature of the minimization of the number of objects whose duration is modified makes objective (2) the most difficult to be tackled by optimization algorithms.

In this work, we propose a new formulation and a heuristic to the problem of scheduling hypermedia documents, minimizing the number of objects which should be shrinked or stretched with respect to their ideal durations. This approach can be easily integrated within the above described bi-criteria strategy. In the next section, we give a mixed 0-1 integer programming formulation to this scheduling problem. This formulation can be trivially extended to take into account the other objectives. A simple primal heuristic based on variable fixations along the solution of a sequence of linear relaxations of this formulation is proposed in Section 4. Computational experiments on real size problems are reported. The effectiveness of the heuristic in finding suboptimal solutions within very small processing times makes it a quite promising approach to be integrated within existing document formatters to perform compile time scheduling or even run time adjustments. A dual heuristic using the Lagrangean multipliers and modified costs generated along the solution of the dual problem are described in Section 5. The dual heuristic consistently improves the solutions obtained by the primal one. Further extensions of this work and concluding remarks are drawn in the last section.

3. Formulation

Let $N_s = \{1, \ldots, n\}$ represent the set of synchronization points, where presentation events either start or stop. We represent a hypermedia document by a directed graph $G = [N, A]$, where the set of nodes $N = \{0\} \cup N_s \cup \{n+1\}$ is obtained from N_s by introducing two dummy nodes indexed by 0 and $n+1$ which represent, respectively, the beginning and the end of the presentation. The arc set A corresponds to the union of all presentation events (arcs with both endpoints in N_s) with dummy arcs originating at node 0 or terminating at node $n+1$. With each presentation event corresponding to an arc $(i, j) \in A$ with both $i, j \in \{1, \ldots, n\}$, we associate minimum, ideal, and maximum durations d_{ij}^{min}, d_{ij}^{ideal}, and d_{ij}^{max}, respectively. Accordingly, $d_{0j}^{min} = 0$, $d_{0j}^{ideal} = 0$ (or any other non-negative value), and $d_{0j}^{max} = \infty$ for all nodes $j \in \{1, \ldots, n\}$ from which the presentation can start. Equivalently,

$d^{min}_{i,n+1} = 0$, $d^{ideal}_{i,n+1} = 0$ (or, again, any other non-negative value), and $d^{max}_{i,n+1} = \infty$ for all nodes $i \in \{1, \ldots, n\}$ which can be the last one in the presentation.

A continuous variable $t_i \geq 0$ is associated with each node $i \in N$. This variable represents the time in which synchronization takes place at this node, i.e. the time in which one presentation event finishes and another one starts. The presentation starts at $t_0 = 0$ and finishes at t_{n+1}. The duration of each presentation event $(i,j) \in A$ is represented by a continuous variable $x_{ij} \in [d^{min}_{ij}, d^{max}_{ij}]$. Let y_{ij} be a binary variable associated with each presentation event $(i,j) \in A$, so that $y_{ij} = 1$ if the duration of event (i,j) is either shrinked or stretched with respect to its ideal duration d^{ideal}_{ij}; $y_{ij} = 0$ otherwise (i.e., the event is scheduled to be presented exactly with its ideal duration). Let $c_{ij} = 1$ for every presentation event $(i,j) \in A, i \neq 0, j \neq n+1$; $c_{ij} = 0$ otherwise. Then, the problem of automatic scheduling hypermedia documents (ASHD) minimizing the number of objects which should be shrinked or stretched with respect to their ideal durations can be formulated as

$$
\text{ASHD:} \left\{
\begin{array}{lrrlll}
z^* = \min & & \displaystyle\sum_{(i,j)\in A} c_{ij} y_{ij} & & & \\
\text{with:} & & & & & \\
& t_i - t_j + x_{ij} & & = 0 & \forall (i,j) \in A & (1) \\
& x_{ij} & -b_{ij} y_{ij} & \leq d^{ideal}_{ij} & \forall (i,j) \in A & (2) \\
& -x_{ij} & +a_{ij} y_{ij} & \leq -d^{ideal}_{ij} & \forall (i,j) \in A & (3) \\
& & y_{i,j} & \in \{0,1\} & \forall (i,j) \in A & (4) \\
& & t_i & \geq 0 & \forall i \in N \setminus \{0\} & \\
& & t_0 & = 0; & &
\end{array}
\right.
$$

where $b_{ij} = d^{max}_{ij} - d^{ideal}_{ij}$ and $a_{ij} = d^{min}_{ij} - d^{ideal}_{ij}$ are the maximum values by which the duration of event (i,j) can be, respectively, stretched or shrinked. Constraints (1) define time instants in which synchronization takes place. Constraints (2) and (3) together establish that the duration of presentation event (i,j) is either equal to d^{ideal}_{ij} in case $y_{ij} = 0$, or that it can assume any value within the interval $[d^{min}_{ij}, d^{max}_{ij}]$ otherwise.

The linear relaxation $\overline{\text{ASHD}}$ of the above problem is obtained by replacing constraints (4) by inequalities $0 \leq y_{ij} \leq 1$ for all $(i,j) \in A$. We note that whenever the linear relaxation $\overline{\text{ASHD}}$ is feasible, so is the original problem ASHD. A feasible solution of problem ASHD in 0-1 variables can be obtained from any feasible solution to the linear relaxation by rounding to one all fractional variables in the latter. Accordingly, the number of nonzero variables in any feasible solution to the linear relaxation gives an upper bound to the optimal solution of problem ASHD.

4. Primal heuristic

We present in Figure 2 the pseudo-code of the primal heuristic PH for problem ASHD. Each iteration starts by the solution of the linear relaxation $\overline{\text{ASHD}}$ of the current problem (first iteration, line 1; other iterations, line 24). Let $(\overline{y}, \overline{x}, \overline{t})$ be its optimal solution.

The loop from line 2 to 25 progressively fixes all y variables to either zero or one, depending on the value they assume in the current solution $\overline{y}$, and stops when all variables are integer. Each iteration begins in lines 3-4 by the fixation at one (resp. zero) of all variables y_{ij} equal to one (resp. zero) in the current solution. Next, the algorithm investigates the variable which is the closest to zero in the

procedure PH
1 Solve $\overline{\text{ASHD}}$ and let $(\overline{y}, \overline{x}, \overline{t})$ be its optimal solution;
2 **while** $\exists (i,j) \in A : 0 < \overline{y}_{ij} < 1$ **do**
3 Fix to one in ASHD all variables y_{ij} such that $\overline{y}_{ij} = 1$;
4 Fix to zero in ASHD all variables y_{ij} such that $\overline{y}_{ij} = 0$;
5 **if** $\exists (i,j) \in A, i \neq 0, j \neq n+1 : 0 < \overline{y}_{ij} \leq \alpha$
6 **then do**
7 $(s,t) \leftarrow \text{argmin}_{(i,j) \in A, i \neq 0, j \neq n+1} \{\overline{y}_{ij} : 0 < \overline{y}_{ij} \leq \alpha\}$;
8 Fix y_{st} to zero in ASHD;
9 **if** $\overline{\text{ASHD}}$ is not feasible **then** fix y_{st} to one in ASHD;
10 **else if** $\exists (i,j) \in A, i \neq 0, j \neq n+1 : 1 - \alpha \leq \overline{y}_{ij} < 1$
11 **then do**
12 $(s,t) \leftarrow \text{argmax}_{(i,j) \in A, i \neq 0, j \neq n+1} \{\overline{y}_{ij} : 1 - \alpha \leq \overline{y}_{ij} < 1\}$;
13 Fix y_{st} to one in ASHD;
14 **else do**
15 $(s,t) \leftarrow \text{argmin}_{(i,j) \in A, i \neq 0, j \neq n+1} \{\min\{\overline{y}_{ij}, 1 - \overline{y}_{ij}\}\}$;
16 **if** $y_{st} \leq 0.5$
17 **then do**
18 Fix y_{st} to zero in ASHD;
19 **if** $\overline{\text{ASHD}}$ is not feasible **then** fix y_{st} to one in ASHD;
20 **else** fix y_{st} to one in ASHD;
21 **end_if**;
22 **end_if**;
23 **end_if**;
24 Solve $\overline{\text{ASHD}}$ and let $(\overline{y}, \overline{x}, \overline{t})$ be its optimal solution;
25 **end_while**;
26 **return** $\overline{y}$;
end PH;

FIGURE 2. Pseudo-code of the primal heuristic for problem ASHD

current solution. For a given treshold value α given as a parameter, in lines 7-8 we tentatively fix at zero the variable whose current value is closest to zero and under this threshold. If the linear relaxation of the new, reduced problem is not feasible, then this variable is definitely fixed at one in line 8. Next, we investigate the variable closest to one in the current solution. If this variable is greater than or equal to $1 - \alpha$, then it is fixed at one in lines 12-13.

In case no variable can be fixed by any of the above two rules, starting in line 15 we consider the closest variable to an integer value (zero or one). If the current value of this variable is closer to one than to zero, then we set it at one in line 20; otherwise we tentatively set it at zero in line 18. Once again, if the linear relaxation of the new, reduced problem is not feasible, then it is definitely fixed at one in line in line 19. We observe that at least one variable is fixed at each iteration. The next iteration resumes in line 24 with the solution of the linear relaxation of the new, reduced problem obtained by variable fixation. In our implementation, we stop the application of the heuristic when only a few variables remain to be fixed and solve the residual problem exactly using a branch-and-bound code.

The primal heuristic PH above described was implemented in C and compiled with the version 2.95.2 of the gcc compiler. CPLEX version 5.0 was used as the LP solver. The computational experiments have been performed on a SunOS 5.5.1 UltraSPARC station with 256 Mbytes of RAM and a 167 MHz clock.

Test problems have been generated by Mahey and Bruno [12] as follows. Connected temporal graphs with 20, 30, 40, 50, 100, 500, and 1000 nodes have been randomly generated with the maximum allowed durations d_{ij}^{max} equal to 1000 or 100000 (with a flexibility of 20%, i.e. $d_{ij}^{ideal} - d_{ij}^{min} \leq 0.2 \times d_{ij}^{ideal}$ and $d_{ij}^{max} - d_{ij}^{ideal} \leq 0.2 \times d_{ij}^{ideal}$) for every arc. Different graphs have been generated with arc-node ratios of 200%, 400%, 600%, 800%, and 1000%.

Computational results are presented in Table 1. For each instance, we give its identification, the number of nodes (synchronization points plus two dummy nodes) and the number of arcs (presentation events plus at most $2n$ dummy arcs) in the graph, together with run statistics for both the primal heuristic and the exact solution of the original formulation ASHD: the value of the best solution found, the processing time in seconds, and the total number of CPLEX iterations performed in each case.

CPLEX found optimal solutions for instances a1 to b1 and d1. Exact optimal solutions have not been obtained and are not available for the remaining instances. In the case of instances b2, b3, c1, c5, and d2 to c3, CPLEX has been interrupted due to the size of the enumeration tree beeing larger than the available memory. Instances b4, b5, c2 to c4, and e4 to g5 have been interrupted after four hours of processing time.

The primal heuristic PH was quite successful in finding good approximate solutions within small computation times for realistic-size instances. Computation times were kept within reasonable values, except for the very large instances with 500 or more nodes, which clearly surpass the size of current real applications. Among the seven instances reported in the above table for which the exact optimum is known, the primal heuristic found the optimal solutions for two of them and was off by exactly one variable for two others. Integrality gaps between the value of the solutions found by the the primal heuristic and the linear relaxation are shown in Table 2. The average and the maximum gaps over all test instances are 54.4% and 59.6%, respectively. Since very large gaps are observed even in the case of small instances for which the primal heuristic found the exact optimal solution, we conjecture that the solutions provided by algorithm PH are quite good in most of the cases and that the linear programming or dual bounds are very poor. This would be an indication that the search for polyhedral cuts or valid inequalities should be pursued if one wants to develop efficient exact methods for this problem.

5. Dual heuristic

To derive the Lagrangean dual of problem ASHD, for each edge $(i,j) \in A$ we associate non-negative dual multipliers λ_{ij}^{+} and λ_{ij}^{-} to constraints of types (2) and (3), respectively. The Lagrangean function defined by the relaxation of constraints (2) and (3) is

$$L(x,y,t,\lambda^{+},\lambda^{-}) = \sum_{(i,j)\in A} c_{ij}y_{ij} + \lambda_{ij}^{+}(x_{ij} - b_{ij}y_{ij} - d_{ij}^{ideal}) + \lambda_{ij}^{-}(-x_{ij} + a_{ij}y_{ij} + d_{ij}^{ideal}),$$

TABLE 1. Computational results: Primal heuristic

			Best value		Time (s)		CPLEX iterations	
Instance	Nodes	Arcs	PH	Exact	PH	Exact	PH	Exact
a1	20	40	22	20	0.09	2.98	13	6037
a2	20	80	62	62	0.29	25.07	53	30938
a3	20	120	103	102	0.76	2328.75	93	2124483
a4	20	160	144	141	1.05	8212.52	132	4820268
a5	20	200	180	176	1.72	1169.16	167	624242
b1	30	60	32	31	0.16	9.82	23	14176
b2	30	120	93	≤ 90	0.70	* 10449.16	82	10540176
b3	30	180	155	≤ 151	1.46	* 14572.03	141	8864216
b4	30	240	210	≤ 208	2.64	> 4 hs	198	7022161
b5	30	300	272	≤ 271	3.87	> 4 hs	257	5723104
c1	40	80	45	≤ 42	0.28	* 11535.65	33	15082631
c2	40	160	125	≤ 124	1.05	> 4 hs	113	12554845
c3	40	240	203	≤ 205	2.64	> 4 hs	190	8701970
c4	40	320	283	≤ 282	4.71	> 4 hs	271	6864731
c5	40	400	363	≤ 367	7.54	* 12165.24	349	3815599
d1	50	100	53	53	0.33	4246.48	43	4242544
d2	50	200	156	≤ 154	1.54	* 9190.92	141	5952805
d3	50	300	248	≤ 247	3.90	* 10122.71	235	4683837
d4	50	400	354	≤ 358	7.07	* 12148.63	330	3912679
d5	50	500	453	≤ 452	11.62	* 13041.90	430	3172888
e1	100	200	103	≤ 102	1.29	* 10087.15	90	8020126
e2	100	400	303	≤ 304	6.74	* 9715.30	287	2647110
e3	100	600	506	≤ 520	16.69	* 11597.95	480	2164665
e4	100	800	702	≤ 709	30.47	> 4 hs	668	2135646
e5	100	1000	906	≤ 908	51.59	> 4 hs	868	2020568
f1	500	1000	515	≤ 520	42.01	> 4 hs	485	3035630
f2	500	2000	1521	≤ 1553	241.58	> 4 hs	1463	981708
f3	500	3000	2541	≤ 2551	782.12	> 4 hs	2478	919510
f4	500	4000	3533	≤ 3541	987.47	> 4 hs	3395	610230
f5	500	5000	4519	≤ 4553	1659.88	> 4 hs	4354	581753
g1	1000	2000	1045	≤ 1177	208.05	> 4 hs	978	2514172
g2	1000	4000	3043	≤ 3209	1112.78	> 4 hs	2915	1053983
g3	1000	6000	5060	≤ 5136	2563.94	> 4 hs	4891	386816
g4	1000	8000	7053	≤ 7150	5080.02	> 4 hs	6839	2962588
g5	1000	10000	9038	≤ 9062	7332.10	> 4 hs	8690	175434

* maximum size of the enumeration tree exceeded

with $c_{ij} = 1$ if $i \neq 0$ and $j \neq n+1$; $c_{ij} = 0$ otherwise, $\forall (i,j) \in A$. Rearranging the terms in the above definition, we obtain

$$L(x, y, t, \lambda^+, \lambda^-) = \sum_{(i,j) \in A} \Lambda^1_{ij} y_{ij} + \Lambda^2_{ij} x_{ij} + \Lambda^3_{ij} d^{ideal}_{ij},$$

TABLE 2. Duality gaps

Instance	PH	$\overline{z}$	Gap (%)
a1	22	8.88	59.6
a2	62	32.36	47.8
a3	103	44.93	56.4
a4	144	63.76	55.7
a5	180	78.64	56.3
b1	32	16.91	47.1
b2	93	41.70	55.2
b3	155	69.68	55.0
b4	210	95.68	54.4
b5	272	117.67	56.7
c1	45	19.20	57.3
c2	125	59.63	52.3
c3	203	89.63	55.8
c4	283	118.16	58.2
c5	363	165.65	54.3
d1	53	25.72	51.5
d2	156	74.42	52.3
d3	248	114.23	53.9
d4	354	166.12	53.1
d5	453	205.39	54.6
e1	103	48.32	53.1
e2	303	137.70	54.5
e3	506	232.34	54.1
e4	702	323.19	54.0
e5	906	404.82	55.3
f1	515	237.79	53.8
f2	1521	678.83	55.4
f3	2541	1109.97	56.3
f4	3533	1616.21	54.2
f5	4519	2063.82	54.3
g1	1045	479.95	54.1
g2	3043	1368.76	55.0
g3	5060	2274.75	55.0
g4	7053	3157.54	55.2
g5	9038	4143.82	53.8

with $\Lambda^1_{ij} = (c_{ij} + a_{ij}\lambda^-_{ij} - b_{ij}\lambda^+_{ij})$, $\Lambda^2_{ij} = \lambda^+_{ij} - \lambda^-_{ij}$, and $\Lambda^3_{ij} = \lambda^-_{ij} - \lambda^+_{ij}$. The dual function $w(\lambda^+, \lambda^-)$ is defined as:

$$\left\{\begin{array}{llll} w(\lambda^+,\lambda^-) = \text{minimum} & L(x,y,t,\lambda^+,\lambda^-) & & \\ \text{with:} & & & \\ & t_i - t_j + x_{ij} = 0 & \forall (i,j) \in A & (1) \\ & y_{i,j} \in \{0,1\} & \forall (i,j) \in A & (4) \\ & d^{min}_{ij} \leq x_{ij} \leq d^{max}_{ij} & \forall (i,j) \in A & (5) \\ & t_i \geq 0 & \forall i \in N \setminus \{0\} & \\ & t_0 = 0. & & \end{array}\right.$$

Constraints (5) are reincorporated into the dual function so as to ensure the associated problem is bounded. A lower bound to the optimal value z^* can then be given by the maximum of the dual problem, i.e.

$$\text{D(ASDH)}: \quad z^D = \text{maximum}_{\lambda^+,\lambda^- \geq 0} \{w(\lambda^+,\lambda^-)\} \leq z^*.$$

For any non-negative dual multiplier vectors $\lambda^+ = (\lambda^+_{ij})_{(i,j)\in A}$ and $\lambda^- = (\lambda^-_{ij})_{(i,j)\in A}$, the solutions x^{λ^+,λ^-}, y^{λ^+,λ^-}, and t^{λ^+,λ^-} leading to the dual function value $w(\lambda^+,\lambda^-) = L(x^{\lambda^+,\lambda^-}, y^{\lambda^+,\lambda^-}, t^{\lambda^+,\lambda^-}, \lambda^+, \lambda^-)$ can be easily obtained as follows:

Step 1: For each $(i,j) \in A$, set $y^{\lambda^+,\lambda^-}_{ij} = 1$ if $\Lambda^1_{ij} < 0$; $y^{\lambda^+,\lambda^-}_{ij} = 0$ otherwise.

Step 2: Obtain $x^{\lambda^+,\lambda^-} = (x^{\lambda^+,\lambda^-}_{ij})_{(i,j)\in A}$ and $t^{\lambda^+,\lambda^-} = (t^{\lambda^+,\lambda^-}_{ij})_{(i,j)\in A}$ as the optimal solution to the linear program

$$\left\{\begin{array}{lrll}
\text{minimum} & \sum_{(i,j)\in A} \Lambda^2_{ij} x_{ij} & & \\
\text{with:} & & & \\
 & t_i - t_j + x_{ij} = 0 & \forall (i,j) \in A & (1) \\
 & d^{min}_{ij} \leq x_{ij} \leq d^{max}_{ij} & \forall (i,j) \in A & (5) \\
 & t_i \geq 0 & \forall i \in N \setminus \{0\} & \\
 & t_0 = 0. & &
\end{array}\right.$$

The dual problem can be solved by subgradient optimization (see e.g. [13] for an overview). We recall that for any non-negative dual multiplier vectors $\lambda^+ = (\lambda^+_{ij})_{(i,j)\in A}$ and $\lambda^- = (\lambda^-_{ij})_{(i,j)\in A}$, $\gamma^{\lambda^+,\lambda^-} = (\gamma^+,\gamma^-)$ is a subgradient of the dual function $w(.,.)$ at (λ^+,λ^-), where

$$\gamma^+_{ij} = x^{\lambda^+,\lambda^-}_{ij} - b_{ij} y^{\lambda^+,\lambda^-}_{ij} - d^{ideal}_{ij}$$

and

$$\gamma^-_{ij} = -x^{\lambda^+,\lambda^-}_{ij} + a_{ij} y^{\lambda^+,\lambda^-}_{ij} + d^{ideal}_{ij}$$

for any $(i,j) \in A$. The dual problem can be solved by a subgradient-based approach, such as the well known and widely used algorithm of Held et al [6, 7] or the more recent volume algorithm of Barahona and Anbil [1].

We propose a dual heuristic for problem ASHD, based on the combination of the primal heuristic with the solution of the dual problem by subgradient optimization. We refer to the pseudo-code of algorithm `DH` described in Figure 3, which follows a scheme similar to that described by Caprara, Fischetti, and Toth [5]. According with this strategy, we use the dual multipliers obtained along the solution of the Lagrangean dual to periodically generate reduced costs c'_{ij} associated with the integer variables y_{ij}. The primal heuristic `PH` is succcessively applied to problem ASHD, each time with new reduced costs c'_{ij} replacing the original costs $c_{ij} = 1$ associated with the presentation events. The best feasible solution is retained.

The dual heuristic `DH` starts in line 1 by the application of the primal heuristic yielding an integer solution $\overline{y}$. An upper bound $z_{best} = \sum_{(i,j)\in A} c_{ij}\overline{y}_{ij}$ to z^* is computed in line 2. Variable *total_iterations* counts the total number of iterations and is initialized in line 3. The loop from line 4 to 14 is performed until one of the stopping criteria is attained. Starting from the current set of dual multipliers, *block_iterations* iterations of the subgradient algorithm are performed in line

5. Reduced costs c' are computed in line 6, using the current dual multipliers. The primal heuristic PH is applied in line 7 to the current problem after every *block_iterations* subgradient iterations, using the reduced costs c'. In case the new solution $\overline{y}$ improves the current upper bound z_{best}, the latter and the best solution found are updated respectively in lines 10 and 11. The total number of iterations is updated in line 13. The best solution found by successive applications of the primal heuristic is returned in line 15.

procedure DH
1 Apply the primal heuristic PH to compute a feasible solution $\overline{y}$;
2 Set the upper bound $z_{best} \leftarrow \sum_{(i,j)\in A} c_{ij}\overline{y}_{ij}$ and $y_{best} \leftarrow \overline{y}$;
3 *total_iterations* $\leftarrow 0$;
4 **while** (stoping criterion is not attained) **do**
5 Perform *block_iterations* subgradient iterations;
6 Compute reduced costs c' using the dual multipliers (λ^+, λ^-);
7 Apply the primal heuristic PH using the reduced costs c' to compute a new feasible solution $\overline{y}$;
8 **if** $\sum_{(i,j)\in A} c_{ij}\overline{y}_{ij} < z_{best}$
9 **then do**
10 $z_{best} \leftarrow \sum_{(i,j)\in A} c_{ij}\overline{y}_{ij}$;
11 $y_{best} \leftarrow \overline{y}$;
12 **end_then**;
13 *total_iterations* $\leftarrow$ *total_iterations* + *block_iterations*;
14 **end_while**;
15 **return** y_{best};
end DH;

FIGURE 3. Pseudo-code of the dual heuristic to problem ASHD

Reduced costs are computed in line 6 of the above procedure DH according with two different schemes. At each iteration, one of them is randomly selected with probability 1/2. The first scheme is based on the use of the Lagrangean multipliers (λ^+, λ^-) to generate classical reduced costs $c'_{ij} = c_{ij} + a_{ij}\lambda^-_{ij} - b_{ij}\lambda^+_{ij}$ from the Lagrangean function. The second scheme makes use of the primal approximation $\hat{y}$ computed by the subgradient algorithm. In this case, the reduced costs are such that $c'_{ij} = (1 - \hat{y}_{ij})c_{ij}$, so as that variables close to their upper limits will have very small reduced costs and will very likely appear in the solution produced by the primal heuristic.

Two stopping criteria are used. The heuristic stops after a total of 1200 subgradient iterations or when the relative difference between the dual solution values at two consecutive iterations becomes less than 10^{-4}, whatever happens first. The value of the parameter *block_iterations* was set at 240 after some preliminary computational experiments, along which different values in the range from 30 to 240 have been investigated. Although the variation in solution quality is very small, the computation times are much smaller when higher values of *block_iterations* are used, since in this case there are fewer calls to the primal heuristic.

Computational results obtained with these settings are reported in Table 3. For each instance, we give the best solution found by each heuristic, together with the associated computation times in seconds. For the sake of completeness, we also give the optimal value of the linear relaxation and that of the optimal integer solution (whenever known). We notice that the dual heuristic consistently improves upon the solutions found by the primal one, finding better solutions than the latter for 17 out of the 35 problems. Since the computation times observed for the dual heuristic are somehow large, the sequel of this work will contemplate both faster construction algorithms and local search procedures for problem ASHD.

TABLE 3. Computational results: Dual heuristic

			Linear	DH		PH		Exact
Instance	Nodes	Arcs	$\overline{z}$	z_{best}	s	z_{best}	s	z^*
a2	20	40	8.88	22	4.94	22	0.09	20
a2	20	80	32.36	62	9.45	62	0.29	62
a3	20	120	44.93	103	14.25	103	0.76	102
a4	20	160	63.76	142	18.23	144	1.05	141
a5	20	200	78.64	178	24.88	180	1.72	176
b1	30	60	16.91	31	8.35	32	0.16	31
b2	30	120	41.70	93	14.49	93	0.76	$\leq$ 90
b3	30	180	69.68	154	21.87	155	1.46	$\leq$ 151
b4	30	240	95.68	209	31.88	210	2.64	$\leq$ 208
b5	30	300	117.67	272	46.26	272	3.87	$\leq$ 271
c1	40	80	19.20	45	2.38	45	0.28	$\leq$ 42
c2	40	160	59.63	125	19.83	125	1.05	$\leq$ 124
c3	40	240	89.58	202	32.83	203	2.64	$\leq$ 205
c4	40	320	118.16	280	49.20	283	4.71	$\leq$ 282
c5	40	400	165.65	362	67.19	363	7.54	$\leq$ 367
d1	50	100	25.72	53	7.09	53	0.33	53
d2	50	200	74.42	154	28.63	156	1.54	$\leq$ 154
d3	50	300	114.23	248	50.78	248	3.90	$\leq$ 247
d4	50	400	166.12	352	69.73	354	7.07	$\leq$ 358
d5	50	500	205.39	451	95.40	453	11.62	$\leq$ 452
e1	100	200	48.32	103	28.37	103	1.29	$\leq$ 102
e2	100	400	137.70	303	72.45	303	6.74	$\leq$ 304
e3	100	600	232.34	505	142.63	506	16.69	$\leq$ 520
e4	100	800	323.19	699	219.34	702	30.47	$\leq$ 709
e5	100	1000	404.82	906	136.92	906	51.59	$\leq$ 908
f1	500	1000	237.79	510	395.25	515	42.01	$\leq$ 520
f2	500	2000	678.83	1521	552.33	1521	241.58	$\leq$ 1553
f3	500	3000	1109.97	2539	3602.95	2541	782.47	$\leq$ 2551
f4	500	4000	1616.21	3533	3242.95	3533	987.47	$\leq$ 3541
f5	500	5000	2063.82	4514	5678.42	4519	1659.88	$\leq$ 4553
g1	1000	2000	479.95	1043	786.30	1045	208.05	$\leq$ 1177
g2	1000	4000	1368.76	3043	4779.81	3043	1112.78	$\leq$ 3209
g3	1000	6000	2274.75	5060	5094.37	5060	2563.94	$\leq$ 5136
g4	1000	8000	3157.54	7053	7250.78	7053	5080.02	$\leq$ 7150
g5	1000	10000	4143.82	9038	10818.13	9038	7332.10	$\leq$ 9062

6. Concluding remarks

We presented some preliminary investigations about an apparently new problem arising from the automatic scheduling of hypermedia documents. The latter was formulated as a mixed 0-1 integer programming problem. A simple primal heuristic based on variable fixations along the solution of a sequence of linear relaxations of the mixed integer programming formulation is proposed and described. Computational experiments on real size instances illustrate the effectiveness of this heuristic in finding suboptimal solutions within very small processing times. The approach seems to be quite promising not only in algorithmic terms, but also concerning its integration within existing document formatting systems.

We have also discussed results obtained by Lagrangean relaxation. A dual heuristic using the dual multipliers and modified costs consistently improved the solutions obtained by the primal heuristic.

Several extensions of this work are being carried out. We first expect to replace CPLEX by a customized linear programming algorithm to solve the linear relaxation $\overline{\text{ASHD}}$, which might lead to dramatic reductions in computation times of both the primal and dual heuristics. We recall that such improvements are quite important, due to the real-time nature of the problem to be solved. We are also investigating improved construction procedures and local search strategies based on variable complementation and exchange to improve the solutions generated by the primal heuristic.

As a broader goal, we envision the integration of such heuristics and the bi-criteria optimization approach within an existing document formatter to perform compile time scheduling and even run time adjustments.

References

[1] F. Barahona and R. Anbil, "The volume algorithm: Producing primal solutions with a subgradient method", Research Report RC 21103, IBM T.J. Watson Research Center, 1997, to appear in *Mathematical Programming*.

[2] F. Bes, M. Jourdan, N. Layaïda, C. Roisin, L. Tardif, T.T. Thuong, and L. Villard, "MADEUS: An authoring environment for multimedia documents", online document, `http://www.inrialpes.fr/opera/Madeus.en.html`, last visited on February 22, 2001.

[3] M.C. Buchanan and P.T. Zellweger, "Automatically generating consistent schedules for multimedia documents", *Multimedia Systems Journal* 20 (1993), 55–67.

[4] M.C. Buchanan and P.T. Zellweger, "Automatic temporal layout mechanisms", *Proceedings of the First ACM Internacional Conference on Multimedia*, 341–350, 1993.

[5] A. Caprara, M. Fischetti, and P. Toth, "A heuristic method for the set covering problem", *Operations Research* 47 (1999), 730–743.

[6] M. Held and R.M. Karp, "The traveling salesman problem and minimum spanning trees: Part II", *Mathematical Programming* 1 (1971), 6–25.

[7] M. Held, P. Wolfe, and H.P. Crowder, "Validation of subgradient optimization", *Mathematical Programming* 6 (1974), 62–88.

[8] M. Jourdan, N. Layaïda, and C. Roisin, "A survey on authoring techniques for temporal scenarios of multimedia documents", *Handbook of Internet and Multimedia Systems and Applications, Part 1: Tools and Standards* (B. Furht, ed.), CRC Press, 1998.

[9] M. Jourdan, N. Layaïda, and L. Sabry-Ismail, "Presentation services in MADEUS: An authoring environment for multimedia documents", Rapport de recherche RR-2983, INRIA, 1997.

[10] M.Y. Kim and J. Song, "Multimedia documents with elastic time", *Proceedings of the Third ACM Internacional Conference on Multimedia*, 143–154, 1995.

[11] P. Mahey, personnal communication, 1999.

[12] P. Mahey and B. Bachelet, personnal communication, 1999.

[13] M. Minoux, *Programmation Mathématique*, Dunod, 1983.

[14] R.F. Rodrigues, L.F.G. Soares, and G.L. Souza, "Authoring and formatting of documents based on event-driven hypermedia models", *Proceedings of the IEEE Conference on Protocols for Multimedia Systems and Multimedia Networking*, Santiago, 74–83, 1997.

(M.T. Medina) Department of Computer Science, Catholic University of Rio de Janeiro, R. Marquês de São Vicente, 225, Rio de Janeiro, RJ 22453-900 Brazil

E-mail address, M.T. Medina: maira@inf.puc-rio.br

(C.C. Ribeiro) Department of Computer Science, Catholic University of Rio de Janeiro, R. Marquês de São Vicente, 225, Rio de Janeiro, RJ 22453-900 Brazil

E-mail address, C.C. Ribeiro: celso@inf.puc-rio.br

(L.F.G. Soares) Department of Computer Science, Catholic University of Rio de Janeiro, R. Marquês de São Vicente, 225, Rio de Janeiro, RJ 22453-900 Brazil

E-mail address, L.F.G. Soares: lfgs@inf.puc-rio.br